Social Malcode

Benutzerabhängige Schadprogramme

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
der Universität Mannheim

vorgelegt von

Markus Engelberth
aus Waldbröl

Mannheim, 2013

Dekan:	Professor Dr. Heinz Jürgen Müller, Universität Mannheim
Referent:	Professor Dr. Felix Freiling, Friedrich-Alexander-Universität Erlangen-Nürnberg
Korreferent:	Professor Dr. Thorsten Holz, Ruhr-Universität Bochum

Tag der mündlichen Prüfung: 11. September 2013

Für meine Eltern: Gerd Engelberth und meine leider viel zu früh verstorbene Mutter Carmen Engelberth.
»… Vielen Dank für alles!«

Zusammenfassung

In den letzten Jahren ist ein Wandel bezüglich der Verbreitungsmethoden von Schadprogrammen, wie Viren, Würmern und Trojanern, zu verzeichnen. Während sich frühere Schadprogramme vermehrt voll automatisiert durch das Ausnutzen von Sicherheitslücken verbreiteten, rückt diese Methode der Verbreitung heutzutage aufgrund von immer sichereren Kommunikationsprotokollen und Betriebssystemen in den Hintergrund. Stattdessen nutzen Schadprogramme reguläre Kommunikationswege, wie z. B. E-Mails oder Instant Messaging Programme, für ihre Verbreitung. Die Autoren der Schadprogramme sind jedoch auf die *Mithilfe* ihrer Opfer angewiesen: Sie müssen diese durch Social Engineering dazu bewegen, beispielsweise den Anhang einer E-Mail auszuführen oder einem zugeschickten Verweis zu folgen, damit solch ein Schadprogramm den oder die Rechner eines Opfers infizieren kann.

Diese immer häufiger anzutreffende Art von Schadprogrammen haben wir *Social Malcode* genannt. In der vorliegenden Dissertation haben wir die Verbreitung solcher Schadprogramme näher untersucht. Neben einer formalen Definition von Social Malcode, haben wir deren Verbreitung simuliert und die Ergebnisse durch anschauliche Verbreitungsverläufe visualisiert. Die dafür notwendigen Simulationsparameter, die unter anderem das menschliche Verhalten von Computerbenutzern modellieren, haben wir durch umfangreiche Versuche und Skripte experimentell bestimmt. Zusammenfassend kann man festhalten, dass die Verbreitung von Social Malcode von wesentlich mehr Faktoren abhängt als die Verbreitung autonomer Schadprogrammen. So verbreitet sich Social Malcode beispielsweise aufgrund der Abhängigkeit von menschlichen Handlungen wesentlich langsamer.

Abstract

A change in propagation methods of malware like viruses, worms, or trojan horses can be observed in the recent years. While early malware mainly propagated by exploiting vulnerabilities in a fully automated manner, nowadays this propagation method loses ground, due to more secure communication protocols and more secure operation systems. Instead todays malware uses regular communication channels, like email oder instant messaging applications, to infect new machines. This propagation method depends on some sort of user interaction: The authors of malware have to persuade their victims to, e. g., execute an email attachment or to follow a link – thereby the users' machines will be infected with the malware. Often this user assistance is tried to be enforced by means of social engineering.

This more and more common sort of malware has been denoted by us as *Social Malcode*. In this work we have examined its propagation closely. Besides a formal definition of Social Malcode, we also have simulated its propagation and visualized the simulation results employing propagation graphs. We obtained the needed simulation parameters values, which among other things model the human behavior of computer users, by the use of extensive experiments and self written scripts. In summary, it can be stated that the propagation of Social Malcode depends on more factors than the propagation of autonomous spreading malware. Thus, Social Malcode spread much slower, for example, due to the dependence on human actions.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Code-Blöcke

Abkürzungsverzeichnis

API .... **A**pplication **P**rogramming **I**nterface (Seite 125)
Schnittstelle zur Anwendungsprogrammierung

ASLR .. **A**ddress **S**pace **L**ayout **R**andomization (Seite 1)
Schutzfunktion, die den Aufbau des Adressraums zufällig gestaltet

BHO ... **B**rowser **H**elper **O**bject (Seite 61)
Plug-in zur Erweiterung der Funktionalität des Internet Explorer

CPU .... **C**entral **P**rocessing **U**nit (Seite 21)
Hauptprozessor eines Computers

DDoS .. **D**istributed **D**enial **o**f **S**ervice (Seite 60)
Verteilter Angriff auf die Verfügbarkeit eines Systems

DEP .... **D**ata **E**xecution **P**revention (Seite 1)
Verhindert die Ausführung von Code aus Seiten des Arbeitsspeichers, die als nicht ausführbar gekennzeichnet sind

DNS ... **D**omain **N**ame **S**ystem (Seite 51)
Dienst zur Umsetzung von Domainnamen in IP-Adressen

DOM ... **D**ocuments **O**bject **M**odels (Seite 20)
Schnittstelle für den Zugriff auf HTML- oder XML-Dokumente

DoS .... **D**enial **o**f **S**ervice (Seite 60)
Angriff auf die Verfügbarkeit eines Systems

HTML .. **H**yper**t**ext **M**arkup **L**anguage (Seite 22)
Hypertext-Auszeichnungssprache

HTTP .. **H**yper**t**ext **T**ransfer **P**rotocol (Seite 77)
Übertragungsprotokoll der Anwendungsschicht des ISO/OSI-Schichtenmodells

ID ..... **Id**entifikationsnummer (Seite 78)
Eindeutiges Merkmal eines Objekts/Subjekts

IDE .... **I**ntegrated **D**evelopment **E**nvironment (Seite 125)
Integrierte Entwicklungsumgebung

IM ..... **I**nstant **M**essaging (Seite 24)
Textbasierte Kommunikationsmethode

IMAP .. **I**nternet **M**essage **A**ccess **P**rotocol (Seite 77)
Übertragungsprotokoll zum Zugriff auf E-Mails auf einem Mailserver

IP ..... **I**nternet **P**rotocol (Seite 126)
Übertragungsprotokoll der Vermittlungsschicht des ISO/OSI-Schichtenmodells

ISP ..... **I**nternet **S**ervice **P**rovider (Seite 18)
Anbieter eines Internetdienstes

IT ..... **I**nformations**t**echnik (Seite 3)
Oberbegriff für die Informations- und Datenverarbeitung

MD5 ... **M**essage-**D**igest Algorithm **5** (Seite 88)
Kryptographische Hashfunktion

MIME .. **M**ultipurpose **I**nternet **M**ail **E**xtensions (Seite 118)
Standard zur Beschreibung des Typs eines E-Mail-Dateianhangs/Parts

MITM .. **M**an **I**n **T**he **M**iddle (Seite 109)
Transparentes Umleiten des Datenverkehrs über einen Server des Angreifers

NAT .... **N**etwork **A**ddress **T**ranslation (Seite 20)
Übersetzung zwischen öffentlichen und privaten IP-Adressen

NED ... **Ne**twork **D**escription (Seite 126)
Netzbeschreibungssprache des Simulations-Frameworks OMNeT++

OMNeT++ **O**bject-oriented **M**odular **Ne**twork **T**estbed in C**++** (Seite 125)
Objektorientierte, modulare Testumgebung in C++ für Netze

OS ..... **O**perating **S**ystem (Seite 126)
Betriebssystem

PC ..... **P**ersonal **C**omputer (Seite 17)
Persönlicher Computer (Einzelplatzrechner)

PHP .... **P**HP: **H**ypertext **P**reprocessor (Seite 48)
Skriptsprache (hauptsächlich zur Erstellung von Webanwendungen verwendet)

POP3 ... **P**ost **O**ffice **P**rotocol, Version **3** (Seite 77)
Übertragungsprotokoll zum Abholen von E-Mails von einem Mailserver

SHA ... **S**ecure **H**ash **A**lgorithm (Seite 97)
Klasse von standardisierten kryptologischen Hash-Funktionen

SMTP .. **S**imple **M**ail **T**ransfer **P**rotocol (Seite 23)
Übertragungsprotokoll zum Erzeugen und Weiterleiten von E-Mails zwischen Mailservern

TCP .... **T**ransmission **C**ontrol **P**rotocol (Seite 126)
Übertragungsprotokoll der Transportschicht des ISO/OSI-Schichtenmodells

TLD .... **T**op **L**evel **D**omain (Seite 119)
Endung von Internetadressen und damit höchste Ebene der Namensauflösung

UDP ... **U**ser **D**atagram **P**rotocol (Seite 126)
Übertragungsprotokoll der Transportschicht des ISO/OSI-Schichtenmodells

URL ... **U**niform **R**esource **L**ocator (Seite 39)
Eindeutige Identifikation/Adressierung einer Ressource in einem Computernetz

VBA ... **V**isual **B**asic for **A**pplications (Seite 80)
Skriptsprache (entwickelt für Microsoft-Office-Programme)

VoIP .... **V**oice **o**ver **IP** (Seite 24)
Telefonie über Computernetze

WWW .. **W**orld **W**ide **W**eb (Seite 19)
Ein über das Internet abrufbares Hypertext-System

XML ... E**x**tensible **M**arkup **L**anguage (Seite 49)
Erweiterbare Auszeichnungssprache

KAPITEL I

Einleitung

Dieses Kapitel stellt den Einstieg in das Thema der Dissertation dar, die sich inhaltlich mit einer ganz bestimmten Klasse von Schadprogrammen beschäftigt: Fokus der Arbeit sind Schadprogramme, deren Verbreitung (im Sinne von Infektionen noch nicht durch das Schadprogramm befallener Rechner) von dem Verhalten der Benutzer der neu zu infizierenden Rechner abhängig ist.

In dieser Einleitung werden wir in Abschnitt 1.1 das Thema der benutzerabhängigen Schadprogramme motivieren. Der folgende Abschnitt 1.2 liefert eine kurze, informelle Definition des Begriffs *Social Malcode*, der zugleich auch Titel dieser Dissertation ist. Die Ziele der Dissertation werden in Abschnitt 1.3 festgelegt und in Abschnitt 1.4 geben wir einen Überblick über die von uns erzielten Ergebnisse. Bevor in Abschnitt 1.6 ausgewählte Veröffentlichungen des Autors präsentiert werden, stellt Abschnitt 1.5 den weiteren Verlauf dieser Arbeit vor. Dabei wird kurz umrissen, welche Themen die einzelnen Kapitel der Arbeit behandeln und wie diese im Gesamtzusammenhang der Dissertation einzuordnen sind. Abgeschlossen wird diese Einleitung durch einen Abschnitt, in dem allen mehr oder minder an der Erstellung dieser Arbeit beteiligten Personen gedankt wird, ohne die diese Arbeit nicht so geworden wäre, wie sie jetzt ist.

1.1. Motivation

Immer häufiger laufen gezielte Angriffe über sogenannte *Client-Side-Exploits* ab. Diese Art von Angriffen wird in letzter Zeit immer beliebter, um gezielt in ein Firmennetz einzudringen. Dies liegt vor allem an den immer sicherer werdenden Betriebssystemen und Netzdiensten. Schützende Technologien wie zum Beispiel *Data Execution Prevention* (DEP), *Address Space Layout Randomization* (ASLR) und Sandboxing machen das Ausnutzen von Sicherheitslücken immer anspruchsvoller [EL13]. Somit bieten Betriebssysteme und Netzdienste einem potentiellen Angreifer immer weniger Angriffsfläche. In gleichem Maße steigt auch die Effektivität von Antiviren-Lösungen und es werden immer ausgefeiltere Firewallsysteme entwickelt, die das Kompromittieren der Rechner eines Netzes durch einen Angreifer erschweren.

Doch was passiert, wenn Angreifer diese Absicherungen umgehen und versuchen, das möglicherweise schlechte Sicherheitsbewusstsein der Mitarbeiter auszunutzen, um darüber in ein Firmennetz einzudringen? Die besten und teuersten Sicherheitskonzepte, die das Netz und die Server eines Unternehmens vor Angriffen schützen sollen, bieten in diesen Fällen nicht mehr unbedingt einen Schutz. Erhalten Mitarbeiter eines Unternehmens beispielsweise E-Mails mit bösartigen Dateianhängen oder Verweisen auf bösartig präparierte Internetseiten, stellt dies bei einem unbedachten Mitarbeiter eine große Gefahr dar. Werden diese Anhänge heruntergeladen und ausgeführt oder folgt ein Mitarbeiter solch einem Verweis, ist die Gefahr groß, dass danach der Rechner des Mitarbeiters kompromittiert ist.

Studien zeigen, dass sich das Angreiferverhalten immer mehr in diese Richtung verlagert. Vor wenigen Jahren schrieben Mitarbeiter von *Microsoft* bezüglich dieser Entwicklung sogar von einer Malware-Revolution. In ihrem Artikel *Malware Revolution: A Change in Target* [Lan07] betonen sie explizit die Verlagerung der Angriffsziele:

> *„A significant evolution has occurred in the malware landscape over the past five years* [...]. *But in the past year, a more abrupt shift has taken place: a change in target, with users squarely in the bulls-eye.“*

Vor allem der letzte Satz verdeutlicht, dass immer häufiger der Mensch im Fokus aktueller Angriffe steht und nicht mehr nur der Computer an sich. Und auch ganz aktuelle Vorhersagen für das Jahr 2013 belegen, dass diese Entwicklung stetig voranschreitet. So geht James Lyne, Director of Technology Strategy von *Sophos*, zwar nicht davon aus, dass Exploits völlig von der Bildfläche verschwinden werden, es wird jedoch einen erheblichen Rückgang geben. Dieser könne aber von einem starken Anstieg im Bereich Social Engineering-Attacken über verschiedenste Plattformen hinweg mehr als ausgeglichen werden [EL13].

Neben Angriffen auf Wirtschaftsunternehmen wurde dieser Angriffsvektor auch in gezielten Angriffen gegen Regierungsbehörden, das Militär oder andere öffentliche Einrichtungen genutzt. Beispielsweise beruhten etliche Angriffe aus den Jahren 2007 und 2008 gegen europäische Regierungen und amerikanische Verteidigungsorganisationen auf Angriffen mit bösartig präparierten Microsoft Office Dokumenten [EFG+09b].

Aber nicht nur Schadprogramme eines gezielten Angriffs sind bei ihrer Verbreitung auf eine „Mithilfe“ der Opfer angewiesen. Auch immer mehr große Botnetze, deren Ziel es ist, möglichst viele Rechner zu infizieren, verwenden Verbreitungsroutinen, die eine Benutzerinteraktion erfordern. Die Betreiber solcher Botnetze legen in der Regel keinen großen Wert darauf, ob die befallenen Rechner einer abgegrenzten, geographischen Region oder einem bestimmten Unternehmen zuzuordnen sind. Ziel des Betreibers sind also möglichst viele Neuinfektionen, damit ihm noch mehr Rechenkapazität durch das Botnetz zur Verfügung steht. Viele aktuelle Botnetze nutzen mittlerweile mehrere Verbreitungsroutinen, die sowohl mit als auch ohne Benutzerinteraktion auskommen. Zwei solcher aktuellen Botnetze und deren Verbreitungsroutinen werden in den Abschnitten 3.2.1 und 3.2.2 genauer betrachtet.

Analysiert man die aktuell häufigsten Bedrohungen und deren Verbreitungsstrategien, so erkennt man sogar, dass alle diese Bedrohungen mindestens eine Verbreitungsroutine besitzen, die auf eine Benutzerinteraktion angewiesen ist. Tabelle 1.1 enthält

die 10 häufigsten Bedrohungen im Februar 2013. Ermittelt wurde diese Liste von *Eset*, einem Unternehmen für Sicherheitssoftware [Har13]. Die in der Tabelle angegebenen Prozentzahlen geben den Anteil der jeweiligen Bedrohung an der Menge aller durch *Eset* beobachteten Bedrohungen an. Zwar sind in der Auflistung auch einige Bedrohungen zu finden, die sich komplett autonom verbreiten können, wie zum Beispiel der Computerwurm *Conficker*, jedoch benutzt jeder dieser Bedrohungen mindestens eine Verbreitungsroutine, die auf eine Benutzerinteraktion angewiesen ist.

Tabelle 1.1.: Die häufigsten Bedrohungen im Februar 2013
Die Tabelle enthält die laut *Eset* häufigsten Bedrohungen im Februar 2013 [Har13]. Jede einzelne Bedrohung enthält mindestens eine Verbreitungsroutine, die einen manuellen Eingriff seitens der Computerbenutzer erfordert.

Rang	Bedrohung	Anteil	Verbreitungsroutine mit Benutzerinteraktion
1	INF/Autorun	3,32 %	✓
2	HTML/Iframe.B	2,99 %	✓
3	Win32/Sality	2,17 %	✓
4	HTML/ScrInject.B	1,96 %	✓
5	Win32/Dorkbot	1,81 %	✓
6	Win32/Ramnit	1,74 %	✓
7	Win32/Conficker	1,39 %	✓
8	Win32/Qhost	1,31 %	✓
9	JS/TrojanDownloader.Iframe.NKE	0,84 %	✓
10	Win32/Virut	0,79 %	✓

Die Bedeutung und somit auch die Gefahr, die von dieser Art Schadprogrammen ausgeht, hat also schlagartig zugenommen. Die Schadprogramme im Fokus dieser Dissertation stellen somit eine nicht vernachlässigbare und sehr ernst zu nehmende Bedrohung für die Sicherheit der heutigen IT-Landschaft dar.

1.2. Thema der Dissertation

Das Thema der Dissertation sind Schadprogramme, die für eine erfolgreiche Verbreitung auf ein manuelles Eingreifen der Opfer in deren Verbreitungsprozess angewiesen sind. Wie im späteren Verlauf der Dissertation ersichtlich wird, ist dies immer mit einer Form von *Social Engineering* verbunden, wodurch der Angreifer das Opfer zu diesem Eingreifen bewegen/verleiten möchte.

Aus diesem Grund haben wir der von uns betrachteten Klasse von Schadprogrammen den Namen *Social Malcode* gegeben. Das erste Wort *Social* spiegelt den Aspekt des Social Engineering wider und *Malcode* ist ein Kunstwort aus den beiden englischen Begriffen *malicious* (dt. *bösartig*) und *code*. Bei Social Malcode handelt es sich also um bösartigen Code, dessen erfolgreiche Verbreitung davon abhängt, ob ein Angreifer seine Opfer dazu bewegen kann, eine bestimmte Aktion auszuführen und

somit die Verbreitung erst zu ermöglichen. Meist verwendet ein Angreifer für diese Beeinflussung verschiedenste Techniken und psychologische Manipulationen aus dem Bereich des Social Engineering.

1.3. Ziele der Dissertation

Ziel der Dissertation ist es, die Verbreitung von Social Malcode näher zu untersuchen. Wie bei sich autonom verbreitenden Schadprogrammen soll im Rahmen der Dissertation ein parametrisiertes Modell entstehen, welches die Verbreitung von Social Malcode beschreibt. Im Gegensatz zu sich autonom verbreitenden Schadprogrammen spielt bei der Verbreitung von Social Malcode definitionsgemäß der Mensch eine wesentliche Rolle (siehe Kapitel 3). Genau in diesem Faktor Mensch besteht die größte Schwierigkeit bei der Modellierung der Verbreitung von Social Malcode, da sich das Handeln eines Menschen schlecht bis kaum formalisieren und vorhersagen lässt. Aus diesem Grund werden folgende Ziele für die Dissertation festgelegt:

1. Da der Fokus der Dissertation auf einer bestimmten Sorte von Schadprogrammen liegt, soll der zentrale Begriff *Social Malcode* formalisiert werden. Diese Formalisierung soll eine klare und eindeutige Abgrenzung gegenüber anderen Schadprogrammen erlauben, die nicht in die Kategorie der benutzerabhängigen Schadprogramme fallen.
2. Da das Verhalten eines Benutzers in die Verbreitung von Social Malcode involviert ist, muss auch dieses Verhalten innerhalb des Verbreitungsprozesses formal beschreibbar sein. Um dieses Problem in den Griff zu bekommen, sollen Modelle von Verbreitungsroutinen erstellt werden, in einer ähnlichen Form wie sie auch für endliche Automaten benutzt werden. Das Benutzerverhalten soll in diesen Modellen gekapselt sein. Zustände der Modelle sind charakteristische Systemzustände der am Verbreitungsprozess beteiligten Systeme und die Übergänge dazwischen repräsentieren Aktionen der Benutzer (z. B. *Benutzer folgt einem Verweis*) oder externe Ereignisse (z. B. *eine neue Spam-Mail trifft ein*).
3. Damit später das parametrisierte Modell verwendet werden kann, müssen die beteiligten Parameter mit Werten belegt werden. Deshalb muss das Benutzerverhalten quantifiziert beziehungsweise gemessen werden, um somit die Werte für die Modellparameter zu erhalten.
4. Durch eine abschließende Simulation sollen die Verbreitungsverläufe eines Social Malcode visuell dargestellt werden. Dazu werden die Parameter mit den vorher bestimmten Werten gefüllt und in Form eines Programms einem Netzwerk-Simulator als Eingabe übergeben. Des Weiteren soll auf diese Weise auch der Einfluss der einzelnen Parameter auf die gesamte Verbreitung des Social Malcode untersucht werden.

1.4. Überblick über die Ergebnisse der Dissertation

Dieser Abschnitt liefert einen Überblick über die Ergebnisse, die wir im Rahmen unserer Arbeit erzielt haben. Da sich die Ergebnisse durch die vier, zuvor gesteckten Ziele

der Arbeit ergeben, haben wir hier den Überblick über die Ergebnisse unserer Arbeit anhand dieser Ziele strukturiert:

1. **Formalisierung des Begriffs Social Malcode** (*Ergebnis des 1. Ziels*):
   Im 3. Kapitel der Arbeit geben wir eine formale Definition des Begriffs *Social Malcode*. Dafür definieren wir zuerst die Begriffe *Ressource*, *Code*, *nicht-autorisierter Zugriff*, *Benutzerinteraktion*. *Schadcode* und *Verbreitungsroutine*. Mit Hilfe dieser Definitionen, die die Grundlagen für Social Malcode formalisieren, präsentieren wir letztendlich eine formale Definition von Social Malcode.

   Social Malcode ist ein Schadcode, bestehend aus einer Menge von Schad- und Verbreitungsroutinen. Wenn die Menge der Verbreitungsroutinen nicht leer ist, muss mindestens eine Verbreitungsroutine eine Sequenz von drei Zustandsübergängen enthalten, die sich kausal bedingen. Diese Zustandsübergänge repräsentieren eine *Herausforderung einer Benutzerinteraktion*, eine *Benutzerinteraktion* und eine *Code-Ausführung*, die letztendlich die eigentliche Infektion des zugrunde liegenden Systems verursacht. Die Definition beinhaltet jedoch auch Schadcodes, die sich nicht aktiv verbreiten – bei denen also die Menge der Verbreitungsroutinen leer ist. Folglich ist zum Beispiel auch eine Phishing-Seite laut der von uns präsentierten Definition Social Malcode.

2. **Modelle von Verbreitungsroutinen** (*Ergebnis des 2. Ziels*):
   Im Zuge der Formalisierung des Begriffs Social Malcode haben wir ebenfalls den Begriff der Verbreitungsroutine eines Schadprogramms M formalisiert: Eine Verbreitungsroutine ist ein endlicher, gerichteter Graph, dessen Knotenmenge aus Zuständen der am Verbreitungsprozess beteiligten Systeme besteht und dessen Kantenmenge Übergänge zwischen diesen Systemzuständen enthält. Innerhalb des Graphen muss ein Zyklus existieren, der eine ausgewiesene Kante beinhaltet, dessen Start- und Endzustand Zustände zweier, unterschiedlicher Systeme sind. Diese Kante repräsentiert somit den Übergang des Schadprogramms M von einem System zum nächsten. Dabei ist das System, dem der Startzustand zuzurechnen ist, ein bereits durch das Schadprogramm M infiziertes System, das ein anderes System, dem der Endzustand der ausgewiesenen Kante zuzuordnen ist, mit dem Schadprogramm M infiziert. Neben der genauen Definition einer Verbreitungsroutine befinden sich in Kapitel 3 auch einige Beispiele, in denen Verbreitungsroutinen grafisch dargestellt sind – in einer ähnlichen Form wie sie auch für endliche Automaten benutzt wird.

3. **Benutzerverhalten quantifizieren/messen** (*Ergebnis des 3. Ziels*):
   Um das Verhalten von Computerbenutzern in Situationen zu erfassen, die möglicherweise zu einer Infektion mit Social Malcode führen können, haben wir verschiedene Experimente durchgeführt. Beispielsweise haben wir in Kooperation mit einem IT-Sicherheitsunternehmen den Mitarbeitern mehrerer deutscher Unternehmen E-Mails zugeschickt und gemessen, wie diese auf den Erhalt der E-Mails reagieren. Dabei zeigte sich, dass über 15 % der Empfänger dem Verweis innerhalb der von uns verschickten E-Mails folgten oder etwa dass fast 20 % der Empfänger sensible Daten auf einer von uns aufgesetzten Phishing-Seite eingaben.

Des Weiteren haben wir im Rahmen der Dissertation ein Skript geschrieben, das auf einem Mailserver die Zeitspannen zwischen dem Eintreffen und dem Lesen einer E-Mail auslesen kann. Dieses Skript haben wir auf den Mailserver der Univeristät Mannheim angewandt. Eine Analyse der Zeitspannen zeigte, dass diese bei den E-Mail-Konten, auf welche unser Skript angewandt werden durfte, einer Verteilung nach einem Potenzgesetz folgt. Der Großteil der Benutzer liest seine E-Mails also recht schnell und einige wenige E-Mails bleiben sehr lange ungelesen.

Auch die Zeitspannen zwischen einer Rechnerinfektion und -säuberung spielen bei der Verbreitung von Social Malcode eine Rolle. Um diese zu bestimmen, haben wir Daten zweier großer Keylogger-Familien untersucht. Dabei zeigte sich, dass diese Zeitspannen in den uns vorliegenden Keylogger-Daten einer Exponentialverteilung folgen.

Neben dem Verhalten potentieller Opfer ist auch das Verhalten eines Angreifers relevant für die Verbreitung von Social Malcode. Deshalb haben wir über 17 Millionen Spam-Mails analysiert und festgestellt, dass der häufigste Angriffsvektor in der Verwendung von Verweisen liegt oder dass im Schnitt 2,3 Empfängern eine initiale Spam-Mail zugeschickt wird.

4. **Simulation von Social Malcode** (*Ergebnis des 4. Ziels*):
   Abschließend haben wir das Modell einer Verbreitungsroutine und das gemessene Benutzerverhalten benutzt, um die Verbreitung eines Social Malcode zu simulieren. Dafür haben wir einen E-Mail-Wurm simuliert, der sich über Verweise auf eine schadhaft präparierte Internetseite verbreitet. Die Gesamtpopulation unserer Simulationen bestand aus 50 000 Rechnern und die Verteilung der Adressbuchgrößen dieser Rechner folgt einem Potenzgesetz.

   Unsere Simulationen haben gezeigt, dass die Anzahl der infizierten Simulationen proportional von der Anzahl der Rechner abhängt, die initial eine Spam-Mail in ihrem Posteingang haben. Ebenfalls steigt mit den Wahrscheinlichkeiten, mit denen Benutzer eine neue E-Mail lesen und mit denen sie dem darin enthaltenen Verweis folgen, die Anzahl der Infektionen. Die Zeitspannen zwischen dem Eintreffen und dem Lesen einer E-Mail stellen eine menschlich verursachte Verzögerung in dem Verbreitungsprozess des simulierten E-Mail-Wurms dar. Je länger E-Mails ungelesen in einem Posteingang verweilen, desto langsamer verbreitet sich auch der E-Mail-Wurm. Eine ebenfalls von uns untersuchte Verzögerung ist die Manipulation der Zustellgeschwindigkeit von E-Mails. Diese wird jedoch nicht durch die Empfänger verursacht, sondern kann bewusst zur Eindämmung von Social Malcode benutzt werden. Auch hier gilt: Je höher diese Verzögerung gewählt wird, desto langsamer verbreitet sich der E-Mail-Wurm. Je niedriger die Zeitspannen zwischen einer Rechnerinfektion und -säuberung ausfallen, desto weniger Rechner infizieren sich im Schnitt mit dem simulierten Wurm. Die Exponentialverteilung dieser Zeitspannen spiegelt sich auch in einem Querschnitt durch die Verbreitungsverläufe wider. Durch die Wahrscheinlichkeit, mit der ein Rechner nach einer Infektion immun gegen den Wurm wird, ergibt sich eine Senke im 3-dimensionalen Verbreitungsverlauf des simulieren E-Mail-Wurms: Je höher diese Wahrscheinlichkeit ist, desto schneller sinkt im Verlauf der Simulation die Anzahl der Rechner, die noch anfällig für den Wurm

sind. Bei einer konstanten Wahrscheinlichkeit ergibt sich im Verlauf der Simulation auch eine fallende Anzahl an anfälligen Rechnern, da sich nach jeder Infektion eines einzelnen Rechners entscheidet, ob dieser nun immun gegen den Wurm ist. Zuletzt haben wir noch den Effekt einer gezielten Immunisierung zentraler Rechner des Simulations-Netzes untersucht. Es zeigte sich, dass eine gezielte Immunisierung nur in einem gewissen Ausmaß sinnvoll ist: Werden zu wenige Rechner immunisiert, wirkt sich dies nicht auf die Verbreitung des E-Mail-Wurms aus. Werden zu viele Knoten immunisiert, ist dies mit einem zu hohen Immunisierungs-Aufwand verbunden und die Immunisierung kann nicht mehr als gezielt bezeichnet werden.

1.5. Gliederung

Nach diesem Kapitel folgt mit Kapitel 2 eine Einführung in die begrifflichen Grundlagen des Themas. Diese ist hilfreich für das weitere Verständnis der Arbeit oder, je nach Vorwissen, sogar auch notwendig. Da im Fokus der Arbeit Schadprogramme stehen, deren Verbreitung von einer Benutzerinteraktion abhängen, liefert das Grundlagenkapitel zuerst einen Überblick über das Themengebiet der Schadprogramme. Anschließend werden deren mögliche Verbreitungswege genauer betrachtet. Auch werden diese Verbreitungswege, in Hinblick auf die vorläufige Definition von Social Malcode, danach klassifiziert, ob sie auf eine Benutzerinteraktion angewiesen sind. Weil solch eine Interaktion häufig durch Social Engineering provoziert wird, folgt anschließend ein Abschnitt, der dieses Thema behandelt.

Um nicht länger mit einer vorläufigen Definition zu arbeiten, wird in Kapitel 3 der Begriff Social Malcode durch eine formale Definition präzisiert (Ziel 1). Die Herleitung der Definition erfolgt in zwei Schritten: In einem ersten Schritt wird eine intuitive Definition gegeben. Diese besitzt jedoch noch einige Ungenauigkeiten, die nach der Präsentation von Beispielen und Gegenbeispielen erörtert werden. Aus dieser Diskussion heraus ergibt sich in einem zweiten Schritt eine genaue und endgültige Definition von Social Malcode. Ein wesentlicher Punkt der endgültigen Definition, der die Ungenauigkeiten der ersten Definition beseitigt, sind Modelle von Verbreitungsroutinen, in denen das Verhalten von Benutzern gekapselt ist (Ziel 2).

Damit später mit Hilfe eines Netzwerksimulators Verbreitungsverläufe von Social Malcode erstellt werden können, muss zuvor das Verhalten der Benutzer quantifiziert werden. Dieser Problematik widmet sich Kapitel 4. Es werden mehrere Experimente vorgestellt, deren Ziel es ist, das Verhalten von Benutzern zu bestimmen (Ziel 3) und durch Kennzahlen auszudrücken. Zum Beispiel wurden Spam-Experimente durchgeführt, um unter anderem den Anteil der Empfänger zu bestimmen, die einem in den Spam-Mails enthaltenen Verweis folgen oder eine angehängte Datei öffnen.

In Kapitel 5 wird eine Simulation der Verbreitung von Social Malcode durchgeführt. Die Simulation lässt sich durch verschiedene Parameter steuern, deren Wertebelegung sich durch die zuvor ermittelten Kennzahlen ergibt. Durch das parametrisierte Verhalten ergeben sich verschiedene Visualisierungen der Verbreitung, die den Einfluss der einzelnen Parameter verdeutlichen (Ziel 4).

Die erstellten Verbeitungsverläufe werden anschließend in Kapitel 6 mit den Ergebnissen anderer Arbeiten verglichen. Dafür geben wir zuerst einen Überblick über verwandte Arbeiten, die inhaltliche Parallelen zu unserer Arbeit aufweisen. Deren Ergebnisse und Erkenntnisse stellen wir dann unseren Ergebnissen gegenüber und beleuchten sowohl die Gemeinsamkeiten als auch die Unterschiede. In einem letzten Abschnitt diskutieren wir zudem die Limitierungen unserer Arbeit und die sich daraus ergebenden Ratschläge/Ideen, wie unsere Arbeit weitergeführt bzw. weiter vertieft werden könnte.

Kapitel 7 ist eine abschließende Zusammenfassung der Dissertation. In dieser wird ein kurzer Rückblick auf die einzelnen Kapitel gegeben und deren Kernaussagen und Ergebnisse kurz wiederholt.

1.6. Veröffentlichungen des Autors

Die Dissertation beruht teilweise auf veröffentlichten als auch auf nicht veröffentlichten Arbeiten des Autors. Auch haben weiterführende Gedanken und anschließende Diskussionen nach der Veröffentlichung einiger Arbeiten Einfluss auf den Inhalt der Dissertation genommen.

Die Arbeit *The InMAS Approach* [EFG+10] stellt das *Internet Malware Analyse System* (kurz *InMAS*) vor. Dies ist ein verteiltes und großflächiges System zum Sammeln (mittels Honeypots und Spamtraps) und Analysieren (vor allem dynamische Analysen) von Schadprogrammen. Durch InMAS wird zum Beispiel auch Social Malcode eingefangen, der sich per Spam-Mails verbreitet.

Über das Medium E-Mail verbreitet sich beispielsweise auch der Bot Waledac. In der Arbeit *Walowdac - Analysis of a Peer-to-Peer Botnet* [SGE+09] wird eine Beschreibung und tief gehende Analyse dieses Bots präsentiert. Die in dieser Arbeit veröffentlichten Ergebnisse beruhen auf einer Infiltration des Botnetzes.

Auch die Arbeit *MalOffice – Detecting malicious documents with combined static and dynamic analysis* [EWH09] behandelt Schadprogramme der Klasse Social Malcode. Es wird ein kombinierter Analyseansatz speziell für schadhaft präparierte Anwendungsdateien, wie zum Beispiel *Microsoft Word Dokumente* oder *Adobe PDF-Dateien*, vorgestellt. Diese werden häufig über Tauschbörsen oder E-Mails verbreitet.

Eine mögliche Lösung für das Problem der Verbreitung von Schadprogrammen durch E-Mails präsentiert die Arbeit *Mail-Shake* [EGGT09]. In der Arbeit wird eine Technologie zur Reduzierung von Spam-Mails vorgestellt. Diese beruht auf der Verwendung von öffentlichen und privaten E-Mail-Adressen. Letztere erhält ein Absender nur, indem er eine Art Handshake mit der öffentlichen Adresse durchführt.

Finanzielle Aspekte und Hintergründe zu digitalen Diebesgütern, d. h. durch Schadprogramme gestohlene Informationen und Daten, sind Inhalt der Arbeit *Learning More About the Underground Economy: A Case-Study of Keyloggers and Dropzones* [HEF09]. Im Rahmen dieser Arbeit haben wir zwei Keylogger-Familien und deren Funktionsweise analysiert. Auf diese Weise konnten wir nachvollziehen, wohin die mitgeschnittenen Daten verschickt wurden und diese ebenfalls näher untersuchen. Des Weiteren befinden sich in den mitgeschnittenen Daten der Keylogger Zeitstempel, die

Aufschluss über das Verhalten der Benutzer bezüglich der Säuberung eines infizierten Systems zulassen. Dieser Aspekt wird in Kapitel 4 noch eingehender untersucht.

1.7. Danksagung

Mein größter Dank gilt *Prof. Dr. Felix Freiling*, der mir erst durch die Anstellung an seinem Lehrstuhl die Erstellung dieser Arbeit ermöglichte. Vor allem möchte ich ihm für die stets sehr konstruktive und angenehme Betreuung danken. Durch viele Gespräche, sowohl mit den Mitarbeitern des Lehrstuhls *Praktische Informatik 1* der Universität Mannheim als auch mit Herrn Freiling persönlich, formte sich das Thema dieser Dissertation.

In gleichem Maße möchte ich mich deshalb bei jedem Mitarbeiter des Lehrstuhls für die wunderschönen Jahre in Mannheim bedanken, die nicht nur durch die berufliche Zusammenarbeit, sondern vor allem auch durch unzählige private Unternehmungen geprägt waren. Namentlich geht mein Danke also an: *Michael Becher, Zinaida Benenson, Sabine Braak, Andreas Dewald, Jan Göbel, Christian Gorecki, Thorsten Holz, Ralf Hund, Jürgen Jaap, Ioannis Krontiris, Martin Mink, Christian Moch, Philipp Trinius, Stefan Vömel* und *Carsten Willems*. Ein doppelter Dank gilt Thorsten Holz, der zu Beginn meiner Promotionszeit zu den Kollegen gehörte, und sich, mittlerweile als Lehrstuhlinhaber der Ruhr-Universität Bochum, bereit erklärt hat, als Zweitgutachter dieser Arbeit zu fungieren.

Ebenfalls dürfen in dieser Auflistung *Christian Lambertz, Moritz Martens, Christoph Minnameier, Peter Schillen, Patrick Schindler* und *Nils Semmelrock* nicht fehlen, die allesamt Kollegen an anderen Lehrstühlen waren oder sind und ohne die meine Zeit hier in Mannheim nur halb so schön gewesen wäre. Ebenso geht mein Dank an alle externen Kollegen und die viele studentischen Hilfskräfte, die ich hier aus Platzgründen nicht alle einzeln aufzählen kann.

Daneben sind viele andere Personen an der Erstellung der Arbeit beteiligt gewesen. Diese haben die vorliegende Arbeit, zumindest in dieser Form, erst ermöglicht. Deshalb möchte ich mich bei folgenden Personen bedanken:

- *Götz Schartner* und *René Mathes* des Unternehmens *8com GmbH & Co. KG*, durch deren Zusammenarbeit ich das Verhalten von Benutzern bei Erhalt einer E-Mail untersuchen konnte.
- *Christian Vjekoslav Tunjic* und *Dr. rer. nat. Heinz Kredel*, die es mir als Mitarbeiter des Rechenzentrums der Universität Mannheim ermöglichten, ein selbst geschriebenes Skript auf dem Mailserver der Universität Mannheim auszuführen. In diesem Zusammenhang gilt mein Dank ebenfalls allen Studenten und Mitarbeitern der Universität, die sich dazu bereit erklärt haben, dass dieses Skript auf ihrer Mailbox angewandt werden darf.
- *Chris Morrow*, der mir freundlicherweise Zugriff auf einen Spam-Korpus gewährt hat, nachdem unsere eigenen Spamtraps aufgrund eines Hardware-Defektes nicht mehr zugänglich waren.
- Zu guter Letzt möchte ich meinem Vater *Gerd Engelberth*, *Isabelle Häfner* und allen anderen Korrekturlesern danken, die wahrscheinlich schon Teile dieser Ar-

beit auswendig kennen. Ein besonderer Dank geht dabei an Isabelle, die mir in zahlreichen Gesprächen immer wieder neue Dankanstöße für die Arbeit gegeben hat.

Vielen lieben Dank!

KAPITEL II

Grundlagen

In diesem Kapitel werden die grundlegenden Begriffe und Konzepte erläutert, die zum Verstehen dieser Arbeit notwendig sind. Da Social Malcode eine spezielle Art von *Malware* (dt. *Schadprogramm*) ist, wird im ersten Abschnitt dieses Kapitels eine Übersicht über Schadprogramme präsentiert. Es wird erklärt, welche Klassen von Schadprogrammen existieren und wodurch sich diese voneinander abgrenzen. Aufgrund der großen Anzahl von unterschiedlichen Schadprogrammen ist diese Übersicht nicht vollständig, sondern stellt nur die wichtigsten und bekanntesten Schadprogrammarten vor.

Der zweite Abschnitt widmet sich einem speziellen Aspekt von Schadprogrammen: Der *Verbreitung* von Schadprogrammen. Die ersten bekannten Schadprogramme waren üblicherweise darauf angewiesen, dass ein Benutzer aktiv in den Verbreitungs- und Aktivierungsprozess eingreifen musste. Später hingegen verbreiteten sich Schadprogramme immer häufiger autonom, indem sie Schwachstellen in Diensten und in Betriebssystemen ausnutzten. Aktuell ist ein Trend festzustellen, der sich durch eine Kombination aus autonomer und benutzerabhängiger Verbreitung auszeichnet. In diesem Abschnitt werden die wichtigsten Methoden und Techniken vorgestellt, die Schadprogramme für eine Rechnerinfektion benutzen.

Die Schadprogramme, die im Fokus dieser Arbeit stehen, zeichnen sich dadurch aus, dass Social Engineering einen wesentlichen Einfluss auf deren Verbreitung nimmt. Entsprechend behandelt der dritte Abschnitt dieses Grundlagenkapitels das Thema *Social Engineering*. In diesem Abschnitt wird versucht, ein Gefühl dafür zu vermitteln, was Social Engineering ist, welche Motive für einen Angreifer bestehen, eine Social Engineering Attacke durchzuführen, welche Ziele ein Angreifer durch solch einen Angriff erreichen möchte und welche Methoden und Techniken dabei zum Einsatz kommen.

Abgeschlossen wird dieses Kapitel mit einer Zusammenfassung der präsentierten Grundlagen. Ein Vorstellung verwandter Arbeiten befindet sich in Abschnitt 6.1 der Arbeit. Solch ein Überblick über existierende Forschung auf dem gleichen Themengebiet wird oftmals bereits in dem Grundlagen- oder Hintergrundkapitel einer Arbeit gegeben. Jedoch haben wir uns dazu entschlossen, diesen Überblick erst später zu präsentieren. Dies liegt vor allem daran, dass wir die von uns erzielten Ergebnisse in

Bezug mit den Ergebnissen anderer Arbeiten setzen möchten. Dies macht natürlich erst nach der Vorstellung der eigenen Arbeit, und vor allem deren Ergebnissen, Sinn.

2.1. Malware

Der Begriff *Malware* ist ein Kunstwort und setzt sich aus Teilen der beiden englischen Wörter *malicious* (dt. *böswillig*) und *software* (dt. *Computerprogramm*) zusammen. Als Malware werden Computerprogramme bezeichnet, die teilweise oder ausschließlich eine unerwünschte und schädliche Funktionalität besitzen. Im restlichen Teil dieser Arbeit werden die Begriffe Malware und *Schädling*/*Schadprogramm* (dies ist die gängige deutsche Übersetzung) synonym für solch eine Art von Computerprogrammen verwendet. Mögliche Schadfunktionen von Schadprogrammen können zum Beispiel das Löschen von Dateien oder die Umgehung von installierten Sicherheitsprogrammen sein. Eine ebenfalls weit verbreitete und von Computerbenutzern unerwünschte Funktionalität ist das Ausspähen von Daten. Meist sind Schadprogramme so programmiert, dass die schadhafte Funktionalität dem Benutzer eines Computers verborgen bleibt und ohne dessen Einwilligung zur Ausführung kommt.

Ein konkretes Beispiel für die zuletzt genannte Schadroutine sind *Keylogger*. Dies sind Programme, die unbemerkt die Tastatureingaben eines Benutzers aufzeichnen und diese in regelmäßigen Abständen über das Internet an einen Angreifer schicken. Das häufigste Ziel von Keyloggern ist es, Zugangsdaten oder Kreditkarten- und Bankinformationen von ihren Opfern aufzuzeichnen und zu stehlen. Diese können anschließend von einem Angreifer benutzt werden, um sich selbst als eines seiner Opfer auszugeben oder sich finanziell zu bereichern. Ersteres bezeichnet man mit dem Begriff *Identitätsdiebstahl*. Durch den zweiten Verwendungszweck hat sich eine Art „Schwarzmarkt“ für elektronisches Diebesgut entwickelt, auf dem mit den gestohlenen Daten gehandelt wird und diese zu variierenden Preisen veräußert werden [HEF09]. Dieses Thema wird in Kapitel 4 noch eingehender diskutiert.

Das Entfernen eines Schadprogramms von einem infizierten System ist normalerweise mit einem hohen Aufwand verbunden, da es sich häufig sehr tief in das Computersystem einnistet. So lassen sich nur äußerst mühsam alle Teile eines Schadprogramms entfernen. Oft bleiben Fragmente eines Schadprogramms nach dem Entfernen im System zurück und können weiterhin Schaden anrichten. Der Reinigungsprozess wird zusätzlich dadurch erschwert, dass viele Schadprogramme ihre Bestandteile vor dem Benutzer und vor dem System verstecken. Dies ist beispielsweise dadurch möglich, dass Systembefehle durch die Schadprogramme so modifiziert werden, dass die Systembefehle bestimmte Dateien, Prozesse oder Objekte der Schadprogramme ignorieren beziehungsweise übersehen.

Unter dem Begriff *Malware* werden mehrere Arten von Computerprogrammen zusammengefasst. Gemein ist ihnen die schadhafte Funktionalität. Dennoch können sie nach ihrer Funktionsweise, dem Sinn und Zweck ihrer Ausführung für den Angreifer und nach der Art ihrer Verbreitungsroutine unterschieden werden. Meist fällt diese Unterscheidung schwer und ist nicht eindeutig. Dies liegt vor allem daran, dass die Grenzen zwischen den einzelnen Kategorien von Schadprogrammen fließend sind und ein Schadprogramm Eigenschaften mehrerer Kategorien aufweisen kann. Im Fol-

genden werden die am häufigsten anzutreffenden Arten von Schadprogrammen kurz vorgestellt. Die Liste der hier vorgestellten Schadprogrammarten ist aber keineswegs vollständig. Eine gute Übersicht und genauere Beschreibung von Schadprogrammen bieten unter anderem *Skoudis/Zeltser* [SZ04] und *Ligh et al.* [LAHR10].

2.1.1. Viren

Die Einteilung eines Schadprogramms in die Kategorie *Virus* geschieht aufgrund der Art und Weise, wie es sich verbreitet. Viren verbreiten sich über die Weitergabe von infizierten Dateien durch Benutzer. In der Regel sind Viren keine eigenständigen Programme und benötigen daher ein Wirtsprogramm. Viren können sowohl ausführbare Dateien befallen als auch nicht ausführbare Dokumentdateien, in die sie sich integrieren. Im ersten Fall ist das infizierte Programm das Wirtsprogramm und der Virus wird aktiv, sobald das infizierte Programm ausgeführt wird. Wenn ein Virus eine nicht ausführbare Datei befällt, beispielsweise ein Word-Dokument, wird der Virus aktiv, sobald das Dokument von dem Wirtsprogramm verarbeitet wird. Bei dem Beispiel des Word-Dokuments ist dies etwa das Textverarbeitungsprogramm *Microsoft Word*.

Zudem sind Viren selbstreproduzierend, d. h. dass Viren andere Dateien auf dem lokalen System infizieren, falls das Wirtsprogramm einen Virus zur Ausführung bringt. Ein Virus verbreitet sich demnach auf dem lokalen System. Dateien auf anderen Systemen können nur infiziert werden, falls eine bereits mit dem Virus infizierte Datei manuell auf ein weiteres System kopiert und dort zur Ausführung gebracht wird. Im Falle eines infizierten Dokuments, muss dieses auf dem neuen System natürlich verarbeitet werden – beispielsweise muss ein Word Dokument in einer anfälligen Version von *Microsoft Word* oder in einer sonstigen, kompatiblen Anwendung geöffnet werden.

Es existieren sehr viele unterschiedliche Arten von Viren, wie zum Beispiel

- *Bootsektorviren*, die ihren Code in den Bootsektor einer Festplattenpartition kopieren, damit dieser noch vor dem Betriebssystem zur Ausführung kommt,
- *Makroviren*, deren Code sich in makrofähigen Dateien, wie beispielsweise Microsoft Office Dateien, befindet und immer dann aktiv wird, wenn solch ein infiziertes Dokument von einem entsprechenden Wirtsprogramm verarbeitet wird,
- oder auch *Skriptviren*, die etwa PHP- oder VBS-Skripte befallen und dabei eine Kopie von sich selbst in noch nicht infizierte Skripte einfügen.

Die am weitesten verbreitete Virusart sind jedoch *Dateiviren*, die nochmals in zwei unterschiedliche Klassen eingeteilt werden. Zum einen existieren *parasitäre Dateiviren*, die sich an eine Programmdatei anhängen. Dadurch behält das Wirtsprogramm seine Funktionalität, jedoch verändert sich bei dieser Infektionsmethode die Dateigröße, was zu einer leichteren Erkennung der Infektion führt. Zum anderen gibt es *überschreibende Dateiviren*, die den Programmcode des Wirtsprogramms mit dem eigenen Schadcode überschreiben. Daraus resultiert eine schwerere Erkennbarkeit, aber auch, dass das Wirtsprogramm seine Funktionalität verliert [Szo05].

2.1.2. Würmer

Würmer sind genau wie Viren ebenfalls selbstreproduzierend, benötigen im Gegensatz zu Viren aber kein Wirtsprogramm, sondern sind jeweils ein autonom lauffähiges Computerprogramm. Die Klassifizierung eines Schadprogramms als Wurm bezieht sich auf dessen Verbreitungsroutine. Mit der immer dichteren Vernetzung von Computern durch das Internet in den 90er Jahren des 20. Jahrhunderts ging auch ein Wandel bezüglich der Schadprogramme einher. Während Viren noch auf die Weitergabe der infizierten Dateien durch einen Benutzer angewiesen waren, konnten sich Würmer über Computernetze verbreiten.

Zu diesem Zweck suchen klassische Würmer in Computernetzen nach Rechnern, die verwundbare Dienste anbieten. Ein Wurm nutzt dann eine Schwachstelle dieses Dienstes aus, um eine Kopie von sich selbst auf den gefundenen Rechner zu laden und diese dort zur Ausführung zu bringen. Durch diese Technik, die vollständig autonom abläuft und bei der kein Einschreiten eines Benutzers nötig ist, verbreitet sich diese Art von Würmern sehr schnell und unkontrolliert. Bekannte Computerwürmer sind der *Morris-Wurm*, *Code Red* oder auch *Blaster*, die im Abschnitt 3.3 auf Seite 54 genauer beschrieben werden. Eine detailliertere Beschreibung der typischen Verbreitungsroutine eines Wurms befindet sich im Anfang des Abschnitts 3.4.2.1 auf Seite 59 dieser Arbeit.

Neben diesen klassischen Würmern, bei deren Verbreitung keine menschliche Interaktion erforderlich ist, existieren auch Computerwürmer, die sich nur mit Hilfe einer Benutzerinteraktion verbreiten können. Klassische Beispiele sind E-Mail- oder auch Instant-Messaging-Würmer. Diese nutzen elektronische Kommunikationsmedien, wie etwa E-Mail oder Textnachrichtensysteme, mit denen elektronische Nachrichten verschickt werden können. Diese Medien werden für die Verbreitung von Würmern insofern missbraucht, als in den Nachrichten eine Instanz des Wurms oder zumindest ein Verweis auf den Wurm verschickt wird. Die Nachrichtenempfänger müssen den Wurm herunterladen und ausführen, damit sich diese Sorte von Computerwürmern verbreiten kann. Dadurch, dass ein Benutzer aktiv in den Verbreitungsprozess eingreifen muss, damit neue Systeme infiziert werden können, verbreiten sich solche Würmer langsamer als die zuvor beschriebenen klassischen Würmer. Alle Arten haben jedoch gemeinsam, dass sie keine Wirtsprogramme benötigen und sich über Computernetze verbreiten.

2.1.3. Bots

Als *Bot* werden Schadprogramme bezeichnet, die einem Angreifer erlauben, einem infizierten Rechner Befehle zukommen zu lassen und diesen auf diese Weise fernzusteuern. Selbstständig führen Bots keine schadhaften Aktivitäten aus, sondern erwarten lediglich Befehle, die sie über ein Computernetz von dem Angreifer erteilt bekommen. Diesem Umstand verdanken Bots auch ihren Namen: Der Begriff *Bot* leitet sich aus dem englischen Begriff *robot* ab und verdeutlicht, dass ein mit einem Bot infizierter Rechner wie ein Roboter die Befehle seines Programmierers/Programms ausführt. Eine besondere Gefahr geht von Bots aus, wenn sehr viele Rechner auf solch eine Art und Weise unter der Kontrolle eines einzelnen Angreifers stehen. Solch ein Verbund

aus kompromittierten Rechnern wird als *Botnetz* bezeichnet und ist durch die enorme Rechenkraft und Bandbreite, die einem Angreifer durch die infizierten Maschinen zur Verfügung steht, eine der gefährlichsten Bedrohungen in der heutigen IT-Sicherheit.

Im Laufe der Zeit haben sich einige Veränderungen bezüglich der Struktur der Botnetze erkennen lassen: Botnetze der sogenannten ersten Generation erhielten ihre Befehle von einem zentralen Rechner. Dieser wird *Command & Control Server* (kurz: C&C-Server) genannt und steht unter der Kontrolle des Angreifers, der das Botnetz betreibt. Vorteilhaft für den Angreifer ist, dass diese Botnetze sehr einfach aufzubauen sind. Als Nachteil hat sich jedoch der zentrale C&C-Server erwiesen. Wenn dieser Server der Kontrolle des Angreifers entzogen werden konnte, hatte er keine Möglichkeit mehr, den Bots Befehle zukommen zu lassen und demnach war das Botnetz wertlos für ihn [CJM05, FHW05, RZMT06]. Botnetze der zweiten Generation umgehen dieses Problem, indem sie auf einer dezentralisierten Netzstruktur basieren, wie etwa einem *Peer-to-Peer-Netz*. Bekannte Beispiele für diese Sorte Botnetz sind der *Sturm-Wurm* und *Waledac*, die unter anderem in Abschnitt 3.2 auf Seite 50 genauer beschrieben werden. Befehle erhalten Bots der zweiten Generation, indem diese von dem Angreifer in einen beliebigen Knoten des Netzes eingeschleust werden und dann durch das Netz propagieren. Es existiert also kein zentraler Rechner mehr, durch dessen Abschaltung das Botnetz untauglich gemacht werden könnte [GSN+07, KLE+08].

Die häufigsten Aufgaben eines Botnetzes bestehen aus dem Versand von Spam-Mails, etwa zur Infektion neuer Rechner und somit zur Weiterverbreitung des Bots oder etwa zum Bewerben von unterschiedlichsten Produkten. Oft werden Botnetze aber auch für *Distributed Denial of Service* Attacken (siehe Seite 60) genutzt, um die Verfügbarkeit eines Internetdienstes zu beeinträchtigen.

2.1.4. Trojanische Pferde

Trojanische Pferde zeichnen sich dadurch aus, dass diese Schadprogramme eine verborgene, schadhafte Funktionalität besitzen. Ein Opfer wird durch eine attraktiv klingende Beschreibung oder einen unauffälligen Dateinamen eines Computerprogramms dazu gebracht, dieses Programm und somit auch dessen verborgene Schadfunktion auszuführen. Dabei ist es bei der Klassifizierung eines Schadprogramms als Trojanisches Pferd unerheblich, ob dieses Programm die angebliche, beschriebene Funktionalität besitzt oder nicht. Es ist nur entscheidend, dass das Opfer bei der Ausführung des Programms keine Kenntnis von der schadhaften Funktionalität besitzt. Jedoch weckt ein Trojanisches Pferd, das nicht die versprochene Funktionalität besitzt, viel schneller Zweifel bei den Opfern als ein Trojanisches Pferd, das neben der verborgenen und schadhaften Funktionalität auch zusätzlich die beschriebene Funktionalität aufweist.

Wie sich ein Schadprogramm verbreitet, ist für die Einstufung als Trojanisches Pferd unerheblich. Sie können zum Beispiel über Tauschbörsen, Wechseldatenträger oder per Downloads verbreitet werden. Die Verbreitungsgeschwindigkeit Trojanischer Pferde hängt im Wesentlichen von der Nützlichkeit und der Attraktivität der angeblichen Programmfunktionalität ab.

2.1.5. Spyware

Das Ziel von *Spyware* ist es, unbemerkt Informationen über den oder die Benutzer des kompromittierten Rechners zu sammeln und diese an Dritte zu verschicken. Jedes Schadprogramm, das solch eine Funktionalität aufweist, bezeichnet man als Spyware. Die Namensgebung beruht auf den englischen Wörtern *to spy* und *software* und verdeutlicht den ausspionierenden Charakter dieser Programme.

Häufig wird Spyware von Unternehmen verwendet, um ihre Produkte zu vermarkten. Zu diesem Zweck zeichnet Spyware das Verhalten bei der Internetnutzung auf – also Informationen darüber, wann welcher Systembenutzer des kompromittierten Rechners, welche Internetseite besucht hat. Diese Informationen werden anschließend dazu genutzt, den entsprechenden Benutzern auf sie zugeschnittene Werbung und Pop-Ups zu präsentieren. Die Auswahl der so beworbenen Produkte basiert auf dem aufgezeichneten Internetverhalten und ist den möglichen Interessen des Benutzers angepasst.

Auch die auf Seite 12 beschriebenen Keylogger sind eine Form von Spyware. Diese zeichnen unbemerkt die Tastatureingaben der Benutzer eines befallenen Computers auf. In regelmäßigen Abständen übermitteln die Keylogger die aufgezeichneten Daten direkt oder indirekt an den Angreifer, damit sich dieser zum Beispiel finanziell an den gestohlenen Daten bereichern kann.

2.1.6. Rootkits

Unter dem Begriff *Rootkit* versteht man streng genommen eine Sammlung von Werkzeugen (engl.: *kit*), die dazu dienen, Administratorrechte auf einem System zu erhalten. Unter Unix-ähnlichen Betriebssystemen ist der Benutzer *Root* der Benutzer mit den höchsten Privilegien – vergleichbar mit dem Benutzer *Administrator* oder dem *Systemkonto* eines Windows-Systems. Ein Angreifer, der Zugang zu einem System erlangt hat, kann die einzelnen Software-Werkzeuge eines Rootkits dazu einsetzen, um Zugriff auf das System mit den höchsten Privilegien zu erhalten.

Immer häufiger werden jedoch auch alle Schadprogramme als Rootkit bezeichnet, die sich vor dem Benutzer, dem Betriebssystem und/oder Antivirenprogrammen verbergen. Zu diesem Zweck stellen Rootkits Werkzeuge zur Verfügung, die es erlauben, beliebige Objekte auf dem Computer zu verstecken, wie zum Beispiel Netzwerkverbindungen, Dateien oder auch ganze Prozesse. Aus diesem Grund kommen Rootkits oft in Verbindung mit anderen Arten von Schadprogrammen zum Einsatz, um deren Existenz zu verschleiern.

Allgemein kann man Rootkits anhand ihres Ansatzpunktes im System unterscheiden. Auf der einen Seite gibt es *Kernel-Level Rootkits*, die tief im System auf Ebene des Betriebssystems operieren. Diese sind schwer erkennbar, jedoch auch schwierig zu erstellen, da kleinste Fehler auf der Betriebssystemebene oftmals Abstürze des gesamten Systems nach sich ziehen. Auf der anderen Seite stehen die *User-Level Rootkits*, die zwar einfacher zu implementieren sind, jedoch auch besser detektierbar sind. User-Level Rootkits arbeiten häufig so, dass sie eigenen Code in die Abarbeitung von Systembefehlen einschleusen und so deren Ergebnis verändern können. Beispielsweise können aus der Liste der laufenden Prozesse bestimmte Einträge herausgefiltert wer-

den, um Prozesse zu verbergen. Mittlerweile existieren auch immer häufiger Mischformen, die die Vorteile beider Rootkit-Arten vereinen und somit stabil und relativ leicht zu implementieren sind.

Zusammenfassend enthält Tabelle 2.1 einen Überblick über die gerade beschriebenen Schadprogrammarten. In der Tabelle sind zudem die typischen Merkmale angegeben, die für eine Klassifizierung entscheiden sind. Wie bereits erwähnt, ist dieser Überblick keineswegs vollständig und bezieht sich auch nur auf Schadprogramme, die einen Personal Computer (PC) befallen. Immer häufiger treten auch Schadprogramme für mobile Endgeräte auf, wie zum Beispiel Smartphones. Diese stehen jedoch nicht im Fokus dieser Arbeit und werden deshalb hier auch nicht näher beschrieben.

Tabelle 2.1.: Verschiedene Arten von Schadprogrammen
Die Tabelle enthält einen Überblick über Schadprogrammarten und deren charakteristischen Merkmale. Diese Liste ist nicht vollständig und beinhaltet nur die verbreitetsten Arten von Schadprogrammen.

Schadprogrammart	Charakteristische Merkmale
Virus	Selbstreproduzierend (befällt Dateien des lokalen Systems)
	Benötigt Wirtsprogramm
Wurm	Selbstreproduzierend (infiziert weitere Systeme des Netzes)
	Benötigt kein Wirtsprogramm
Bot	Empfängt Befehle und führt diese aus
Trojanisches Pferd	Täuscht das Opfer durch attraktive Funktionalität
	Die eigentliche Schadfunktion bleibt verborgen
Spyware	Spioniert das Verhalten des Opfers aus
Rootkit	Ermöglicht einem Angreifer Rootzugriff auf das infizierte System
	Versteckt Objekte des Schadprogramms vor dem System/Benutzer

Abschließend sei nochmals auf die Tatsache hingewiesen, dass sich Schadprogramme in der Regel nicht genau einer Kategorie der hier vorgestellten Schadprogrammarten zuordnen lassen. So verwenden zum Beispiel viele Computerwürmer Techniken, um ihre Existenz auf den infizierten Rechnern zu verbergen und spähen persönliche Daten der Opfer aus. Somit sind sie neben der Kategorie *Wurm* auch den Kategorien *Rootkit* und *Spyware* zuzuordnen.

2.2. Verbreitungswege von Malware

Nachdem wir nun typische Schadprogramme kennengelernt haben, schauen wir uns als nächstes deren Verbreitungsmöglichkeiten genauer an. Schadprogramme können sich über zahlreiche Wege verbreiten und immer wieder kommen neue Mechanismen hinzu, mit deren Hilfe sie weitere Rechner infizieren können. E-Mails, Tauschbörsen, Instant Messenger oder auch Datenträger werden für die Weiterverbreitung genutzt.

Generell können alle Wege, über die beliebige Daten auf einen Rechner gelangen, auch von Schadprogrammen genutzt werden, um einen Rechner zu infizieren. Für eine Infektion kann es sogar schon ausreichend sein, lediglich den Rechner mit dem Internet oder einem lokalen Netz zu verbinden. Auch der reine Besuch einer speziell präparierten Internetseite kann einen sogenannten *Drive-By-Download* auslösen, bei dem unbemerkt vom Benutzer ein Schadprogramm auf den Rechner geladen wird. Die folgenden Abschnitte enthalten Beschreibungen der am häufigsten von Schadprogrammen verwendeten Verbreitungsroutinen.

2.2.1. Netzdienste mit Sicherheitslücken

Wie bereits im vorherigen Abschnitt erklärt, verbreiten sich Computerwürmer autonom über angeschlossene Computernetze. Zu diesem Zweck generieren sie nach der Kompromittierung eines Computers mehr oder weniger zufällig IP-Adressen. Die Computer mit den generierten IP-Adressen sind die nächsten Angriffsziele der Computerwürmer. Werden von diesen Computern Dienste angeboten, die eine Sicherheitslücke aufweisen, so nutzen Computerwürmer diese, wenn möglich, aus, um eine Kopie von sich selbst auf den noch nicht infizierten Computer zu laden und auszuführen. Durch die Ausführung sind auch diese Computer infiziert und der Prozess beginnt auf dem gerade infizierten Computer wieder von vorn, d. h. es werden neue IP-Adressen generiert, es wird geprüft, ob sich auf diesen Rechnern Dienste mit Sicherheitslücken befinden, und wenn ja, dann werden diese Sicherheitslücken ausgenutzt, um einen weiteren Rechner zu infizieren.

Sobald ein Computer an ein Netz angeschlossen ist, in welchem er für die anderen Rechner im Netz erreichbar bzw. zugänglich ist, besteht also theoretisch die Gefahr, dass er durch Schadprogramme infiziert wird. Dabei nimmt die Anzahl der Rechner des Netzes wesentlichen Einfluss auf die Wahrscheinlichkeit einer Infektion: Je größer das Netz ist, desto wahrscheinlicher ist es, dass eine Infektion stattfindet. Beispielsweise ist die Gefahr einer Kompromittierung in einem sehr kleinen Netz, das aus nur fünf Computern besteht, wesentlich geringer als beim Internet, das viele Millionen von Computern verbindet. Dies liegt im Wesentlichen daran, dass sich die Anzahl schlecht gewarteter Rechner proportional zur Gesamtgröße eines Netzes verhält.

An der Universität Mannheim wurde im Jahr 2007 von Laura Itzel eine Diplomarbeit erstellt, bei der unter anderem die Zeitspanne gemessen wurde, die zwischen dem Verbinden eines Rechners mit dem Internet und einer Kompromittierung des Rechners durch ein Schadprogramm vergeht. Als Datengrundlage der Experimente dienten die Daten von 10 Honeypots, die in einem Zeitraum von 10 Wochen über verschiedene Internet Service Provider (ISP) mit dem Internet verbunden waren. Das Ergebnis dieses Experiments war, dass bei einem Computer, auf dem das Betriebssystem *Windows XP Service Pack 2* installiert ist und das dynamisch eine IP-Adresse vom ISP bezieht, durchschnittlich nach 2,5 Stunden das Herunterladen eines Schadprogramms festgestellt werden konnte. Einer der 10 Honeypots wurde im Schnitt sogar bereits nach 30,6 Minuten mit Schadprogrammen infiziert [Itz07]. Vor allem beim Aufsetzen eines neuen Systems sind diese Werte von besonderem Interesse. Installiert man ein Betriebssystem, das anschließend noch durch Updates aus dem Internet auf den aktu-

ellen Sicherheitsstand gebracht werden muss, so ist die Gefahr größer, dass dabei der Computer kompromittiert wird, je länger dieser Update-Vorgang andauert.

Die Sicherheitslücken in einem Netzdienst, oder generell in einer Anwendung, basieren meistens auf einer nachlässigen Programmierweise. Aber auch die falsche Planung oder Konzeption einer Anwendung kann dafür verantwortlich sein, dass die Anwendung während ihrer Ausführung Programmzustände annimmt, die so von den Entwicklern nicht beabsichtigt waren und im schlimmsten Fall ein unkontrolliertes Verhalten der Anwendung nach sich ziehen. Sicherheitslücken sind zum Beispiel *Race Conditions* [NM92, FF00], *ungenügende Überprüfung von Benutzereingaben* [BM10] oder auch *Pufferüberläufe* [CPM+98]. Dabei ist die letztgenannte Sicherheitslücke wohl die bekannteste und für eine Vielzahl von Sicherheitsvorfällen in den letzten Jahren verantwortlich. Ein Pufferüberlauf (engl. Buffer Overflow) passiert immer dann, wenn Daten in einen Puffer, d. h. eine Variable fester Länge, geschrieben werden, ohne dass eine Überprüfung stattfindet, ob diese Daten tatsächlich in diesen Puffer passen. Sind die zu schreibenden Daten länger als der Puffer, so werden die Speicherbereiche hinter diesem Puffer mit Daten überschrieben. Ein Angreifer kann diese Technik beispielsweise ausnutzen, um die Rücksprungadresse eines Funktionsaufrufs auf dem Stack beliebig zu manipulieren. Auf diese Weise ist es ihm möglich, den Programmfluss einer Anwendung auf eigenen Code umzulenken, so dass dieser im Kontext der verwundbaren Anwendung zur Ausführung kommt [One96].

Die Verbreitung von Schadprogrammen über verwundbare Netzdienste läuft vollständig autonom ab – also ohne Eingreifen eines Computerbenutzers. Darüber hinaus wird der Infektionsprozess von den Benutzern in der Regel nicht bemerkt. Sowohl eine gut konfigurierte Firewall als auch ein Intrusion Prevention System können vor einer Infektion über diesen Verbreitungsweg schützen.

2.2.2. World Wide Web

Eine der wichtigsten Anwendungen des Internets stellt das *World Wide Web* (WWW) dar. Dieses wurde im Jahr 1991 zur öffentlichen Nutzung freigegeben und besteht aus elektronischen Dokumenten, die als Internetseiten bezeichneten werden. Diese können mit Hilfe eines Internetbrowsers betrachtet werden und sind über Querverweise miteinander verbunden. Die Verweise erlauben somit einem Benutzer eine komfortable Navigation durch das WWW. Neben vielen Privatpersonen besitzen heutzutage auch nahezu jedes Unternehmen, jede Organisation/Institution oder auch viele Vereine eine Internetseite, über die Informationen über den entsprechenden Inhaber der Seite abgerufen werden können. Dies liegt vor allem in den geringen Kosten einer Internetpräsenz verglichen zu dem riesigen Empfängerkreis, den man über das WWW erreichen kann. Im Mai 2013 setzte sich das WWW schätzungsweise aus mehr als 672,8 Millionen Internetseiten zusammen. Alleine im April 2013 sind ca. 23,8 neue Seiten hinzugekommen. Dies entspricht einer Wachstumsrate von über 3,6 % in einem Monat und verdeutlicht das rasante Wachstum des World Wide Web [NL13].

Auch Autoren von Schadprogrammen machen sich die große Popularität und das rasante Wachstum des WWW zunutze und versuchen ihre Schadprogramme über Internetseiten zu verbreiten. Die einfachste Möglichkeit besteht darin, diese auf einer Internetseite zum Herunterladen anzubieten. Da in der Regel kein Benutzer seinen Com-

puter freiwillig mit Schadprogrammen infiziert, täuschen Malware-Autoren die Seitenbesucher mit einer falschen Beschreibung der angebotenen Datei und verschweigen deren schadhafte Funktionalität. Die Geschwindigkeit, mit der sich Schadprogramme auf solch eine Art und Weise verbreitet, hängt im Wesentlichen davon ab, wie gut oder glaubwürdig die täuschende Beschreibung formuliert ist. Definitionsgemäß handelt es sich bei solchen Schadprogrammen also um Trojanische Pferde, da nur ein Teil der Funktionalität der zum Herunterladen angebotenen Datei dem Seitenbesucher bekannt ist.

Aber bereits der reine Besuch einer Internetseite kann zur Infektion mit einem Schadprogramm ausreichen. Bei diesem als *Drive-By-Download* bezeichneten Infektionsmechanismus werden Schadprogramme über bösartig präparierte Internetseiten verbreitet [PMM+07]. Außer dem Besuch einer solchen Seite erfordert ein Drive-By-Download keine Aktivität des Opfers und wird von diesem auch nicht wahrgenommen. Voraussetzung für einen erfolgreichen Drive-By-Download ist, dass das Opfer beim Besuch der Seite einen für solch einen Angriff verwundbaren Browser verwendet. Ein Drive-By-Download nutzt ebenso wie der zuvor beschriebene Verbreitungsweg eine Sicherheitslücke aus. Nur diesmal findet das Ausnutzen der Sicherheitslücke nicht auf der Serverseite in einem Netzdienst statt, sondern auf der Clientseite – nämlich im Browser eines Benutzers. Durch das Ausnutzen solch einer Sicherheitslücke wird fremder Code, der sich in der bösartig präparierten Internetseite befindet, innerhalb des Prozesskontextes des Browsers ausgeführt. Der Vorteil dieser Vorgehensweise liegt darin, dass ein Angreifer nicht aktiv einen Computer kompromittiert, sondern die Kompromittierung von den Opfern initiiert wird, die die bösartig präparierten Internetseiten besuchen. Auf diese Weise muss ein Angreifer etwa keine Firewalls umgehen und kann auch Computer kompromittieren, die sich hinter einem Router befinden, der eine *Network Address Translation* (NAT) durchführt.

Eine momentan sehr verbreitete Methode zur Realisation eines Drive-By-Downloads ist ein *Heap-Spraying*-Angriff [RLZ08]. Dieser besteht im Wesentlichen aus zwei Schritten:

1. Bösartiger Code, der einem Angreifer die Kontrolle über den zu kompromittierenden Rechner ermöglicht, muss auf den Computer geschleust werden. Dieser Code wird als *Shellcode* bezeichnet.
2. Der Computer muss vom Angreifer dazu gebracht werden, dass er den Shellcode ausführt.

Eine Möglichkeit, Shellcode über eine Internetseite auf den Computer eines Opfers zu schleusen, liegt in der Verwendung von *JavaScript*. Dies ist eine objektorientierte Skriptsprache zur Gestaltung von dynamischen Internetseiten und wird häufig dazu benutzt, um vom Benutzer eingegebene Formulardaten zu verarbeiten oder auf Seite des Clients auf die Objekte des *Document Object Model* (DOM) zuzugreifen. Der Kern von JavaScript wurde als *ECMAScript* in der Spezifikation *ECMA-262* zum Industriestandard erklärt [Int99].

Der erste Schritt eines Heap-Spraying-Angriffs wird häufig dadurch realisiert, dass ein Angreifer JavaScript-Code in eine Internetseite einbettet. Dieser wird beim Besuch der Internetseite vom Browser des Opfers ausgeführt und sorgt dafür, dass der Shellcode im Heap des Browsers ablegt wird. Der Heap ist der Speicher, der für dynamisch

allozierte Objekte bereitsteht. Code-Block 2.1 zeigt ein typisches Beispiel eines JavaScripts, das genau diese Aufgabe erfüllt: Der Shellcode ist in diesem Beispiel aus Darstellungsgründen abgekürzt und wird in einer gleichnamigen Variablen abgelegt (Zeile 2). Da ein Angreifer nicht die genaue Speicheradresse des Shellcodes innerhalb des Heap voraussagen kann, wird dem Shellcode eine sogenannte *NOP-Slide* vorangestellt. Dies ist eine Folge von NOP-Instruktionen, während deren Ausführung der Prozessor keine Aktionen ausführt (NOP ist eine Kurzform von *no operation*). Zeigt der Instruktionszeiger der CPU auf eine Stelle der NOP-Slide, so werden nach und nach die NOP-Instruktionen abgearbeitet, bis der Shellcode zur Ausführung kommt. Die NOP-Slide fungiert im Prinzip als Landebahn für den Shellcode. Je länger diese ist, desto wahrscheinlicher ist es, dass der Instruktionszeiger so manipuliert werden kann, dass dieser auf die NOP-Slide zeigt und somit letztendlich auch der Shellcode ausgeführt wird. In den Zeilen 3 bis 8 wird die NOP-Slide iterativ und mit einem exponentiellen Wachstum aufgebaut: Sie wird solange mit sich selbst konkateniert (Zeile 7), bis eine bestimmte Länge erreicht ist (Zeile 6). Abschließend wird ein Feld generiert, das 1 000 Objekte beinhaltet, die eine Kombination aus der NOP-Slide und dem Shellcode sind (Zeilen 10 bis 13). Durch dieses mehrmalige Generieren dieser Objekte, wird die Wahrscheinlichkeit, dass ein Angreifer den Instruktionszeiger so manipulieren kann, dass er auf eines dieser Objekte zeigt, weiter deutlich erhöht. Der Name dieses Angriffs leitet sich aus genau diesem Versprühen/Verstreuen des Shellcodes über den Heap der Zielanwendung ab.

```
1  <SCRIPT language="text/javascript">
2    shellcode = unescape("%u4343%u4343%...");
3    oneblock = unescape("%u0D0D%u0D0D");
4
5    var fullblock = oneblock;
6    while (fullblock.length<0x40000) {
7      fullblock += fullblock;
8    }
9
10   sprayContainer = new Array();
11   for (i=0; i<1000; i++) {
12     sprayContainer[i] = fullblock + shellcode;
13   }
14 </SCRIPT>
```

Code-Block 2.1: JavaScript eines Heap-Spraying-Angriffs [RLZ08]

Im Gegensatz zum ersten Schritt eines Heap-Spraying-Angriffs, erfordert der zweite Schritt von einem Angreifer in der Regel die Durchführung einer illegalen Handlung. Er muss den Computer eines Opfers dazu bringen, dass der im ersten Schritt eingeschleuste Shellcode auch ausgeführt wird. Dazu wird eine Sicherheitslücke des Browsers vom Angreifer ausgenutzt, beispielsweise durch einen Pufferüberlauf des Stack oder Heap, um den Instruktionszeiger der CPU auf den eingeschleusten Code im Heap zeigen zu lassen. Auf diese Weise wird nach Abarbeitung der NOP-Slide der Shellcode mit den Privilegien des Browsers zur Ausführung gebracht. Der Shellcode lädt üblicherweise Schadprogramme auf den kompromittierten Computer nach und ermöglicht so dem Angreifer die Kontrolle über diesen.

Um die Geschwindigkeit zu erhöhen, mit der sich ein Schadprogramm über eine Internetseite verbreitet, ist ein Angreifer daran interessiert, dass möglichst viele Benutzer die entsprechende Internetseite besuchen. Dies kann über zwei verschiedene Ansätze realisiert werden:

- Zum einen kann der Angreifer dafür sorgen, dass der Bekanntheitsgrad dieser Seite höher wird. Beispielsweise kann er in vielen Foren oder Blogs Verweise auf die Seite hinterlegen oder auch massenhaft Verweise per E-Mail verschicken. Weitere Möglichkeiten, die Besucherzahl der Seite zu steigern, sind etwa das zielgerichtete Gestalten der Internetseite, so dass diese in den Ergebnislisten großer Suchmaschinen weit oben geführt wird [LM06], oder das Registrieren von Domains, die sehr populären Domains ähneln und sich etwa nur durch einen Buchstabendreher von diesen unterscheiden. Durch Tippfehler landen Besucher auf der Internetseite, über die das Schadprogramm verbreitet wird, obwohl sie diese gar nicht besuchen wollten. Ein Beispiel einer Domain mit Buchstabendreher ist *mircosoft.com*, in der im Vergleich zu der populären Microsoftdomain *microsoft.com* nur die Buchstaben *c* und *r* vertauscht sind. Diese Technik wird als *Typosquatting* bezeichnet [WBW+06].

- Zum anderen kann ein Angreifer bereits existierende und populäre Internetseiten nutzen, um über diese seine Schadprogramme zu verbreiten. Der Großteil des Inhalts einer Internetseite wird normalerweise vom Besitzer der Seite erstellt, jedoch kommt es auch immer häufiger vor, dass fremder Inhalt in die eigene Internetseite eingebunden wird. Über das `iframe`-Element von HTML oder auch über JavaScript lassen sich teilweise komplette Internetseiten, auch ohne dass ein Seitenbesucher diese sieht, in die eigene Seite integrieren. Weiterleitungen lassen sich auf diese Weise ebenfalls problemlos und transparent für einen Besucher realisieren. Ein Angreifer, der unautorisierten Zugriff auf eine eigentlich harmlose Internetseite hat, kann auf diese Weise unbemerkt eigene Inhalte in bestehende Seiten einbetten. Auch bei Foren- und Blogeinträgen wird fremder Inhalt in eine Internetseite eingebettet. Dies ist vor allem dann problematisch, wenn diese Einträge sämtliche HTML-Elemente erlauben, wie etwa das `script`-Element [PMM+07].

  Aber auch Werbebanner auf bestehenden Internetseiten können für die Verbreitung von Schadprogrammen benutzt werden. Sogenannte *Werbenetzwerke* (engl. *advertising networks*) haben sich auf die Vermittlung von Werbebannern spezialisiert. Ein Betreiber einer Internetseite, der einen Werbeplatz seiner Seite vermarkten möchte, bekommt Werbebanner von einem Werbenetzwerk. Dieses wiederum bekommt die Banner von den Kunden, die selbst auf anderen Seiten werben möchten. Eingebunden werden die Banner häufig über dynamische Technologien wie zum Beispiel JavaScript oder Ajax. Ein Angreifer kann dies nutzen, um schadhafte Werbebanner in die Werbenetzwerke einzuschleusen. Ein Kunde, der diese Banner ungeprüft auf seiner Internetseite anzeigt, setzt somit seine Seitenbesucher der Gefahr einer möglichen Infektion mit Schadprogrammen aus [KWK+07, Eva08, Bac09].

2.2.3. E-Mail

Das Medium E-Mail [Res08] bietet zwei Möglichkeiten, um darüber Schadprogramme zu verbreiten: Zum einen können die Nachrichten einen Verweis auf eine Internetseite enthalten und zum anderen können Schadprogramme als Dateianhänge verschickt werden. In beiden Fällen ist für eine Infektion des Computers eines E-Mail-Empfängers eine Benutzerinteraktion notwendig. Im ersten Fall muss der Empfänger durch einen Mausklick dem Verweis folgen, das Schadprogramm manuell von der Internetseite herunterladen und abschließend ausführen. Ebenfalls infiziert sich sein Rechner, wenn die Internetseite hinter dem Verweis bösartig präpariert ist und er einen entsprechend anfälligen Browser verwendet. Folgt ein Empfänger einem Verweis und besucht die Internetseite, die sich hinter diesem Verweis befindet, so bieten sich einem Angreifer also alle Möglichkeiten zur Infektion des Empfängerrechners, die im Abschnitt über die Verbreitung durch das World Wide Web beschrieben sind. Ist das Schadprogramm hingegen in dem Dateianhang der E-Mail enthalten, ist es für eine Infektion erforderlich, dass ein Empfänger den Anhang lokal speichert und ausführt – was wiederum auch eine Benutzerinteraktion darstellt.

Ebenso wie die Verbreitungsrate über das World Wide Web hängt auch die Geschwindigkeit der Verbreitung eines Schadprogramms über das Medium E-Mail davon ab, wie gut ein Empfänger einer E-Mail durch deren Inhalt getäuscht werden und zur Ausführung des Schadprogramms verleitet werden kann. Der reine Empfang einer E-Mail, die ein Schadprogramm als Dateianhang enthält oder einen Verweis auf dieses beinhaltet, ist für eine Rechnerinfektion in Normalfall nicht ausreichend. In beiden Fällen muss ein Angreifer den Empfänger erst dazu bewegen, etwa durch Social Engineering Tricks, eine Benutzerinteraktion durchzuführen.

Sollte ein E-Mail-Client jedoch eine Sicherheitslücke aufweisen, so können sich Schadprogramme auch durch deren Ausnutzen verbreiten. Im Gegensatz zu Netzdiensten, die passiv hinter einem bestimmten Port auf eine eingehende Verbindung warten, fragen E-Mail-Clients aktiv einen Mailserver nach neuen E-Mails ab. Das Ausnutzen der Sicherheitslücke findet in diesem Fall durch speziell gestaltete E-Mails statt, die beim Anzeigen im E-Mail-Client eine möglicherweise vorhandene Sicherheitslücke ausnutzen. Bestimmte Versionen von *Microsoft Outlook* wiesen im Jahr 2010 etwa eine Sicherheitslücke auf, die es einem Angreifer erlaubten, beliebigen Code auf dem System eines Benutzers auszuführen, wenn der Client speziell gestaltete E-Mails nur anzeigte [CVE10].

Um die Wahrscheinlichkeit, dass ein Empfänger auch tatsächlich das Schadprogramm ausführt, weiter zu erhöhen, kann ein Angreifer zusätzlich die Header-Einträge der E-Mail fälschen. Eine E-Mail besteht neben dem Inhalt der E-Mail aus dem SMTP-Envelope, ähnlich dem Briefumschlag eines Briefes, der die für die Zustellung einer E-Mail relevanten Informationen beinhaltet, und den Header-Einträgen, die vergleichbar mit dem Briefkopf eines nicht-elektronischen Briefes sind. Die Informationen des Envelope werden in der Regel nur von den Mailservern zur Weiterverarbeitung der E-Mail ausgewertet und dem Empfänger der E-Mail nicht angezeigt. Dieser sieht hingegen die Einträge aus dem E-Mail-Header, die jedoch von einem Angreifer beliebig gesetzt werden können. Beispielsweise kann an Angreifer, wenn er einen Kontakt eines Opfers kennt, die E-Mail so verfassen, dass das Opfer denkt, der Autor der E-Mail

wäre dieser Kontakt. Da ein Empfänger einer bekannten Person mehr vertraut als einer fremden, ist die Wahrscheinlichkeit, dass er den Anhang ausführt oder einem möglichen Verweis folgt, höher [KK06b].

Einige E-Mail-Würmer machen sich diesen psychologischen Trick zunutze. Nach einer erfolgreichen Infektion versuchen sie, das Adressbuch des Opfers oder dessen Posteingang auszulesen, um dadurch an E-Mail-Adressen zu gelangen, die in Kontakt mit dem Opfer stehen. Nachdem die Würmer solch eine Adressliste aufgebaut haben, schicken sie E-Mails mit einer Instanz von sich selbst an die entsprechenden Adressen. Ein bekanntes Beispiel solch eines Wurms ist *ILOVEYOU*, der in Abschnitt 3.2.3 auf Seite 52 genauer beschrieben wird. Würmer, die den Posteingang auslesen, verschicken die E-Mails häufig als Antwort auf die gefundenen E-Mails – was wiederum ebenfalls zum Erhöhen der Infektionswahrscheinlichkeit beiträgt.

2.2.4. Instant Messanging

Instant Messanging (IM) ist ein sehr weit verbreiteter Dienst, der die Kommunikation über Textnachrichten im Internet in Echtzeit ermöglicht. Es folgt zumeist dem Client-Server-Modell und erlaubt den Nutzern bzw. den Kommunikationspartnern, sich gegenseitig über die IM-Clients, die als Instant Messenger bezeichnet werden, Nachrichten zu schicken. Neben diesem reinen Nachrichtenaustausch bieten fast alle aktuellen Instant Messenger auch die Möglichkeit, Dateien auszutauschen, gegen andere IM-Teilnehmer Spiele zu spielen oder auch sich per Video-Chat und *Voice over IP* (VoIP) zu unterhalten.

Der IM-Dienst wird von sehr vielen Menschen genutzt und die IM-Clients bieten ihren Nutzern immer mehr Funktionen an. Da jedoch die nötigen Sicherheitsfeatures fehlen, sind Instant Messenger oft das Ziel von Angreifern, um darüber Schadprogramme zu übertragen und somit den Rechner eines Kommunikationspartners zu kompromittieren: Viele Instant Messenger führen heutzutage immer noch keine Überprüfung der ausgetauschten Dateien durch [Wan09]. Im Gegensatz zum E-Mail-Verkehr ist das Filtern und Analysieren des Datenstroms einer IM-Kommunikation an einem zentralen Gateway nicht ohne Weiteres möglich. Dies liegt vor allem daran, dass es sehr viele verschiedene Instant Messenger gibt, die größtenteils eigene Protokolle verwenden. Zudem werden sowohl die Clients als auch die Protokolle ständig modifiziert [MvO05].

Die Möglichkeiten, die zur Verbreitung von Schadprogrammen über Instant Messenger verwendet werden können, sind denen zur Verbreitung über E-Mails ähnlich. Zum einen können Textnachrichten Verweise auf schadhaft präparierte Internetseiten enthalten und zum anderen können die Schadprogramme direkt über die Dateiaustausch-Funktion der IM-Clients einem Kommunikationspartner übermittelt werden. In beiden Fällen ist eine Benutzerinteraktion erforderlich: Entweder wird dem Verweis per Mausklick gefolgt oder der Dateitransfer muss akzeptiert und die so übermittelte Datei geöffnet oder ausgeführt werden. Nach einer Infektion versuchen Würmer in der Regel die Kontakte in der Kontaktliste eines Opfers zu infizieren.

Beim Instant Messanging existieren ebenso Verbreitungswege, die keine Benutzerinteraktion benötigen [HC03]. Sobald ein Benutzer einen IM-Client benutzt, der eine Sicherheitslücke aufweist, können Schadprogramme diese ausnutzen und auch dar-

über die entsprechenden Rechner infizieren. Beispielsweise enthielt der *AOL Instant Messenger* 2002 einen Pufferüberlauf, der es einem Angreifer durch eine Spielanfrage mit einem überlangen Parameter erlaubte, beliebigen Code auf einem Rechner auszuführen [CVE02].

2.2.5. Soziale Netzwerke

Soziale Netzwerke erfreuen sich seit einigen Jahren einer wachsenden Beliebtheit. Teilnehmer können eigene Seiten erstellen, die Profile genannt werden und es den Benutzern ermöglichen, sich selbst im Internet darzustellen. Sie können Freundesnetzwerke aufbauen und mit anderen Teilnehmern des Netzwerkes Textnachrichten austauschen. Das zur Zeit erfolgreichste Soziale Netzwerk ist *Facebook* [FI11a]. Dieses wurde im Jahr 2004 entwickelt und der Unternehmenswert wird nach nur 6 Jahren auf ca. 50 Mrd. US-Dollar geschätzt. Damit ist Facebook eines der größten Unternehmen weltweit und besitzt laut eigenen Angaben seit Juli 2010 mehr als 500 Millionen aktive Mitglieder [FTD11].

Auch Autoren von Schadprogrammen haben dieses riesige Potential, das in sozialen Netzwerken steckt, für ihre Aktivitäten entdeckt. Dieses Potential basiert vor allem auf den hohen Mitgliederzahlen und der vielen Zeit, die Benutzer innerhalb solcher Netzwerke verbringen. Mittlerweile existieren Schadprogramme, die nach einer Infektion eines Computers diesen nach Zugangsdaten zu sozialen Netzwerken durchsuchen. Werden solche Zugangsdaten gefunden, melden sich diese Schadprogramme automatisiert in den entsprechenden Netzwerken an und verschicken Nachrichten an jeden Teilnehmer in der Kontaktliste des Opfers. Diese Nachrichten sind Teil des Verbreitungsprozesses des Schadprogramms und enthalten einen Verweis auf eine Internetseite.

Der wohl bekannteste Computerwurm, der sich auf diese Weise verbreitet, ist *Koobface*. Der Name ist ein Anagramm des Wortes *Facebook*, über das sich der Computerwurm verbreitet. Neben Facebook verbreitet sich Koobface zusätzlich auch über die sozialen Netzwerke *Myspace* [MI11], *hi5* [NI11], *Bebo* [BI11], *Friendster* [FI11b] und *Twitter* [TI11]. Die Verweise innerhalb der von Koobface verschickten Nachrichten zeigen meist auf Internetseiten, auf denen angeblich ein Video zu sehen ist. Möchte sich ein Seitenbesucher dieses Video anschauen, wird ihm eine Fehlermeldung präsentiert, die besagt, dass ihm zum Betrachten des Videos ein notwendiges Adobe Flash Plug-in fehlen würde. Gleichzeitig wird ihm dieses Plug-in zum Herunterladen angeboten. Auf diese Weise sollen die Seitenbesucher dazu gebracht werden, das angebliche Plug-in zu installieren. Tatsächlich handelt es sich bei dem Plug-in jedoch um eine Instanz von Koobface [TN10]. Die kompromittierten Rechner schließen sich nach der Infektion einem Botnetz an und warten auf die Befehle des Angreifers.

2.2.6. Filesharing

Als *Filesharing* bezeichnet man eine Methode des Dateiaustauschs zwischen zwei Benutzern über das Internet. Für diesen Zweck existieren verschiedene Protokolle und Anwendungen, die basierend auf dem Internet, ein Netz aufspannen, in dem jeder Teilnehmer bestimmte Dateien freigeben kann. Andere Teilnehmer können über die-

ses Netz Dateien anfragen und von den Computern der Teilnehmer, die diese Datei freigegeben haben, auf ihren eigenen Computer herunterladen. Filesharing-Netze sind auch unter dem Namen *Internet-Tauschbörsen* bekannt. Die ersten Netze dieser Art hatten eine zentralisierte Client-Server-Struktur. Später hingegen entwickelten sich immer mehr Filesharing-Netze, in denen jeder Teilnehmer sowohl als Client als auch als Server fungiert. Diese aus Sicht der Zuverlässigkeit und Verfügbarkeit weitaus robusteren Netze nennt man Peer-to-Peer-Netze und bilden mittlerweile den Großteil der Filesharing-Netze.

Wie bereits erwähnt, kann ein Schadprogramm jeden Weg nutzen, um einen Rechner zu infizieren, über den auch beliebige andere und harmlose Dateien auf diesen Rechner gelangen – also auch per Filesharing. Für einen Autor eines Schadprogramms ist die Verbreitung über solch ein Netz deshalb interessant, da er sein Schadprogramm unter jedem beliebigen Dateinamen anbieten kann und die Filesharing-Netze keine oder kaum inhärente Schutzmaßnahmen bieten. Viele Autoren von Schadprogrammen beziehen die Dateinamen, unter denen sie ihre Schadprogramme in einem Filesharing-Netz anbieten, auf aktuelle Ereignisse oder aktuell populäre Suchanfragen. Auf diese Weise erreichen sie eine möglichst gute Verbreitung der Schadprogramme. Nach einer Infektion eines Rechners befallen diese sogar teilweise die von diesem freigegebenen Dateien oder kopieren sich unter anderem Namen und leicht modifiziert in den freigegebenen Ordner der Filesharing-Anwendung. Eine Untersuchung ergab, dass im Februar und Mai 2006 mehr als 22 % beziehungsweise 15 % aller angebotenen Dateien eines beliebten Filesharing-Netzes mit Schadprogrammen infiziert waren [SJB06]. Eine andere Studie, in deren Fokus ein anderes beliebtes Filesharing-Netz stand, ergab sogar, dass 68 % der heruntergeladenen Dateien Schadprogramme oder Archive waren, die ein Schadprogramm enthalten [KAG06].

Auch die Clientanwendungen, die benutzt werden, um in Filesharing-Netzen Dateien auszutauschen, können Sicherheitslücken aufweisen. Diese können von Schadprogrammen ausgenutzt werden, um darüber Rechner zu infizieren.

2.2.7. Netzfreigaben

Die *Netzfreigabe* ist in einem lokalen Netz das Pendant zum Filesharing. Eine Netzfreigabe ermöglicht es, Dateien bestimmter Ordner oder ganzer Partitionen zum Lesen, Schreiben oder Ausführen freizugeben. Dies bedeutet, dass die entsprechenden Aktionen auch von anderen Rechnern aus über ein lokales Netz ausgeführt werden können. Häufig wird von dieser Methode des entfernten Zugriffs Gebrauch gemacht, wenn mehrere Personen auf die gleichen Dateien zugreifen müssen, zum Beispiel bei einem Teamprojekt eines Unternehmens.

Aber auch Schadprogramme können von dieser Möglichkeit des entfernten Zugriffs profitieren: Wird ein Rechner von einem Schadprogramm infiziert, so kann dies überprüfen, ob von dem kompromittierten Rechner Schreibzugriffe auf die Dateien einer Netzfreigabe möglich sind. Werden solche Dateien gefunden, kann das Schadprogramm diese Dateien ebenfalls infizieren. Führt anschließend ein Benutzer eines solchen entfernten Rechners eine dieser modifizierten Dateien aus, so ist auch dieser Rechner mit dem Schadprogramm infiziert.

2.2.8. Wechseldatenträger

Klassische Viren (siehe Seite 13) sind zur Infektion eines Rechners auf die manuelle Weitergabe eines Benutzers angewiesen. Vor der weltweiten Vernetzung durch das Internet bestand die häufigste Form des Dateiaustauschs in der Weitergabe von Dateien über *Wechseldatenträger* wie Disketten, CDs oder USB-Sticks. Ein Virus, der auf einem angeschlossenen Wechseldatenträger aktiv wird, oder von diesem auf das lokale System kopiert und dann dort aktiv wird, kann auf diese Weise Dateien des lokalen Systems infizieren.

Seit *Windows 95* bieten die Betriebssysteme aus dem Hause *Microsoft* eine Funktionalität, die sogar eine wurmhafte Verbreitung über Wechseldatenträger ermöglicht: Beim Anschließen eines Wechseldatenträgers wird dieser vom Betriebssystem nach der Datei *autorun.inf* durchsucht. Innerhalb dieser Datei kann ein Programm spezifiziert werden, das beim Anschließen des Datenträgers automatisch ausgeführt wird. Gedacht war diese Funktionalität zur leichteren Installation von Programmen durch technisch nicht-versierte Endanwender. Autoren von Schadprogrammen missbrauchen diese Funktionalität jedoch so, dass ihre Schadprogramme nicht mehr auf eine Benutzerinteraktion (das manuelle Starten des Schadprogramms auf dem Wechseldatenträger) angewiesen sind, sondern automatisch von Windows beim Anschließen des Datenträgers geladen werden. In den letzten Jahren konnte ein Anstieg an Schadprogrammen registriert werden, der diese *AutoRun* genannte Funktionalität ausnutzt. Das amerikanische Militär hat aus Angst vor einem Computerwurm, der sich auf diese Weise verbreitete, im Jahr 2008 sogar teilweise die Nutzung von Wechseldatenträgern und insbesondere USB-Sticks untersagt [VTM09].

Aber nicht immer müssen die Wechseldatenträger unter Benutzern weitergereicht werden, damit deren Rechner mit Schadprogrammen infiziert werden. Im Oktober 2010 startete der deutsche Künstler Aram Bartholl ein Projekt namens *Dead Drops*, das er als anonymes und öffentliches Offline-Peer-to-Peer-Netz bezeichnet. Hierbei werden keine Wechseldatenträger oder USB-Sticks weitergegeben, sondern sind fest in Häuserwänden oder sonstigen, festen Gegenständen des alltäglichen Lebens installiert. Somit müssen Benutzer aktiv diese sogenannten Dead Drops aufsuchen und ihren Rechner an diese anschließen, um Dateien von ihnen herunterzuladen oder eigene Dateien auf diese abzulegen. Abbildung 2.1 zeigt einen Dead Drop, der in eine Hausfassade eingelassen ist. Jede Person kann neue Dead Drops installieren und deren Position auf der Internetseite des Projekts veröffentlichen. Begonnen hat das Projekt mit 5 Dead Drops. Knapp über zwei Jahre später waren es aber schon fast 1 200 und täglich kommen neue hinzu. Sollte dieses Kunstprojekt in naher Zukunft großen Anklang finden, so ist dies eine Möglichkeit für Schadprogrammautoren, anonym ihre Schadprogramme zu verteilen und darüber einen großen und heterogenen Personenkreis zu gefährden [Bar13].

Tabelle 2.2 zeigt zusammenfassend einen Überblick über die von uns vorgestellten Verbreitungsroutinen von Schadprogrammen. Für jede einzelne Verbreitungsroutine haben wir ebenfalls die infektionsauslösenden Ereignisse angegeben und ob bei diesen aktive Handlungen der Benutzer erforderlich sind. Bei den Verbreitungswegen, die das Ausnutzen einer Sicherheitslücke in einer speziellen Client-Anwendung beinhalten, haben wir in der Tabelle jeweils angegeben, dass keine Benutzerinteraktion

Abbildung 2.1.: Dead Drop in einer Mauer
Ein in einer Hauswand installierter Dead Drop (USB-Stick), auf den beliebige Personen Dateien hochladen oder Dateien von diesem herunterladen können [Bar13].

erforderlich ist. Maßgebend hierbei ist natürlich, wo man die Grenzen zieht: Beispielsweise kann eine Sicherheitslücke in einem E-Mail-Client nur dann ausgenutzt werden, wenn ein Benutzer den Client öffnet und diesen dazu benutzt, eine bösartig präparierte E-Mail anzuzeigen, die die entsprechende Sicherheitslücke ausnutzt. Sowohl das Öffnen als auch das Auswählen der E-Mail sind beides Benutzerinteraktionen. Allerdings muss der Empfänger nicht dazu gebracht werden, einen Anhang herunterzuladen oder einem Verweis zu folgen. Die Angaben der letzten Tabellenspalte beziehen sich immer auf den Zeitpunkt nach dem Öffnen einer Datei in dem entsprechenden Client. Gleiches gilt für eine Verbreitung per Drive-By-Download. Der Aufruf einer präparierten Internetseite in einem Webbrowser stellt zwar eine Benutzerinteraktion dar, aber für eine Infektion ist kein aktives Herunterladen und Ausführen einer Datei seitens des Benutzers notwendig. Auch hier bezieht sich die Angabe der Tabelle auf den Zeitpunkt nach dem Öffnen der Internetseite.

2.3. Social Engineering

Ein wichtiger Aspekt bei der Verbreitung von Social Malcode stellt das Social Engineering dar. Der Autor eines Schadprogramms versucht potentielle Opfer derart zu täuschen oder zu manipulieren, dass diese das Programm ausführen beziehungsweise installieren. Das Ziel dieses Abschnitts ist es, einen Überblick zum Thema Social Engineering zu geben. Die Literatur zu diesem Thema ist sehr vielfältig, da der Kreativität des Menschen kaum Grenzen gesetzt sind. Das bezieht sich sowohl auf die Ziele als auch auf die Methoden, mit denen Angreifer versuchen, unerlaubten Zugang zu

Tabelle 2.2.: Häufig verwendete Verbreitungsroutinen
Dies ist ein Überblick über die am häufigsten verwendeten Verbreitungsroutinen von Schadprogrammen. Zu jeder Verbreitungsroutine ist mindestens ein Ereignis angegeben, das die Infektion auslöst und ob dazu eine Benutzerinteraktion erforderlich ist.

Verbreitungsroutine	Infektionsauslösendes Ereignis	Erfordert Benutzerinteraktion?
Netzdienste mit Sicherheitslücken	Liste der IP-Adressen der nächsten Opfer enthält die eigene IP-Adresse	Nein
World Wide Web	Manuelles Herunterladen und Ausführen einer Datei	Ja
	Drive-By-Download nach dem Öffnen einer präparierten Internetseite	Nein
E-Mail	Dateianhang ausführen	Ja
	Folgen eines Verweises auf eine Internetseite	Ja
	Sicherheitslücke im Client wird ausgenutzt	Nein
Instant Messanging	Dateitransfer annehmen	Ja
	Folgen eines Verweises auf eine Internetseite	Ja
	Sicherheitslücke im Client wird ausgenutzt	Nein
Soziale Netzwerke	Folgen eines Verweises auf eine Internetseite	Ja
Filesharing	Manuelles Herunterladen und Ausführen einer Datei	Ja
	Sicherheitslücke im Client wird ausgenutzt	Nein
Netzfreigaben	Ausführen einer entfernt freigegebenen Datei	Ja
Wechseldatenträger	Ausführen einer Datei des Datenträgers	Ja
	Aktivierter AutoRun: Anschließen des Datenträgers	Ja

Daten und Informationen zu bekommen – was stets das Ziel des Social Engineering ist.

In Abschnitt 2.3.1 wird beschrieben, was man allgemein unter dem Begriff Social Engineering versteht und verschiedene Definitionen der Literatur werden aufgeführt. Anschließend beleuchtet der Abschnitt *Motive und Ziele* die Intention eines Angreifers und geht der Frage nach, warum ein Angreifer überhaupt einen solchen Angriff durchführt und was er üblicherweise durch diesen erreichen möchte. Der Frage nach dem „Wie“ widmet sich Abschnitt 2.3.4, der die typischen Vorgehensweisen und Techniken beim Social Engineering vorstellt. Bevor das Thema Social Engineering durch eine Vorstellung von Gegenmaßnahmen abgeschlossen wird, werden in Abschnitt 2.3.5 einige Erklärungsversuche aus der Literatur präsentiert. Diese beleuchten die psychologischen Aspekte (vor allem auf Seiten der Opfer) eines solchen Angriffs.

2.3.1. Definition

Eine eindeutige und allgemein akzeptierte Definition für Social Engineering zu finden, ist nur schwer bis nahezu unmöglich. Es existiert eine Vielzahl an Methoden und Techniken, die für einen Social-Engineering-Angriff eingesetzt werden, jedoch nicht prägnant zusammengefasst werden können. In der Literatur zu diesem Thema finden sich viele Definitionsansätze, die mehr oder weniger eine ähnliche Kernaussage beinhalten. Die Gewichtung und Beschreibung der Methoden und/oder Techniken unterscheiden sich jedoch meist recht stark [Gra01].

Der ehemalige Hacker und heutige Sicherheitsexperte Kevin Mitnick, der als der bekannteste Social Engineer gilt, schrieb zum Thema Social Engineering das Buch *Die Kunst der Täuschung* [MS06]. Der Titel des Buches beschreibt das, was die meisten Autoren generell unter Social Engineering verstehen, am treffendsten: Es ist eine Kunst – wie die Rolle in einem Schauspiel. Ein Social Engineer spielt dementsprechend eine Rolle. Er benutzt eine falsche Identität, um an Informationen zu gelangen, ohne dabei in ein Computer System einbrechen zu müssen. Dabei täuscht und manipuliert er seine Opfer mit technologischen und psychologischen Tricks dahingehend, dass sie das machen, was der Angreifer von ihnen möchte. Social Engineering nutzt menschliche Schwächen durch Lügen und Manipulationen aus. Meistens steht das natürliche Verlangen der Menschen nach Vertrauen im Mittelpunkt des Angriffs [Gra01]. Für den Angreifer gilt es, seine Rolle so überzeugend zu spielen, dass sein Opfer keinen Verdacht schöpft. Dazu gehört ein gewisse Routine und eine kriminelle Energie, die nicht jeder wie selbstverständlich in sich trägt. James E. Keeling nennt dies auch „die praktische Anwendung soziologischer Prinzipien für spezielle soziale Probleme“ [Kee01].

In einem Beitrag für den *SANS Institute Reading Room* zitiert Malcom Allen einen unbekannten Autor, der Social Engineering als *nicht-technische* oder *low-tech-Attacke* beschreibt, um beispielsweise durch Lügen, Identitätsdiebstahl, Tricks, Bestechungen, Erpressungen oder Drohungen ein Informationssystem anzugreifen [All06]. Diese Beschreibung verdeutlicht sehr gut, dass Social Engineering, im Gegensatz zu einer high-tech-Attacke, oft mit einfachen technischen Mitteln auskommt und auch funktioniert.

Karen J. Bannan definiert einen Social Engineer als einen Hacker, der eher seinen Kopf als die Rechenleistung eines Computers einsetzt, um zum gewünschten Ziel zu kommen [Ban01]. Das Ziel sind immer Informationen, zu denen der Angreifer regulär keinen Zugriff hat und auch nicht haben sollte. In der Regel sind dies öffentlich nicht zugängliche Daten, die interessante Objekte für eine Attacke darstellen. Mitnick betont, dass ein Angriff mittels Social Engineering meist sehr komplex und langwierig ist. Manchmal muss der Social Engineer seine Vorgehensweise tagelang planen und vorbereiten, um seine Ziele zu erreichen und um dabei nicht aufzufallen [MS06]. Er erschleicht sich in seiner Rolle das Vertrauen der Opfer.

2.3.1.1. Angriffsablauf

Gartner Inc. entwickelte 2005 ein kreisförmiges Ablaufschema eines typischen Social-Engineering-Angriffs, das einen definierenden Charakter besitzt und ebenfalls die Komplexität eines Social-Engineering-Angriffs hervorhebt [ANFM05]. Dieses Sche-

ma ist in einer leicht abgewandelten und übersetzten Form in Abbildung 2.2 dargestellt und besteht aus vier Phasen. Der zyklische Aufbau des Schemas verdeutlicht, dass ein Social Engineering Angriff aus mehreren, sich ständig wiederholenden Phasen bestehen kann. Dies ist sogar recht häufig der Fall, da ein Angreifer nicht sofort an sein Ziel kommt, sondern dieses nur über mehrere, kleinere Zwischenziele erreicht.

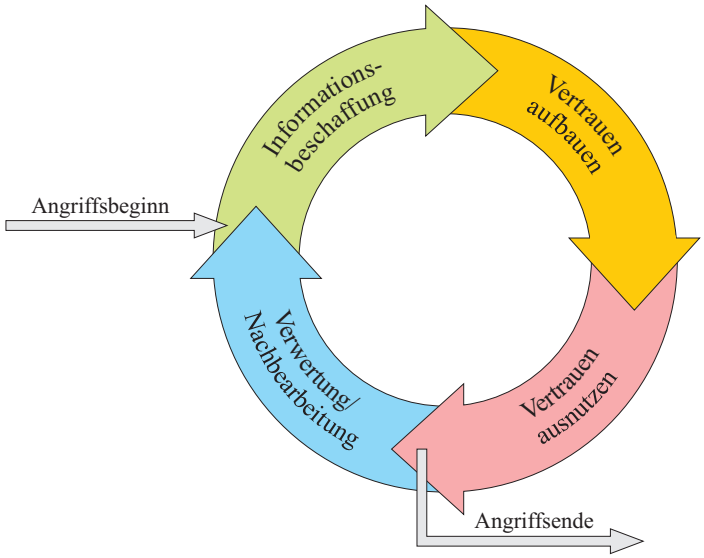


Abbildung 2.2.: Typischer Ablauf eines Social-Engineering-Angriffs
Gartner Inc. unterteilt einen Social-Engineering-Angriff in vier Phasen, die sich wiederholen, bis der Angreifer sein Angriffsziel erreicht hat. *Hinweis: In Anlehnung an [ANFM05]*

1. Phase: **Informationsbeschaffung**
   In der ersten Phase, die den Einstieg in den Angriffsablauf darstellt, versucht der Angreifer möglichst viele Informationen über Personen und Objekte des Zielunternehmens zu sammeln, die ihm beim Erreichen seines Angriffsziels behilflich sein könnten. Dies geht über Telefonlisten oder Mitarbeiterhierarchien/-zusammenhänge hin zur Struktur und Aufbau des Unternehmensnetzes oder Quellcode und Metainformationen über die unternehmensintern verwendete Software.
2. Phase: **Vertrauen aufbauen**
   Ziel der zweiten Phase ist es, das Vertrauen der Mitarbeiter zu gewinnen, die in einer direkten oder indirekten Verbindung mit dem Angriffsziel stehen. Dieser Vorgang kann unter Umständen sehr viel Zeit beanspruchen und basiert auf den gesammelten Informationen der ersten Phase. Beispielsweise kann ein Angreifer in einem ersten Telefongespräch den Gesprächspartner lediglich um einen kleinen Gefallen bitten – dabei aber durch belanglos und nebensächlich erscheinende Fragen und Aussagen eine Vertrauensbasis aufbauen. Dabei ist es natürlich wichtig, dass der Angreifer unter einer falschen Identität auftritt, die dem Gesprächspartner zwar nicht bekannt ist, aber dennoch mit diesem in Zusammenhang steht, etwa als neuer Kollege einer entfernten Außenstelle. Mit der Zeit verschiebt sich dann das Verhältnis zwischen Belanglosigkeiten und für

den Angreifer wichtigen Inhalten. Ein geübter Social Engineer verwendet dabei auch Komplimente, Schmeicheleien oder Techniken wie etwa *Name Dropping* (beiläufig den Namen eines anderen, meist hochrangigen, Mitarbeiters fallen lassen), um das Vertrauen seinen Gegenübers schneller zu gewinnen.

3. Phase: **Vertrauen ausnutzen**
   Hat ein Angreifer erst einmal das volle Vertrauen einer Zielperson, nutzt er dieses dafür aus, das Angriffsziel zu erreichen – zum Beispiel der Erhalt von vertraulichen Daten. Je mehr Zeit ein Angreifer in die zweite Phase des Vertrauensaufbaus investiert hat, desto leichter wird es ihm fallen, seine Ziele in dieser dritten Phase zu erreichen. Dies hat rein menschliche Gründe: Menschen sind schneller dazu bereit einer Person einen Gefallen zu tun, je besser sie diese Personen kennen und je mehr Vertrauen sie ihr entgegenbringen. Sollte das erreichte Ziel das Hauptziel des Angriffs sein, so ist der Angriff beendet.
4. Phase: **Verwertung/Nachbearbeitung**
   Ist das in der dritten Phase erreichte Ziel lediglich ein Zwischenziel, so geht der Angriff über die vierte Phase in den nächsten Zyklus. Dazu benutzt der Angreifer die in der vorherigen Phase erreichten Informationen (die Zwischenziele), um seinen Angriff neu zu überdenken und neu zu planen. Auf diese Weise erhält der Angreifer immer mehr kleine Puzzlestücke, die nach und nach ein großes Gesamtbild ergeben und ihm das Erreichen des Hauptangriffsziels ermöglichen.

Folgendes Beispiel soll einen Angriff mit mehreren Zyklen verdeutlichen: Ein Angreifer möchte die Baupläne für ein Produkt eines Zielunternehmens stehlen, die digital auf einem Server gespeichert sind. Zu Beginn des Angriffs ermittelt er die Telefonnummer einer Unternehmenshotline. Von einem Hotline-Mitarbeiter erhält er jedoch nicht die IP-Adresse des Zielservers, sondern lediglich den Namen und die Durchwahl eines zuständigen Systemadministrators. Diese Daten nutzt der Angreifer, um weitere Informationen über diesen Administrator zu sammeln, um dadurch in der zweiten Phase des zweiten Zyklus dessen Vertrauen zu gewinnen. Obwohl der Angreifer in dem ersten Angriffszyklus nicht sein Hauptziel erreichen konnte, nutzt er die erhaltenen Informationen des Zwischenziels (Name und Durchwahl des Administrators), um sich seinem Hauptziel weiter zu nähern.

2.3.2. Motive

Die Motive eines Social Engineer können sehr unterschiedlich sein. Der Hauptgrund für eine solche Attacke ist in der Regel aber ein finanzielles Interesse. Weitere motivierende Aspekte sind persönliches Selbstinteresse oder auch Rache [All06]. Alle drei Beweggründe werden in den nächsten Abschnitten genauer beleuchtet.

2.3.2.1. Finanzielles Interesse

Bei den finanziellen Interessen unterscheidet man direkte und indirekte finanzielle Interessen. In ersterem Fall versucht ein Angreifer an persönliche Daten, wie etwa die Bankverbindung und oder Zugangsdaten zum Online-Banking zu gelangen, um vom Opfer Geld zu stehlen. Auch elektronische Warenhäuser, die Bankdaten und Kreditkartennummern der Benutzer zur einfacheren und schnellen Bestellung speichern, können

bei Verlust der Zugangsdaten von einem Angreifer benutzt werden, um sich durch Bestellungen auf Kosten des Opfers persönlich zu bereichern.

Angriffe, die durch indirekte finanzielle Interessen motiviert sind, führt ein Social Engineer im Auftrag eines Dritten durch. Ein anschauliches Beispiel dafür ist eine Detektivfirma, die beauftragt wurde, im Zuge einer Scheidung sensible finanzielle Daten über einen der Eheleute zu besorgen, die im Unterhaltsstreit verwendet werden sollen [MS06]. Im Allgemeinen interessiert sich der Auftraggeber nicht für die Methoden, wohl aber für die Resultate der Ermittlungen.

2.3.2.2. Persönliches Selbstinteresse

Ist ein Social-Engineering-Angriff durch persönliches Selbstinteresse motiviert, möchte ein Angreifer an Informationen über Personen im näheren Umfeld gelangen. Das können die Partner sein, aber auch andere Familienmitglieder, Freunde, Kollegen oder Nachbarn. Ob es sich dabei um Misstrauen oder Neugier handelt, ist von Fall zu Fall verschieden.

Auch Eifersucht ist ein Grund, warum die Methoden des Social Engineering angewendet werden, z. B. um die Zugangsdaten des E-Mail-Kontos des Partners herauszubekommen und somit in der Lage zu sein, dessen persönliche Nachrichten zu lesen. Ein anderes Ziel sind die Zugangsdaten zum Online-Banking, um zu überprüfen, wie viel Geld der Partner wann und wofür ausgegeben hat.

2.3.2.3. Rache

Rache ist ebenfalls ein persönliches Selbstinteresse, unterscheidet sich aber in einem wesentlichen Punkt von den beschriebenen Motiven des vorherigen Abschnitts: Möchte sich ein Social Engineer an einer Person rächen, so verschafft er sich die Informationen nicht aus Neugier, sondern um der Zielperson zu schaden. Ist Rache die treibende Kraft hinter einem Social-Engineering-Angriff, so werden die gestohlenen Informationen gezielt dafür eingesetzt, einer Person Schaden zuzufügen. Beispielsweise könnte sich ein Social Engineer an einem ehemaligen Lebenspartner rächen wollen, indem er über E-Mail oder in sozialen Netzwerken kompromittierende Fotos veröffentlicht oder im Namen des Opfers (für dieses unvorteilhafte) Nachrichten verschickt.

Ein weiteres Beispiel für das Motiv Rache ist ein entlassener Mitarbeiter, der sich von seinem ehemaligen Arbeitgeber ungerecht behandelt fühlt und sich dafür revanchieren will. Das Gefährliche daran ist, dass ein ausgeschiedener Mitarbeiter meist auf Insiderwissen zurückgreifen kann. Damit wäre eine böswillige Attacke ohne Weiteres möglich. Die Bandbreite solch einer Racheaktion kann sehr groß sein und reicht von einem bösen Streich bis hin zur Sabotage oder Diebstahl von Firmengeheimnissen.

2.3.3. Ziele

All jene Dinge eines Opfers, die die drei beschriebenen Motive und damit auch die Bedürfnisse eines Social Engineer befriedigen können, sind potentielle Ziele eines Social-Engineering-Angriffs. Im Allgemeinen ist es schwierig, Beispiele für die Ziele

eines Social-Engineering-Angriffs aus der Praxis zu finden. Dies ist vor allem durch zwei Dinge begründet:

- Zum einen muss das Opfer eines Social-Engineering-Angriffs erkennen, dass ihm etwas gestohlen oder abhanden gekommen ist. Im Gegensatz zu einem technischen Einbruch in ein Firmennetz, ist dies beim Social Engineering jedoch schwieriger, da kaum bis keine digitale Spuren hinterlassen werden.
- Zum anderen machen Unternehmen einen entdeckten Angriff meistens nicht publik. So wird auch nicht öffentlich bekannt, welche Informationen oder Daten Ziel des Angriffs waren. Stattdessen wird unternehmensintern ermittelt und den Vorkommnissen nachgegangen. Auf diese Weise wird versucht, negative Presse zu vermeiden und sowohl Kunden als auch Geschäftspartner nicht abzuschrecken. Andernfalls könnte deren Vertrauen in die unternehmenseigene IT-Sicherheit erschüttert werden. Damit einher geht eine oftmals schlechte Dokumentation des Vorfalls, wodurch nachträglich nicht genau entschieden werden kann, ob es sich bei einem Angriff um einen Social-Engineering-Angriff handelte oder nicht [Gra01].

Wie in Abschnitt 2.3.1.1 beschrieben, besteht ein Social-Engineering-Angriff aus mehreren Phasen. Bei einem professionellen Angriff auf eine Firma versucht der Angreifer sowohl Daten zu erhalten, die für ihn von Interesse sind (Hauptangriffsziel), als auch Informationen, die dabei helfen, das Hauptziel zu erreichen (Zwischenziele). Obwohl sich aus genannten Gründen öffentlich kaum dokumentierte Beispiele für die Hauptziele eines Social-Engineering-Angriffs finden lassen, sind die Zwischenziele oftmals gleichbleibend. Beispiele dafür sind:

- Telefonbücher, die Namen der Mitarbeiter und deren Durchwahlnummern enthalten, für die sich der Social Engineer im weiteren Verlauf des Angriffs ausgeben kann.
- Aufzeichnungen über die Unternehmenshierarchie, die Angestellte in wichtigen Positionen aufzeigen.
- Handbücher zur Firmenpolitik, die Auskunft über die Sicherheitsrichtlinien enthalten können.
- Informationen zur IT-Infrastruktur, die dem Angreifer Hinweise auf mögliche Schwachstellen liefern.
- Auch ausrangierte Hardware kann von besonderem Interesse sein, wobei speziell alte Datenträger (wenn deren Inhalt nicht sicher gelöscht wurde) besonders viele nützliche Informationen liefern können [Gra01].

Weitere Beispiele sind Kundenlisten und -abrechnungen, Geschäftspläne, Expansionspläne, Marketingpläne, vertrauliche finanzielle Informationen, Aufzeichnungen über Produktions- und Geschäftsprozesse oder technische Zeichnungen der Forschungs- und Entwicklungsabteilung eines Unternehmens [Rob07].

2.3.4. Methoden

Die Vorgehensweise bei einem Angriff durch Social Engineering ist sehr vielfältig. Diese kann aus der Distanz per E-Mail oder Telefon erfolgen, aber auch durch persön-

liches Auftreten direkt vor Ort [Gra01, MS06]. Letzteres ist seltener, da dies für den Social Engineer die Gefahr birgt, eher mit dem Angriff in Verbindung gebracht zu werden und somit die Erfolgswahrscheinlichkeit des Angriffs schmälert. Normalerweise agiert der Social Engineer in sicherer Distanz zum Opfer und sollte nicht länger am Tatort verweilen als unbedingt nötig.

Vor allem bedarf es einer guten Vorbereitung, um einen Angriff erfolgreich durchführen zu können. Das gilt als wichtigste Voraussetzung, um auf jede etwaige Situation richtig reagieren zu können. Daher sollte der Angreifer bestimmte Informationen wie relevante Namen, den üblichen Firmenjargon oder auch Details zur Struktur und den Geschäftsbeziehungen der Zielorganisation parat haben. Anderenfalls läuft er Gefahr, schnell der Lüge überführt zu werden, falls ein Opfer Verdacht schöpft. Außerdem könnte die Zielperson noch Kollegen oder Vorgesetzte warnen, wodurch es für den Social Engineer sehr schwer bis unmöglich wird, seinen Angriff fortzusetzen [MS06].

Neben einer akribischen und gewissenhaften Vorbereitung verwendet ein Social Engineer verschiedene Techniken und Methoden, um an sein Ziel zu kommen. Die folgenden Abschnitte stellen einige dieser Methoden vor. Diese Liste der hier vorgestellten Methoden ist jedoch nicht vollständig. Zudem lassen sich die Techniken durch einen Angreifer beliebig kombinieren oder abwandeln. Eine gute Übersicht, viele anschauliche Beispiele und weitere, zum Teil auch sehr ausgefallene Techniken befinden sich im Buch von Mitnick [MS06].

Der Einfachheit halber wird in den nächsten Abschnitten davon ausgegangen, dass das Angriffsziel eine Organisation ist. Alle Methoden lassen sich aber genauso gut auf einzelne Zielpersonen anwenden.

2.3.4.1. Nachahmung

Eine sehr häufig verwendete Methode eines Social Engineer ist es, sich als eine andere Person auszugeben, um so an für ihn interessante Informationen zu gelangen. Entweder die Nachahmung findet aus sicherer Distanz statt, z. B. durch ein Telefonat oder per E-Mail, oder der Angreifer legt sich eine Tarnung zu und tritt unter einer falschen Identität direkt in der Zielorganisation auf. Im zweiten Fall kann man noch unterscheiden, ob sich der Social Engineer als Mitarbeiter der Zielorganisation oder einer externen Organisation ausgibt. Ein entscheidender Faktor für den Erfolg des Angriffs spielt dabei die verwendete Tarnung. Je authentischer der Angreifer in der Zielfirma erscheint und auftritt, desto besser sind seine Erfolgsaussichten. Dabei kann die Tarnung von einem einfachen Ausdruck eines Namensschildes oder einer Visitenkarte bis hin zur aufwendigen Verkleidung oder qualitativ hochwertigen Fälschung von Sicherheitsausweisen gehen [App09].

Das Nachahmen von Mitarbeitern einer Zielorganisation direkt vor Ort führt ein Angreifer üblicherweise nur in größeren Unternehmen durch, da es in kleineren wahrscheinlicher ist, dass sich die Angestellten untereinander kennen. Gibt sich der Angreifer als Mitarbeiter der Zielorganisation aus, wählt er als nachzuahmende Person in der Regel einen Vorgesetzten oder Personen, die diesem nahestehen, z. B. die Assistenz der Geschäftsleitung. Anzumerken ist bei dieser Variante, dass üblicherweise mehrere Hierarchieebenen zwischen der vermeintlichen Person und dem Opfer liegen, da man im Allgemeinen seinen direkten Vorgesetzten kennt. Zudem ist als Vorgesetzter die

Wahrscheinlichkeit, dass der Angreifer sein Ziel erreicht, höher als wenn er die Rolle eines Mitarbeiters auf der Hierarchieebene des Opfers spielt.

Eine weitere Möglichkeit ist das Annehmen von Identitäten von unternehmensexternen Angestellten. Ein gutes Beispiel dafür ist das Nachahmen eines Mitarbeiters eines Handwerkerbetriebs. Der Vorteil ist, dass der Angreifer sofort als unternehmensfremder Angestellter zu erkennen ist und somit keinen Verdacht dadurch erweckt, dass er keinem Mitarbeiter der Zielorganisation bekannt ist. Auch möglich ist das Auftreten eines Social Engineer als Vertreter einer für die Zielorganisation üblichen Consulting-Firma oder eines Lieferanten. Dabei ist, wie bereits erwähnt, eine gewisse Vorkenntnis über die Geschäftsbeziehungen der Zielorganisation erforderlich.

Sobald ein Social Engineer physischen Zutritt zur Zielorganisation erlangt hat, wird er versuchen, in irgendeiner Art und Weise an die Zielinformationen zu gelangen. Findet er beispielsweise ein unbesetztes Büro, kann es für ihn ein Leichtes sein, Zugriff auf das Netzwerk zu bekommen und Daten per USB-Stick zu entwenden oder diese einfach auszudrucken. In der Praxis hat sich gezeigt, dass viele Personen ihre Zugangsdaten am Monitor, am Mousepad oder unter der Tastatur aufbewahren. Diese Fahrlässigkeit macht es dem Social Engineer natürlich wesentlich leichter, seine Ziele zu erreichen [MS06]. In Büros und Räumen, zu denen selbst die imitierte Person normalerweise keinen Zutritt hat, versucht ein Angreifer so schnell wie möglich zu agieren, da immer die Gefahr besteht, entdeckt zu werden.

Auch das Verwenden von gestohlenen Zugangsdaten für Online-Dienste des Inter- oder Intranets bezeichnet man als Nachahmung. In diesem Fall wird keine physische, sondern eine digitale Identität angenommen. Aus Sicht des Online-Dienstes gibt es in solchen Fällen kaum Möglichkeiten, zu erkennen, ob dies ein autorisierter Zugriff des Mitarbeiters oder ein Zugriff eines Angreifers mit gestohlenen Zugangsdaten ist [App09, HEF09].

2.3.4.2. Dumpster Diving

Als *Dumpster Diving* bezeichnet man das Durchsuchen und -stöbern des Abfalls einer Zielorganisation und es erfordert somit logischerweise das physische Auftreten des Angreifers bei der Zielorganisation. Begünstigt wird Dumpster Diving durch eine zu unachtsame Entsorgung von Organisationseigentum – in der Regel ohne das Schreddern von Dokumenten oder Verzicht auf das sichere Löschen von Datenträgern [App09].

Dumpster Diving kann zum einen den Papierkorb einer bestimmten Person an deren Schreibtisch als Ziel haben. Dabei kann man persönliche Notizen des Mitarbeiters finden, z. B. Memos zu bestimmten Arbeiten, aber auch Informationen über Dienstreisen oder Urlaubszeiten, die der Angreifer nutzen kann, um die weitere Vorgehensweise zu planen. Wenn er weiß, wann welcher Mitarbeiter nicht vor Ort ist, kann er dies benutzen, um beispielsweise einen anderen Mitarbeiter damit unter Druck zu setzen, dass er dringend irgendwelche Daten benötigt und seinen Ansprechpartner nicht erreichen kann. Da der Angerufene weiß, dass der entsprechende Kollege nicht anwesend ist, wird er im günstigsten Fall bereitwillig die gewünschten Informationen herausgeben.

Zum anderen kann Dumpster Diving aber auch den Müllcontainer der Firma zum Ziel haben, der sich im Allgemeinen im Außenbereich befindet [MS06]. Ist dieser nicht verschlossen und öffentlich zugänglich, ist es für jemanden, der sich für den Inhalt interessiert, in vielen Regionen Deutschlands (momentan) noch nicht einmal illegal, den Inhalt zu durchsuchen. Dieser besteht unter Umständen nicht nur aus Dokumenten, die achtlos entsorgt wurden und eventuell geheime Informationen enthalten, sondern auch aus Speichermedien wie CDs oder sogar ganzen Festplatten, die nicht sicher gelöscht wurden und noch vertrauliche Daten enthalten.

2.3.4.3. Shoulder Surfing

Auch das *Shoulder Surfing* erfordert, dass der Angreifer persönlich in der Zielorganisation anwesend sein muss – sieht man von Außendienstmitarbeitern ab. Zweck des Shoulder Surfing im engeren Sinn ist das unauffällige Zusehen bei Tastatureingaben von vertraulichen Daten. Zum Beispiel kann ein Angreifer einem Mitarbeiter der Zielorganisation dabei zusehen, wie sich dieser mit seinen internen Zugangsdaten bei einem Dienst authentifiziert. In diesem Fall ist es das Ziel des Angreifers, sich die Zugangsdaten zu merken und später selbst zu verwenden [All06]. Shoulder Surfing im weiteren Sinn bezeichnet aber auch das heimliche Mitlesen von vertraulichen Informationen von einem Monitor oder aus einem Dokument.

Der Name Shoulder Surfing kommt daher, dass diese Technik für einen Angreifer am leichtesten ist, falls er sich nicht im Blickfeld des Opfers befindet – etwa wenn er hinter dem Opfer steht und ihm bei der Eingabe über die Schulter schaut. Um während der Tastatureingabe in der Nähe des Opfers zu sein, kann ein Angreifer diese Technik mit der *Nachahmung* kombinieren: Der Angreifer tritt beispielsweise als Mitarbeiter der IT-Abteilung auf und gibt an, dass es Probleme mit dem Dienstkonto des Opfers geben würde und sich dieser testweise bei dem betroffenen Dienst anmelden soll.

2.3.4.4. Reverse Social Engineering

Reverse Social Engineering zielt darauf ab, dass das eigentliche Opfer der Attacke den Social Engineer aufsucht, um ihn um Hilfe zu bitten – die Kontaktaufnahme also umgekehrt (engl: *reverse*) erfolgt. Es stellt sich die Frage, warum ein Opfer freiwillig den Kontakt zu einem Angreifer suchen sollte.

Folgendes Beispiel macht diesen zuerst unlogisch erscheinenden Vorgang deutlich: Zur Vorbereitung des Reverse Social Engineering erscheint der Angreifer etwa als neuer Support-Mitarbeiter oder freundlicher Kollege, der seine Hilfe anbietet, falls es irgendwann Schwierigkeiten mit dem Computer geben sollte. Dabei erwähnt er, dass angeblich bei anderen Mitarbeitern bereits Probleme aufgetreten sind und erkundigt sich, ob das Opfer schon ein ähnliches Verhalten beobachten konnte. Dieser verneint diese Frage in der Regel, da das beschriebene Problem lediglich vom Angreifer erfunden wurde und somit das zufällige Auftreten eher unwahrscheinlich ist. Der nächste Schritt besteht darin, dass der Angreifer das erwähnte Problem künstlich verursacht, so dass dieses bei dem Opfer des Angriffs auftritt. Der Angreifer hofft nun darauf, dass sich das Opfer an den freundlichen und hilfsbereiten Kollegen erinnert und diesen kontaktiert. Zwischen dem Ansprechen des Kollegen und dem künstlichen Verursachen des

beschriebenen Problems sollte natürlich eine gewisse Zeit vergehen, damit das Opfer nicht misstrauisch wird und sich über den „zufälligen“ Verlauf der Geschehnisse wundert.

Nelson formalisiert diese Art des Social Engineering und formuliert drei Schritte, die bei dieser Technik zum Einsatz kommen [Nel02]:

1. **Sabotage**: Im Gegensatz zum obigen Beispiel sorgt ein Angreifer in Nelsons Modell bereits in dem ersten Schritt dafür, dass das Opfer auf ein Problem stößt, das es alleine nicht lösen kann.
2. **Ködern**: Als nächstes muss der Angreifer dafür sorgen, dass das Opfer möglichst den Angreifer und niemanden anderen kontaktiert, damit ihm geholfen wird. Dafür kann er sich etwa wie in obigem Beispiel persönlich als kompetenter Ansprechpartner vorstellen, aber beispielsweise auch Visitenkarten oder Werbung in der Nähe des Opfers auslegen.
3. **Helfen**: Nachdem das Opfer Kontakt zu dem Angreifer aufgenommen hat, hilft dieser dem Opfer dabei, das angebliche Problem zu beheben. Ganz beiläufig versucht er gleichzeitig dem Opfer die für ihn interessanten Informationen zu entlocken. Sollte ihm dies nicht gelingen, kann er durch die geleistete Hilfe versuchen, das Vertrauen des Opfers zu gewinnen, um dieses in dem weiteren Angriffsverlauf auszunutzen.

Das Besondere an dieser Technik des Social Engineering ist, dass der Social Engineer nicht erst das Vertrauen der Person gewinnen muss und somit die zweite Phase eines typischen Social-Engineering-Angriffs entfällt (siehe Seite 31). Ein Angreifer, der das Opfer dazu bringt, ihn anzurufen, erlangt sofortige Glaubwürdigkeit [MS06]. Da das Handeln der Zielperson selbstinitiiert ist, wird die Zielperson häufig bedenkenlos den Anweisungen des Angreifers folgen, z. B. der Aufforderung, auf einer bestimmten Internetseite ein Programm herunterzuladen und zu installieren. Anstatt eines Programms zur Behebung des vermeintlichen Problems handelt es sich dabei für gewöhnlich um ein Schadprogramm, das dazu dient, Daten auszuspionieren oder dem Angreifer für einen späteren Zugriff auf den Rechner eine Hintertür einzurichten [MS06].

Ein weiterer Vorteil des Reverse Social Engineering ist, dass dieses sehr schwer zu erkennen ist und meist sogar komplett unerkannt bleibt. Dies liegt daran, dass das Problem, das der Grund für die Kontaktaufnahme war und lediglich künstlich durch den Angreifer hervorgerufen wurde, abschließend wieder durch den Angreifer behoben wird. Auf diese Weise schöpft das Opfer keinen Verdacht und freut sich über den hilfsbereiten und kompetenten Mitarbeiter und darüber, dass es nun wieder problemlos arbeiten kann.

2.3.4.5. Phishing

Auch Software oder Internetseiten können von einem Social Engineer eingesetzt werden, um menschliche Schwächen eines Opfers auszunutzen und seine Ziele zu erreichen. Eine Variante des computerbasierten Social Engineering ist das sogenannte *Phishing*. Darunter versteht man die Versuche, über gefälschte Internetseiten an die Zugangsdaten eines Opfers zu gelangen, wie zum Beispiel die Zugangsdaten fürs

Online-Banking oder das Login-Passwort für einen Online-Dienst. Daher kommt auch der Name: Phishing ist ein Kunstwort aus den beiden englischen Begriffen *Password* und *Fishing* – bezeichnet also das Angeln/Ködern von Passwörtern.

Die Internetseiten, auch Phishing-Seiten genannt, imitieren meist perfekt die Internetseiten der Online-Dienste, deren Kundenpasswörter der Angreifer stehlen möchte [DTH06]. Auf diese Weise werden die Opfer getäuscht und denken, sie würden ihre Zugangsdaten dem regulären Online-Dienst übermitteln, stattdessen landen sie aber beim Angreifer, der die Phishing-Seite aufgesetzt hat. Um die Opfer auf die Phishing-Seite zu locken, muss der Angreifer diese den Opfern bekannt machen. Dies kann mündlich oder auch schriftlich durch Hinterlassen des *Uniform Resource Locator* (URL) passieren. Die aber wohl häufigste Methode besteht darin, sogenannte Phishing-Mails zu verschicken. Dies sind reguläre E-Mails, in denen dem Empfänger (dem Opfer) die Benutzung des angeblichen Online-Dienstes schmackhaft gemacht wird. Damit der Empfänger den Dienst nutzen kann, befindet sich innerhalb der E-Mails ein Verweis auf den Online-Dienst. Das Thema Phishing wird in Abschnitt 4.2.4 nochmals genauer behandelt, in dem unter anderem auch eine selbst erstellte Phishing-Seite und Phishing-Mail abgebildet sind.

Eine speziellere Möglichkeit des Phishing ist das *Spear-Phishing*, das im Vergleich zum normalen Phishing eine gezielte Attacke auf einzelne Personen oder eine kleinere Gruppe von Personen darstellt, oder das *Whaling*, bei dem gezielt Führungspersonen im Fokus des Phishing-Angriffs stehen [JM06]. Die Gefahr für die Opfer ist bei diesen Phishing-Varianten ungleich höher, da die Phishing-Mails hier durch den speziellen und kleineren Opferkreis in der Regel eine personalisierte Betreffzeile und/oder Anrede aufweisen und sich im Inhalt das Wissen um persönliche Umstände der Opfer zunutze machen.

Neben dem reinen Aufsetzen von Phishing-Seiten existieren auch Varianten des Phishing, die durch Schadprogramme ermöglicht werden: Beim sogenannten *Pharming* werden zusätzlich die DNS-Anfragen von Webbrowsern so manipuliert, dass der Benutzer selbst dann die Phishing-Seite angezeigt bekommt, wenn er die reguläre URL eines entsprechenden Online-Dienstes in die Adressleiste des Webbrowsers eingibt [KSTW07]. Eine weitere Phishing-Variante bilden Banking-Trojaner, die in die Anmelde-Formulare von regulären Bank-Internetseiten zusätzliche Eingabefelder einblenden und deren Inhalt beim Abschicken des Formulars dem Angreifer übermitteln [She08]. Abbildung 2.3 zeigt modifizierte Anmelde-Formulare bekannter, amerikanischer Banken. Zusätzlich zu den regulären Eingaben soll in diesen Beispielen ein Opfer seine ATM-Nummer[1], seine Sozialversicherungsnummer[2] oder den Geburtsnamen der Mutter (engl. *mother's maiden name*, kurz *MMN*) eingeben – dieser kann von einem Angreifer für eine möglicherweise später zu beantwortende Sicherheitsabfrage verwendet werden.

[1] Eine Art persönliche Identifikationsnummer amerikanischer Banken – ATM ist die Abkürzung von *Automated Teller Machine* (die amerikanische Bezeichnung für einen Geldautomaten)

[2] Eine der wichtigsten Nummern, die in den USA ausgegeben werden, ist die „Social Security Number" (kurz SSN). Sie dient unter anderem der Identifikation der jeweiligen Person und wird für sehr viele geschäftliche Transaktionen benötigt [Jus03].

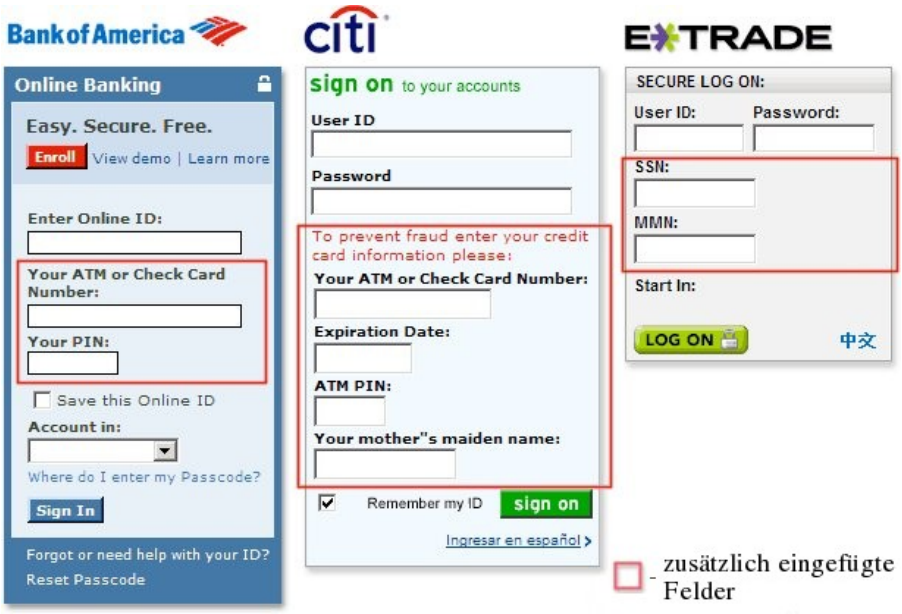


Abbildung 2.3.: Durch Trojaner modifizierte Anmelde-Formulare von Bankseiten
Banking-Trojaner sind heutzutage in der Lage, einen Webbrowser so zu modifizieren, dass dieser zusätzliche Eingabefelder auf Anmeldeformularen von Online-Banking-Internetseiten anzeigt [She08].

2.3.4.6. Post

Wenn ein Social Engineer die normale Briefpost für einen Angriff benutzt, könnte man das als die klassische Version des Spear-Phishing bezeichnen, da der anvisierte Personenkreis eher klein ist. Der Trick dabei ist, einen Brief offiziell und vertrauenswürdig erscheinen zu lassen. Dazu kann er professionelle Desktop-Publishing-Programme wie z. B. *QuarkXpress* [QI11] oder *Pagemaker* [ASI11] benutzen, aber auch einfache Textverarbeitungsprogramme erlauben es heutzutage schon, täuschend echt aussende Geschäftskorrespondenz nachzugestalten. Dabei muss ein Angreifer auch besonderen Wert auf den Briefkopf und das Firmenlogo der Zielorganisation legen, da dieses einem Opfer meist gut bekannt ist. Am besten funktioniert dies in Verbindung mit einem Wasserzeichen und gefälschtem Stempel [Ano97].

Diese Methode des Social Engineering funktioniert, weil Menschen dem gedruckten Wort mehr Glauben schenken als beispielsweise einem elektronischen Dokument oder einer E-Mail. Ein weiterer Vorteil liegt darin, dass normale Briefpost meist keiner automatischen Überprüfung auf Authentizität unterzogen wird, wie dies oftmals bei elektronischen Dokumenten durch digitale Zertifikate der Fall ist [Ano97].

Eine andere Möglichkeit, den Postweg zu nutzen, bietet sich einem Angestellten an, wenn er Daten seines Arbeitgebers stehlen möchte, aber sein Unternehmen sehr auf IT-Sicherheit bedacht ist: Wenn es ihm nicht möglich ist, ein Speichermedium persönlich mitzubringen, weil er beim Betreten und Verlassen der Firma kontrolliert wird, kann er sich z. B. einen USB-Stick per Post zur Arbeitsstelle schicken lassen. Für den Weg heraus aus dem Unternehmen kann er dann ebenfalls den Postweg benutzen,

da dieser für gewöhnlich nicht überprüft wird. Falls eine Überprüfung stattfindet, wie zum Beispiel bei Regierungsorganisationen, dann ist dies meist nur eine Überprüfung auf explosive Stoffe und Inhalte [Rob07].

2.3.4.7. Soziale Netzwerke

Der Erfolg und die große Mitgliederzahl von sozialen Netzwerken hat auch die Gefahr des Informationsdiebstahls erhöht. Dabei kommt es einer Person mit kriminellen Absichten entgegen, dass viele Mitglieder recht unbedarft mit persönlichen Daten umgehen. Sie geben viele private Informationen preis, die sich ein Social Engineer zunutze machen kann, sogar völlig legal, wenn diese durch den Nutzer nicht entsprechend in den Privatsphäreneinstellungen geschützt werden. Es herrscht ein implizites Vertrauen in diese Seiten [Bro10].

Gefährlich sind auch die scherzhaft/unterhaltsam gemeinten Quizfragen zur Persönlichkeit des Nutzers, bei denen nicht klar ist, wer alles Zugriff auf die Antworten hat. Generell nicht vertrauenerweckend sind die Lizenzbedingungen von diversen Applikationen, bei denen man unter Umständen dem Entwickler/Anbieter vollen Zugang zu persönlichen Daten gewähren muss, um diese zu nutzen [Bro10].

Aus der Summe der Informationen ergibt sich ein recht umfangreiches Profil des Nutzers, mit Name, E-Mail-Adresse, persönlichen Angaben und eventuell den Ergebnissen aus den oben genannten Quizfragen. Diese Informationen können insbesondere zur Vorbereitung eines Social-Engineering-Angriffs verwendet werden, um schnell einen Zugang zum Opfer zu finden und leichter dessen Vertrauen zu gewinnen. Damit sind Angriffe wie Spear-Phishing leicht möglich, indem etwa innerhalb einer Nachricht ein Verweis auf ein Video verschickt wird, das ein für das Opfer interessanten Inhalt thematisiert. Häufig sind diese Videos angeblich erst nach einem Update des Videoplayers anzusehen – in Wirklichkeit lädt sich das Opfer dadurch aber ein Schadprogramm auf den eigenen Rechner. Weitere Möglichkeiten, die Informationen aus einem sozialen Netzwerk zu nutzen, sind Identitätsdiebstahl oder gar persönliche Attacken wie zum Beispiel ein Einbruch, da man durch die vom Opfer freiwillig veröffentlichten Informationen weiß, wann sich dieses im Urlaub befindet [Bro10].

2.3.5. Psychologische Aspekte

Angriffe eines klassischen Hackers zielen zumeist auf die technischen Schwachstellen eines Computersystems ab. Social Engineering nimmt hingegen eher die menschlichen Schwächen der Computerbenutzer ins Visier. In der Literatur wird dies auch als Angriff auf die *Wetware* bezeichnet und dient als Ergänzung bzw. Abgrenzung der Termini *Hardware*[3] und *Software*[4]. Ein Social Engineer nutzt also Schwachstellen der Wetware aus, wohingegen klassische Computerangriffe meist Schwachstellen der Software und in seltenen Fällen auch Schwachstellen der Hardware ausnutzen [Pel06, App09].

Peltier definiert Wetware als „den Menschen, der den Computer bedient“ (engl: *the human being attached to a computer system*). Die Hard- und Software eines Rechners

[3] Mechanische und elektronische Ausrüstung eines Computersystems
[4] Programmierbare Konfigurations- und Instruktionsdaten der Hardware

ist relativ einfach gegen Angriffe abzusichern, die Wetware hingegen ist immer wieder der Grund für Sicherheitsvorfälle und somit stellt der Mensch das schwächste Glied der IT-Sicherheitskette dar [Pel06]. Die Ansatzpunkte für Social-Engineering-Angriffe sind menschliche Schwächen[5] wie zum Beispiel das grundsätzliche Vertrauen in andere Menschen, die Autorität in Verbindung mit Angst oder Pflichtgefühl oder aber ein allgemeines Verlangen nach Konformität im täglichen Leben [Gra01, App09]. Welche psychologischen Aspekte eine Rolle beim Ausnutzen dieser Schwächen spielen, wird in den folgenden Abschnitten beschrieben – dies sind die eigentlichen Gründe dafür, dass Social Engineering funktioniert.

2.3.5.1. Vertrauen

Das Vertrauen der Menschen in andere Menschen ist wohl die größte Schwäche, die das Social Engineering ausnutzt. Im Allgemeinen geht jeder Mensch, der zuvor nicht zu viele negative Erfahrungen diesbezüglich gemacht hat, als erstes davon aus, dass man seinem Gegenüber vertrauen kann und dieser einem selbst nichts Böses möchte [Wor07]. Dementsprechend erweckt eine nebensächlich erscheinende Frage am Telefon nach eigentlich nicht für den Anrufer bestimmten Informationen nicht sofort den Verdacht eines Angriffs, da der Angerufene dahinter keine betrügerische Absicht vermutet. Genau diese psychologische Tatsache macht sich ein Angreifer zunutze, um seine Ziele zu erreichen. Dabei versucht er, das Vertrauen seines Opfers zu erlangen. Zum einen durch freundliches und höfliches Auftreten und zum anderen durch das Feststellen und Betonen von Gemeinsamkeiten, sowohl in beruflichen als auch in privaten Angelegenheiten. Zu diesem Zeitpunkt des Angriffs macht sich eine gewissenhafte Vorbereitung des Angreifers bezahlt: Je besser dieser vorbereitet ist, desto mehr kann er ein Opfer blenden und einwickeln, um dadurch sein Vertrauen zu gewinnen. Eine Studie von Workman besagt, dass Menschen den Leuten vertrauen, die sie mögen und die Leute mögen, denen sie vertrauen [Wor07].

Meist sind Social-Engineering-Angriffe längerfristig angelegt. Auf diese Weise wird vorsichtig das Vertrauen und der persönliche Kontakt aufgebaut, um letztendlich zu dem Punkt zu kommen, in welchem das Vertrauen dann zum Erlangen der gewünschten Information ausgenutzt wird. Beim Social Engineering per Telefon dienen die ersten Anrufe beispielsweise oft nur dem Aufbau eines Vertrauensverhältnisses und der Social Engineer versucht noch nicht, seinem Ziel näher zu kommen und etwa schon nach wichtigen Informationen zu fragen. Wenn die Zeit für den eigentlichen Angriff gekommen ist, besteht beim Opfer bestenfalls kein Zweifel mehr an der Authentizität des Angreifers. Das Opfer vertraut der Person, für die sich der Angreifer ausgibt – fällt also auf das Rollenspiel des Angreifers herein [MS06].

Sympathie ist dabei der Schlüssel zum Erfolg. Keiner hilft gern seinem Gegenüber, wenn dieser unsympathisch wirkt. Dabei ist es hilfreich, Gemeinsamkeiten zu schaffen, z. B. mit dem Argument, dass der Chef verärgert sein wird, wenn eine Sache nicht rechtzeitig erledigt bzw. abgearbeitet wurde. Der Social Engineer macht sich zunutze,

[5] Es sei angemerkt, dass z. B. Vertrauen und Hilfsbereitschaft von den Autoren losgelöst vom Thema dieses Abschnitts nicht als Schwäche angesehen werden, sondern vielmehr als starke und sehr positive Eigenschaften. Lediglich die leichte Ausnutzbarkeit dieser Charakterzüge macht sie in Zusammenhang mit Social Engineering zu einer „Schwäche“.

dass die meisten Arbeitnehmer in dieser Situation wahrscheinlich schon einmal gewesen sind und diesen Druck nachempfinden können. Der Social Engineer schlüpft in eine Opferrolle und hofft auf das Mitgefühl und die Hilfe der Kollegen, da er wie ein Opfer seines Chefs erscheint. Dabei stellt die Stimme einen entscheidenden Faktor für den Erfolg des Angriffs dar und kann als ein die Rolle unterstützendes Werkzeug eingesetzt werden. Mit einer freundlichen Stimme erreicht man viel eher die Sympathie eines Opfers, wohingegen eine leicht zittrige und unbeholfene Stimmlage das Mitgefühl des Opfers hervorrufen kann [MS06].

Noch wichtiger ist die Stimme beim persönlichen Auftreten eines Social Engineer. Dabei kommt es nicht nur auf Freundlichkeit an, sondern auf das gesamte Erscheinungsbild. Ein gepflegter, ordentlicher Auftritt im Anzug kann so manche Türen öffnen [Ano97]. Der Angreifer wird dabei wirken, als wäre seine Anwesenheit völlig selbstverständlich, um keinen Verdacht zu erregen. Für gewöhnlich ist er zudem geübt im Umgang mit fremden Personen [MS06].

2.3.5.2. Konformität

Der gesellschaftliche Hang zur Konformität ist ebenfalls ein Erfolgsgrund des Social Engineering [Gra01, MS06]. Darunter versteht man das Streben der Menschen nach Übereinstimmung mit gesellschaftlichen Werten und Normen. Angenommen, der Angreifer tritt beispielsweise als verzweifelter Kollege am Telefon auf. Er behauptet, er habe ein Problem mit seinem Computer und sein Chef würde wegen eines Projekts Druck machen, welches er nun aufgrund der Computerprobleme nicht mehr bearbeiten kann. Oft wird das Opfer in solch einer Situation bereitwillig versuchen zu helfen, da dies in der Natur des Menschen liegt [KK06a]. Dieses Mitgefühl wird zu einer Gefahr, denn es lässt sich leicht gegen die hilfsbereite Person verwenden. Wenn der Social Engineer gut vorbereitet ist, was er im Allgemeinen immer sein sollte, wird er eventuell noch den Namen eines Kollegen in das Gespräch einstreuen, der bereits schon erfolglos seine Hilfe angeboten hat. Das Opfer der Attacke wird dies zum Anlass nehmen, ebenso alles Mögliche zu unternehmen, da der genannte Kollege dies auch getan hat und er nicht als unkollegial dastehen möchte.

2.3.5.3. Autorität

Eine weitere Erklärung, warum Social Engineering zum Erfolg führen kann, ist das Denken in Hierarchien. Wenn es ein Angreifer schafft, als Vorgesetzter oder Person aus dessen Umfeld zu erscheinen, z. B. als Assistent der Geschäftsleitung, dann wird das Opfer bereitwilliger zum Helfen animiert sein [Wor07]. Zum einen liegt das daran, durch besondere Einsatzfreude einen guten Eindruck bei wichtigen Personen hinterlassen zu wollen, zum anderen möchte man nicht die Anfragen von Vorgesetzten zurückweisen. Der mangelnde Wille zur Zusammenarbeit und Hilfsbereitschaft könnte sich negativ auf die Karriere auswirken. Dieses Risiko wird der Angestellte nicht eingehen wollen und so eventuelle Sicherheitsbedenken bezüglich der Legitimität der Anfragen übergehen. Wenn es dem Social Engineer also gelingt, glaubwürdig als Vorgesetzter aufzutreten, was natürlich nur funktioniert, wenn das Opfer diesen nicht persönlich kennt, verbessern sich seine Erfolgsaussichten entscheidend.

2.3.5.4. Überlastung

Es existieren zwei Arten von menschlicher Überlastung, die sich ein Social Engineer zunutze machen kann: Zum einen gibt es die arbeitstechnische Überlastung eines Mitarbeiters einer Zielorganisation. Dieser ist so mit Arbeit überlastet und überfordert, dass er bedingt durch den Stress gar nicht registriert, dass er Opfer eines Angriffs wird. Oftmals ist es so, dass überarbeitete und gehetzte Mitarbeiter so sehr auf ihre Arbeit fokussiert sind, dass sie äußere Ereignisse und Reize nicht bewusst wahrnehmen. Sie neigen dann dazu, äußere Störfaktoren möglichst schnell loszuwerden, damit sie wieder konzentriert ihrer Arbeit nachgehen können. Ein Angreifer kann dies ausnutzen, um einem Opfer Informationen zu entlocken, die er von einem normal belasteten Mitarbeiter nicht erhalten würde. Durch den Stress bringt der Mitarbeiter nicht die notwendige Zeit auf, um Anfragen kritisch zu hinterfragen oder die Autorisierung der anfragenden Person zu überprüfen [Bur91]. Ein weiterer Vorteil des Social Engineer ist, dass sich ein gestresster Mitarbeiter häufig nach kurzer Zeit nicht mehr an die Anfrage erinnern kann und somit der Vorfall häufig unerkannt bleibt.

Zum anderen ist damit die kognitive Überlastung gemeint. Dabei versucht der Angreifer, das logische Denken seines Opfers mit Argumenten zu überfordern. Das Opfer wird durch zu viele Informationen in kürzester Zeit mental überfordert. Dies hat zur Folge, dass das Opfer Informationen nur noch aufnehmen, aber nicht mehr bewerten kann, und so gewillter ist, den Forderungen des Social Engineer nachzukommen [Gra02].

2.3.5.5. Beeinflussung

Bei der Beeinflussung nutzt der Angreifer einen hochemotionalen Zustand des Opfers aus, damit er diesem die gewünschten Informationen entlocken kann. Diese Emotionen, wie Überraschung, Wut oder Angst, schürt er bewusst und dienen der Ablenkung, da sie die logische Denkfähigkeit ebenso wie das Finden möglicher Gegenargumente behindern.

Beispielsweise kann der Angreifer seiner Zielperson von einem hohen Geldgewinn berichten, damit dieses starke, emotionale Freude verspürt. Ein anderes Beispiel für Beeinflussung ist es, wenn der Angreifer die Zielperson darüber informiert, dass sie aller Voraussicht nach bald mit der Kündigung rechnen muss. Dies wiederum ruft eher panische Gefühle in dem Opfer hervor [Gra02].

2.3.5.6. Der neue Angestellte

Der letzte hier vorgestellte psychologische Aspekt ist eher eine Kombination der vorangegangenen Aspekt, die typischer Weise in einer ganz bestimmten Person des Zielunternehmens vereinigt wird: Dem *neuen Angestellten* [MS06]. Da ist zum einen das angesprochene Vertrauen: Eine Person, die neu in ein Unternehmen eintritt, begegnet gewöhnlich nicht gleich von Beginn an mit Argwohn den neuen Kollegen und Kunden. Dieser Effekt ist insbesondere dann festzustellen, wenn der neue Angestellte noch keine Erfahrung mit Social-Engineering-Angriffen gemacht hat – also vor allem bei Berufseinsteigern. Dies gilt allgemein (also nicht nur für Social-Engineering-

Angriffe) für die meisten Menschen, da es sich gezeigt hat, dass Personen, die bereits einmal Opfer von Betrügern geworden sind, in kommenden, vergleichbaren Situationen weitaus argwöhnischer sind als Personen, denen noch nichts in dieser Art zugestoßen ist [Wor07].

Der Wunsch nach Konformität lässt sich bei neuen Angestellten ebenfalls einfacher ausnutzen. Wenn einer der neuen, wahrscheinlich noch unbekannten Kollegen um Hilfe bittet, wird sich der neue Angestellte nicht weigern, dem Wunsch zu entsprechen. Gegenteiliges Handeln wirkt generell unfreundlich und kann sich längerfristig nachteilig auf das Arbeitsklima auswirken. Der neue Angestellte ist eher darauf bedacht einen guten ersten Eindruck zu hinterlassen und sich gut in das neue Arbeitsumfeld zu integrieren. Also kann der Social Engineer davon ausgehen, beim einem neuen Angestellten auf ein leichtes Opfer für einen Angriff zu treffen.

Ähnlich verhält es sich, wenn sich ein vermeintlicher Vorgesetzter meldet. Den direkten Chef wird der neue Angestellte eventuell schon kennen, aber eben nicht alle Vorgesetzten. Falls er diese doch kennt, dann vielleicht aber noch nicht deren Stimme am Telefon. Das macht es leicht für den Angreifer, sich für jemand anderes auszugeben und bei dem neuen Angestellten ist die Wahrscheinlichkeit geringer, dass dieser die gespielte Autorität bezweifelt oder in Frage stellt.

2.4. Zusammenfassung

Dieses Kapitel präsentiert die Grundlagen, die zum weiteren Verständnis der Arbeit notwendig sind. Da der Fokus der vorliegenden Arbeit auf einer speziellen Art von Schadprogrammen liegt, widmet sich der erste Abschnitt dieses Grundlagenkapitels dem Begriff *Malware*. Dies sind Schadprogramme, die eine vom Benutzer unerwünschte Funktionalität besitzen. Sie lassen sich hinsichtlich verschiedenster Kriterien, wie etwa deren Verbreitungsroutine oder Funktionsweise, in unterschiedliche Klassen aufteilen. In diesem Grundlagenkapitel werden die sechs Klassen *Viren*, *Würmer*, *Bots*, *Trojanische Pferde*, *Spyware* und *Rootkits* näher vorgestellt und charakterisiert.

In den folgenden Kapiteln spielt speziell die Verbreitungsroutine eines Schadprogramms eine wesentliche Rolle, so dass die Art und Weise, wie sich ein Schadprogramm verbreiten kann, Gegenstand des zweiten Abschnitts dieses Kapitels ist. Neben einer vollkommen autonomen Verbreitung durch das *Ausnutzen von Sicherheitslücken in angebotenen Diensten* werden auch Verbreitungsmöglichkeiten vorgestellt, bei denen ein potentielles Opfer aktiv in den Verbreitungsprozess des Schadprogramms involviert ist: Verbreitung über das *World Wide Web*, *E-Mails*, *Instant Messanging*, *Soziale Netzwerke*, *Filesharing*, *Netzfreigaben* oder auch über *Wechseldatenträger*. Bei den zuletzt genannten Verbreitungsroutinen ist jeweils eine Benutzerinteraktion für eine erfolgreiche Infektion eines neuen Systems notwendig, zum Beispiel das Folgen eines Verweises, das Herunterladen und Ausführen eines Dateianhangs oder auch das Einlegen eines Datenträgers. Das Ausnutzen von Sicherheitslücken erfordert solch eine Benutzerinteraktion in der Regel nicht.

Damit ein potentielles Opfer auch tatsächlich diese Interaktion durchführt, versucht ein Angreifer ein Opfer durch Täuschung und Manipulation dazu zu bewegen. Diesen Vorgang bezeichnet man als *Social Engineering* und er ist Thema des dritten Ab-

schnitts dieses Kapitels. Nach der Vorstellung einiger geläufiger Definitionen werden als Erstes die möglichen Motive eines Social Engineer diskutiert. Neben einem reinen finanziellen Interesse des Angreifers können dies auch ein persönliches Selbstinteresse, z. B. bedingt durch Eifersucht oder Rache sein. All jene Dinge, die diese Motive des Angreifers befriedigen, sind dessen Ziele. Bei den Zielen unterscheidet man zwischen Hauptzielen und Zwischenzielen. Letztere helfen dem Angreifer seine Hauptziele zu erreichen, die wiederum seine Motive befriedigen. Zum Erreichen der Ziele stehen einem Angreifer viele verschiedene Methoden zur Verfügung, wie etwa das *Dumpster Diving*, bei dem der Abfall der Zielperson/-organisation nach hilfreichen Dingen durchsucht wird, das *Shoulder Surfing*, bei dem heimlich vertrauliche Informationen mitgelesen werden, oder auch *Phishing*, das ein Versuch ist, über gefälschte Internetseiten an vertrauliche Daten zu gelangen. Abschließend liefert das Kapitel noch einige psychologische Erklärungsversuche, warum Social Engineering erfolgreich sein kann. Ansatzpunkte für Social-Engineering-Angriffe sind menschliche „Schwächen“, wie etwa grundsätzliches *Vertrauen* in andere Menschen, die *Autorität* in Verbindung mit *Angst*/*Pflichtgefühl* oder aber ein allgemeines Verlangen nach *Konformität* im täglichen Leben.

KAPITEL III

Social Malcode

In diesem Kapitel wird der für diese Dissertation zentrale Begriff des Social Malcode weiter präzisiert und anhand einiger Beispiele für diese Art von Schadprogrammen verdeutlicht. Des Weiteren wird die Abgrenzung zu anderen Schadprogrammen an Gegenbeispielen und einer Erklärung, warum es sich bei diesen Gegenbeispielen nicht um Social Malcode handelt, herausgearbeitet. Nach einer anschließenden Diskussion, welche die Schwierigkeiten bei der Definition der Begrifflichkeiten verdeutlicht, wird abschließend die endgültige Definition von Social Malcode präsentiert.

3.1. Erste Definition

In Abschnitt 1.2 wurde der Begriff *Social Malcode* vorläufig als die Menge aller Schadprogramme beschrieben, die für ihre Verbreitung auf Social Engineering angewiesen sind. Ein Problem dieser ersten Beschreibung von Social Malcode liegt in der Verwendung des Begriffs *Social Engineering*. Wie bereits in Abschnitt 2.3 herausgestellt, spielt die Intention einer Person eine wesentliche Rolle dabei, ob eine Handlung der Person als Social Engineering zu betrachten ist oder nicht.

Verschickt eine Person beispielsweise unwissentlich ein Schadprogramm per E-Mail ist diese Aktion nicht als Social Engineering zu bezeichnen, da die Person nicht wusste, dass von dem verschickten Programm eine Gefahr ausgeht. Weiß die Person dies hingegen und verschickt es mit dem Ziel, den Rechner des Empfängers zu infizieren und um dadurch möglicherweise Zugriff auf sensible Daten zu erlangen, ist diese Handlung jedoch sehr wohl als Social Engineering zu bezeichnen.

Da es im Allgemeinen schwierig ist, objektiv die Absicht einer Person bezüglich einer Handlung festzustellen beziehungsweise zu messen, wird in diesem Abschnitt eine formalere Definition von Social Malcode herausgearbeitet, die dieses Problem behebt. Um solch eine Definition herzuleiten, werden zuerst verschiedene andere Begriffe definiert, auf denen dann wiederum die Definition von Social Malcode aufbaut.

Definition 3.1: Ressource
Eine *Ressource* ist ein Betriebsmittel eines Computers oder eine Information eines Systembenutzers.

Ein Betriebsmittel eines Computers entspricht der Hardware eines Computers, oder zumindest einem Teil der Hardware, und der durch diese Hardware zur Verfügung gestellten Leistung. Beispiele für Betriebsmittel sind die Festplatte, die CPU, der Hauptspeicher, die Netzkarte, aber auch die durch diese Hardware erbrachte Leistung: Speicherplatz (durch die Festplatte und den Hauptspeicher zur Verfügung gestellte Leistung), Rechenzeit (durch die CPU erbrachte Leistung) oder der Netzverkehr (Leistung der Netzkarte).

Die Informationen eines Systembenutzers sind zum einen Informationen, die abstrahiert durch Dateien und Datenobjekte auf dem Computer abgespeichert sind, und zum anderen Informationen, die für die Benutzung des Computers oder eines externen Dienstes erforderlich sind. Beispiele für die zweite Kategorie von Informationen sind etwa das Benutzerpasswort zur Anmeldung am Betriebssystem des Computers (dieses ist in der Regel nicht im Klartext auf dem Computer abgespeichert) oder auch die Zugangsdaten zu Online-Diensten, wie zum Beispiel für das Online-Banking oder für Online-Auktionshäuser.

Definition 3.2: Code
Code ist ein Text in einer formalen Sprache, wie etwa einer Programmier-/Skriptsprache oder einer Auszeichnungssprache.

Der Quelltext eines Programmes entspricht dieser Definition, da dieser in Textform notiert wird. Handelt es sich um einen Quelltext in einer Programmiersprache, wie zum Beispiel *C*, *C++* oder *Pascal*, wird der Quelltext von einem Compiler in ein lauffähiges Programm übersetzt. Bei einer Skriptsprache, wie etwa *Python*, *Perl* oder *PHP*, wird der Quelltext zur Laufzeit durch einen sogenannten Interpreter interpretiert, der die Anweisungen Schritt für Schritt ausführt. Beide Arten von Quelltext haben gemeinsam, dass diese in textueller Form und in einer formalen Sprache verfasst werden. Nach Definition 3.2 handelt es sich also bei beiden Arten um Code.

Neben dem Quelltext eines Programms entspricht auch insbesondere Shellcode der obigen Definition. Als Malware hingegen kann Shellcode streng genommen nicht bezeichnet werden, da ein Betriebssystem diesen nicht isoliert von anderen Anwendungen ausführen kann – ein Shellcode ist also keine eigenständige Software beziehungsweise Anwendung. Häufig wird Shellcode von einem Angreifer als Payload eines Exploit verwendet. Sinn und Zweck des Shellcodes ist es dann, einem Angreifer auf dem kompromittierten Rechner eine Shell zur Verfügung zu stellen, damit dieser dort weitere Befehle und Aktionen ausführen kann. Dafür wird der Shellcode nach dem Ausnutzen einer Schwachstelle in einen verwundbaren Prozess injiziert. Verfasst wird Shellcode üblicherweise in Maschinencode. Die Definition von Code trifft also auf Shellcode zu [Blo04, Sym06].

Als Auszeichnungssprache bezeichnet man die textuelle Beschreibung eines Dokumentenformats. Teilweise enthalten die Dokumentbeschreibungen noch Informationen darüber, wie einzelne Teile des Dokuments bei der Verarbeitung behandelt werden

sollen. Beispiele für eine Auszeichnungssprache sind *HTML* (zur Beschreibung einer Internetseite), TEX (zur Beschreibung von formatierten Texten – häufig verwendet bei wissenschaftlichen Texten mit mathematischen Formeln) oder *XML* (zur Darstellung von hierarchisch strukturierten Daten in Form von Textdaten).

Definition 3.3: Nicht-autorisierter Zugriff (in Anlehnung an [Eck07])
Ein *nicht-autorisierter Zugriff* ist ein Zugriff auf Ressourcen eines Benutzers durch einen anderen Benutzer, der eigentlich keine Zugriffsrechte für diese Ressourcen besitzt.

Jede Ressource, egal ob Betriebsmittel oder Information eines Benutzers, gehört einem bestimmten Benutzer. So gehört die Hardware eines Computers etwa dem Käufer des Computers und eine Datei in der Regel dem Benutzer, der diese Datei angelegt hat. Der Besitzer einer Ressource kann weiteren Benutzern Zugriffsrechte an der Ressource gewähren. Ein Besitzer eines Computers kann beispielsweise den Angestellten seines Betriebs oder seinen Familienmitgliedern die Benutzung des Computers gestatten. Ein Besitzer einer Datei kann die Zugriffsrechte auf die Datei so festlegen, dass andere Benutzer des Computers die Datei ebenfalls lesen, editieren oder ausführen können. Greift jedoch ein anderer Benutzer, der keine solche Zugriffsrechte vom Besitzer der Ressource erhalten hat, auf die Ressource zu, wird dieser Zugriff laut Definition 3.3 als nicht-autorisiert bezeichnet.

Definition 3.4: Benutzerinteraktion
Als eine *Benutzerinteraktion* wird eine von einem Benutzer ausgehende Kommunikation zwischen dem Benutzer und einem Computerprogramm bezeichnet.

Entscheidend bei der Kommunikation in Definition 3.4 ist deren Richtung: Diese ist laut Definition vom Benutzer zum Computer gerichtet und nicht umgekehrt. Die hier definierte Benutzerinteraktion entspricht somit einer Eingabe des Benutzers. Dies kann beispielsweise eine Eingabe per Tastatur sein, wie sie etwa zur Authentifizierung beim Online-Banking per Kontonummer und PIN notwendig ist, oder eine Eingabe mit der Maus. Ein Beispiel für eine Eingabe mit der Maus ist ein Doppelklick auf den Anhang einer E-Mail in einem E-Mail-Client, wodurch in der Regel das Herunterladen des Anhangs initiiert wird.

Im normalen Sprachgebrauch wird unter einer Benutzerinteraktion zusätzlich noch eine Ausgabe des Computers an den Benutzer verstanden. Diese Art der Interaktion ist in der obigen Definition jedoch explizit durch die geforderte Richtung ausgeschlossen.

Mit Hilfe der bisher in diesem Abschnitt gegebenen Definitionen wird Social Malcode nun folgendermaßen definiert:

Definition 3.5: Social Malcode
Social Malcode ist Code, der

(a) einem Angreifer nicht-autorisierten Zugriff auf eine oder mehrere Ressourcen eines Opfers ermöglicht und

(b) zur Weiterverbreitung eine Benutzerinteraktion des Opfers erfordert.

Dabei fordert die Teilbedingung (a) die schadhafte Funktionalität eines Codes, damit dieser als Social Malcode deklariert werden kann und die Teilbedingung (b) ersetzt die geforderte Verbreitung durch Social Engineering der vorläufigen Beschreibung aus Abschnitt 1.2. Die beiden Begriffe *Angreifer* und *Opfer* haben wir nicht näher formalisiert, da diese im allgemeinen Sprachgebrauch bekannt sind. Wir benutzen die beiden Begriffe so, wie sie intuitiv verstanden werden: Das Opfer einer Handlung ist die geschädigte Person und der Schaden wird von einem Angreifer ausgehend verursacht. Auch wird davon ausgegangen, dass Angreifer und Opfer zwei unterschiedliche Personen sind.

Durch diese Definition besteht nun nicht mehr das Problem, dass man die Intention einer Person feststellen und/oder beurteilen muss. Stattdessen ist nun für die Klassifikation als Social Malcode entscheidend, ob eine Benutzerinteraktion im Verbreitungsprozess eines Schadprogramms enthalten ist oder nicht. Verglichen mit der Beurteilung einer Intention ist die Frage, ob eine Benutzerinteraktion zur Verbreitung erforderlich ist, viel klarer zu fassen und zu beantworten.

3.2. Beispiele

Zur Verdeutlichung der Definition werden in diesem Abschnitt einige Beispiele von Social Malcode vorgestellt. Diese Beispiele lassen erkennen, wann ein Schadprogramm als Social Malcode bezeichnet werden kann – liefern also ein besseres Verständnis und Gefühl für die Definition.

3.2.1. Waledac

Waledac ist ein Bot, der erstmals im April 2008 gesichtet wurde, und dessen Botnetz eine hierarchische Struktur aufweist. Mittlerweile ist bekannt, dass diese Hierarchie aus vier Schichten besteht: Zwei obere Schichten, die von dem Botnetz-Betreiber betrieben werden und zwei untenliegende Schichten, die aus den Rechnern der Waledac-Opfer bestehen. Die beiden unteren Schichten bilden ein Peer-to-Peer-Netz [Cam09]. Dabei dienen die Rechner der untersten Schicht hauptsächlich zur Verbreitung von Spam. Die Rechner der zweiten Schicht von unten leiten die Anfragen der Rechner der untersten Schicht an die darüberliegende Schicht weiter und verschleiern so die beiden vom Botnetz-Betreiber verwalteten Schichten darüber.

Neben dem Versand von Spam können die Waledac-Bots einen befallenen Rechner auch nach Informationen durchsuchen, die für den Betreiber des Botnetzes interessant sein könnten, wie etwa E-Mail-Adressen oder Zugangsdaten. Zudem können sie Dateien aus dem Internet herunterladen und ausführen. Diese Funktionalität wird unter anderem auch für das Aktualisieren des Bots genutzt. Auch die Möglichkeit eines DoS-Angriffs ist in den Instanzen von Waledac integriert, jedoch konnte noch kein solcher Angriff beobachtet werden. Um die Verfügbarkeit der Internetseiten hinter den Verweisen der durch Waledac verschickten Spam-Mails zu erhöhen, befinden sich die-

se Seiten in einem sogenannten *Fast-Flux-Netz*. Dies ist eine Technologie, die den *Domain Name System* (DNS) Dienst missbraucht, um die Verfügbarkeit eines angebotenen Online-Dienstes zu erhöhen. Erreicht wird dies durch eine kurze Gültigkeit der DNS-Einträge, die somit in kurzen Abständen auf immer andere Computer des Botnetzes auflösen. Wird solch ein Computer des Botnetzes ausgeschaltet, löst der Name der Internetseite kurze Zeit später auf einen anderen kompromittierten Rechner auf. Gepaart mit mehreren, rotierenden Einträgen, ist die Internetseite somit immer erreichbar [HGRF08, Hon07].

Neben dem Werben für Produkte dienen die verschickten Spam-Mails auch der Verbreitung von Waledac. Die Infektion kann dabei auf zwei Arten stattfinden: Zum einen durch einen normalen (von einem Benutzer bewusst initiierten) Download und zum anderen durch einen Drive-By-Download Exploit. Bei der zweiten Variante reicht der reine Besuch einer Internetseite aus, damit sich der Benutzer mit Waledac infiziert. Sollte dies nicht funktionieren, besteht immer noch die Möglichkeit, dass der Besucher der Internetseite durch Social Engineering Tricks dazu gebracht werden kann, den Bot manuell herunterzuladen und zu installieren. Bei beiden Varianten werden die potentiellen Opfer über die von Waledac verschickten E-Mails auf die Internetseiten gelockt – beispielsweise indem ihnen mitgeteilt wird, dass auf diesen Seiten Grußkarten für sie hinterlegt sind oder sich dort wichtige und interessante Nachrichtenmeldungen befinden. Um sich möglichst effektiv zu verbreiten, sind die Mails inhaltlich meist an aktuelle Zeitgeschehnisse (etwa die Wahl von Barack Obama zum US-Präsidenten) oder Ereignisse (beispielsweise der Valentinstag oder der NFL Super Bowl) angelehnt und werden als Kampagnen bezeichnet. Abbildung 3.1 zeigt die Internetseite einer Unabhängigkeitstag-Kampagne, auf der vermeintlich ein Video eines Feuerwerks zu sehen ist. Möchte sich ein Benutzer dieses Video anschauen, bekommt er die Datei *video.exe* zum Herunterladen angeboten. Diese Datei enthält jedoch statt des Videos eine Instanz von Waledac.

Unabhängig von der verwendeten Art des Social Engineering, muss der Benutzer jedoch per Benutzerinteraktion dem Verweis in der E-Mail folgen, damit sich Waledac verbreiten kann, so dass Punkt (b) der Definition 3.5 erfüllt ist [BCF09, SGE+09].

3.2.2. Sturm-Wurm

Der *Sturm-Wurm* wurde erstmals im Januar 2007 gesichtet und gilt als der vermeintliche Vorgänger von Waledac, da Waledac das erste Mal über Instanzen des Sturm-Wurms heruntergeladen wurde. Die Funktionalität des Sturm-Wurms und dessen Aufbau ist dem von Waledac sehr ähnlich, nur dass es hier drei Hierarchie-Ebenen gibt. Die Kommunikation zwischen der oberen und der mittleren Ebene entspricht einem Client-Server-Modell und die Netzstruktur zwischen der mittleren und der unteren Ebene entspricht einem Peer-to-Peer-Netz.

Auch die Verbreitung des Sturm-Wurms ähnelt der Verbreitung des Waledac-Bots: Von bereits infizierten Rechnern werden E-Mails versandt, die neue potentielle Opfer durch Social Engineering Tricks auf Internetseiten locken sollen. Beim Besuch dieser Seiten wird, einen verwundbaren Browser vorausgesetzt, ein Drive-By-Download Exploit ausgeführt. Sollte dies nicht funktionieren, besteht immer noch die Möglichkeit, dass der Benutzer dazu gebracht wird, selbst das Herunterladen des Bots zu

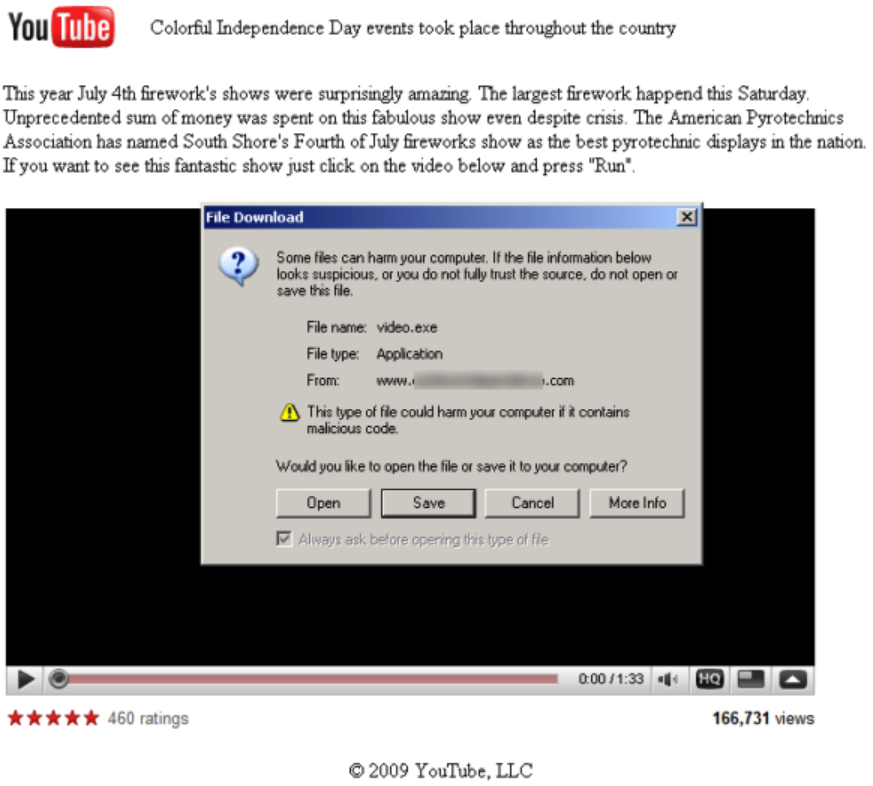


Abbildung 3.1.: Waledac-Kampagne zum Unabhängigkeitstag
Diese Internetseite dient zur Verbreitung von Waledac. In der Kampagne soll das Interesse an einem Feuerwerks-Video den Benutzer dazu bringen, eine ausführbare Datei herunterzuladen und zu starten [WSL09].

starten. Dazu werden dem Seitenbesucher Dateien angeboten, wie beispielsweise eine Grußkarte oder einen Video-Codec, die in Wirklichkeit jedoch eine Instanz des Sturm-Wurms sind. Im Gegensatz zu Waledac gab es beim Sturm-Wurm noch eine dritte Variante der Verbreitung: Am Anfang wurde Sturm-Wurm ausschließlich über E-Mails verbreitet, die eine Instanz des Bots als Dateianhang enthielten. Die E-Mails waren so gestaltet, dass der Empfänger dazu verleitet werden sollte, diesen Anhang herunterzuladen und auszuführen. Auch diese Art der Verbreitung erfordert für eine erfolgreiche Infektion eines Systems wieder Benutzerinteraktion: Der Anhang muss vom Empfänger auf der Festplatte abgespeichert und anschließend ausgeführt werden [HSD+08, Dah08].

3.2.3. ILOVEYOU

Im Mai 2000 wurde zum ersten Mal der Wurm *ILOVEYOU* beobachtet. Dieses auch unter dem Namen *LoveLetter* bekannte Schadprogramm veränderte hauptsächlich Bild- und Musikdateien der befallenen Rechner und hat die auf dem Rechner gespeicherten Passwörter ausgelesen. Der Wurm verbreitete sich extrem schnell und hat nach nur wenigen Stunden bereits weltweit Rechner infiziert. Neben zahlreichen

Wirtschaftsunternehmen wurden auch etliche Computer von Regierungsorganisationen infiziert, wie zum Beispiel dem britischen Parlament oder dem amerikanischen Kongress. Schätzungen gehen davon aus, dass innerhalb der ersten Woche, in der der Wurm freigesetzt wurde, bereits 45 Millionen Computer mit dem Schädling infiziert waren.

Wesentlichen Anteil an der schnellen Verbreitung des ILOVEYOU-Wurms hatte dessen auf Social Engineering beruhende Verbreitungsroutine, die den Wurm per E-Mail propagierte. Die E-Mails hatten stets die Betreffzeile „ILOVEYOU“ und enthielten einen Dateianhang mit dem Namen „LOVE-LETTER-FOR-YOU.TXT.vbs“. Die Standardeinstellungen der damaligen Windows-Betriebssysteme verhinderten, dass die Dateierweiterung .vbs den Empfängern angezeigt wurde. Diese gingen davon aus, dass es sich bei der Datei um eine harmlose Textdatei handelt und erkannten nicht, dass der Anhang ein Visual Basic Skript war. Neben dem Betreff und dem Namen des Dateianhangs sollte auch der Inhalt der E-Mail, dies war eine explizite Aufforderung zum Öffnen des Anhangs, den Empfänger dazu bringen, den Dateianhang zu öffnen. Sobald der Empfänger diesen geöffnet hatte (was wiederum eine Benutzerinteraktion ist), schickte das Visual Basic Skript die gleiche E-Mail an jeden Eintrag des Windows-Adressbuchs, jedoch mit der Absenderadresse des Empfängers der ursprünglichen E-Mail, um die Glaubwürdigkeit des vermeintlichen Liebesbriefes weiter zu erhöhen [Bis00, PP01].

3.2.4. Mydoom

In Erscheinung getreten ist *Mydoom* zum ersten Mal im Januar 2004. Mydoom ist ein E-Mail-Wurm, der nach seiner Installation den ersten freien Port zwischen 3127 bis 3198 öffnet. Verbindet sich ein Angreifer über einen dieser Ports zu einem kompromittierten Computer, kann er diesen als TCP-Proxy verwenden oder beliebige Dateien auf den Computer hochladen und ausführen. Des Weiteren sollte Mydoom zwischen dem 1. und 12. Februar 2004 einen DDoS-Angriff auf die Verfügbarkeit der Internetseite der US-amerikanischen Firma *The SCO Group, Inc.* [SCO10] ausführen.

Auch die Verbreitungsroutinen von Mydoom sind für eine erfolgreiche Verbreitung auf die Interaktionen eines Benutzers angewiesen. Mydoom verfügt über zwei separate Verbreitungsroutinen:

1. **E-Mail**: Nach der Infektion eines Rechners durchsucht der Schädling Dateien mit bestimmten Dateiendungen nach E-Mail-Adressen und liest das Windows-Adressbuch aus. Anschließend werden E-Mails an die gefundenen E-Mail-Adressen verschickt, die eine Instanz des Wurms als Dateianhang enthalten. Inhalt der E-Mails war ein Hinweis, dass angeblich eine andere Nachricht nur fehlerhaft zum Empfänger übermittelt werden konnte oder dass eine andere E-Mail aufgrund eines falschen Encoding nicht angezeigt werden kann. Laut E-Mail-Text ist der Anhang der Teil der anderen Nachricht, der keine Fehler enthält oder die Nachricht in einem binären Format, welches angeblich das fehlerhafte Encoding umgeht. Auf diese Weise soll der Empfänger dazu gebracht werden, den Dateianhang herunterzuladen und auszuführen. Die Betreffzeile der E-Mails ist inhaltlich in gleicher Weise auf diesen Trick abgestimmt: Etwa „Mail Tran-

saction Failed“ oder „Server Report“. Das Herunterladen des Anhangs ist wiederum eine Benutzerinteraktion und entspricht somit der Definition von Social Malcode.

2. **Internet-Tauschbörse**: Die zweite Verbreitungsroutine durchsucht nach der Infektion eines Rechners dessen Windows-Registrierungsdatenbank. Sollte dort ein Eintrag für den freigegebenen Ordner der Internet-Tauschbörse *Kazaa* gefunden werden, so kopiert sich Mydoom unter einem unauffälligen Namen in diesen Ordner. Beispiele für die Dateinamen sind etwa *activation_crack* oder *strip-girl-2.0bdcom_patches*. Damit sich Mydoom über die Tauschbörse Kazaa verbreitet, sind ebenfalls mehrere Benutzerinteraktionen erforderlich: Ein Benutzer muss den Tauschbörsen-Client starten, manuell nach einer der Dateien suchen, als die sich Mydoom tarnt, und diese nach dem vollständigen Herunterladen auch ausführen.

Der Wurm wurde so implementiert, dass mit dem Ende des DDoS-Angriffs ab dem 12. Februar 2004 auch die aktive Verbreitung von Mydoom beendet wurde. Wird eine Instanz des Wurms nach diesem Datum zur Ausführung gebracht, werden keine E-Mails mehr versendet und der Wurm kopiert sich auch nicht mehr in den freigegebenen Ordner der Tauschbörse Kazaa [FC10, Sym10].

3.2.5. Mal/EncPK-LL

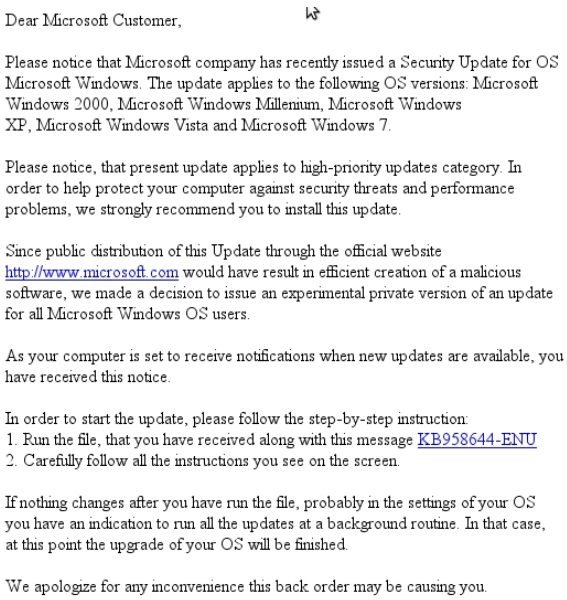
Neben E-Mails mit einem Verweis auf eine Internetseite oder einem Dateianhang kann ein Schadprogramm auch über E-Mails verbreitet werden, die einen Verweis enthalten, der direkt auf das Schadprogramm selbst zeigt. Zum Beispiel hat im Dezember 2009 das IT-Sicherheitsunternehmen *Sophos* einen Schädling untersucht, den sie als *Mal/EncPK-LL* bezeichnen. Dies ist ein in *Delphi* geschriebener Schädling, der mit einem Standardpacker komprimiert ist.

Abbildung 3.2 zeigt ein Beispiel einer E-Mail, über die sich der Schädling Mal/EncPK-LL verbreitete. Dabei handelte es sich um eine gefälschte Nachricht von Steve Lipner, der bei *Microsoft* mitverantwortlich für die Entwicklung sicherer Software ist (Director of Security Assurance Microsoft Corporation). In dieser E-Mail wird der Empfänger dazu aufgefordert, eine Datei aus dem Internet herunterzuladen, auf die ein Verweis innerhalb der E-Mail zeigt. Diese Datei ist angeblich ein Sicherheits-Update für das Microsoft Windows-Betriebssystem, tatsächlich handelt es sich jedoch um den Schädling Mal/EncPK-LL.

Bemerkenswert ist, dass die abgebildete E-Mail im Dezember 2009 verschickt wurde, der Support für das Betriebssystem *Microsoft Millenium* jedoch bereits 2006 eingestellt wurde. Davon abgesehen hat *Microsoft* immer wieder darauf aufmerksam gemacht, dass sie keine Updates per E-Mail verbreiten. Dennoch wurden immer wieder Infektionen mit dem hier beschriebenen Schädling bekannt [Sva09, Con09].

3.3. Gegenbeispiele

In diesem Abschnitt werden drei bekannte Gegenbeispiele für Social Malcode präsentiert, also Schadprogramme, die kein Social Malcode sind. Bei jedem hier vorgestell-



Dear Microsoft Customer,

Please notice that Microsoft company has recently issued a Security Update for OS Microsoft Windows. The update applies to the following OS versions: Microsoft Windows 2000, Microsoft Windows Millenium, Microsoft Windows XP, Microsoft Windows Vista and Microsoft Windows 7.

Please notice, that present update applies to high-priority updates category. In order to help protect your computer against security threats and performance problems, we strongly recommend you to install this update.

Since public distribution of this Update through the official website http://www.microsoft.com would have result in efficient creation of a malicious software, we made a decision to issue an experimental private version of an update for all Microsoft Windows OS users.

As your computer is set to receive notifications when new updates are available, you have received this notice.

In order to start the update, please follow the step-by-step instruction:
1. Run the file, that you have received along with this message KB958644-ENU
2. Carefully follow all the instructions you see on the screen.

If nothing changes after you have run the file, probably in the settings of your OS you have an indication to run all the updates at a background routine. In that case, at this point the upgrade of your OS will be finished.

We apologize for any inconvenience this back order may be causing you.

Abbildung 3.2.: Gefälschte E-Mail mit einem Verweis auf eine ausführbare Datei
Über diese vermeintlich von *Microsoft* stammende E-Mail verbreitete sich der Schädling Mal/EncPK-LL. Der Empfänger wird darauf hingewiesen, dass ein neues Update für Windows-Betriebssysteme zur Verfügung steht, auf das ein Verweis in der E-Mail zeigt [Sva09].

ten Schadprogramm wird kurz dessen Schadroutine beschrieben und herausgestellt, wie sich dieses verbreitet. Dadurch wiederum wird die Abgrenzung zwischen Social Malcode und anderen Schadprogrammarten deutlicher.

3.3.1. Morris-Wurm

Im Jahr 1988 schrieb der damalige Student Robert Tappan Morris den mutmaßlich ersten Computerwurm. Dieser befiel ab November 1988 viele Rechner im Internet und wurde nach seinem Autor benannt: der *Morris-Wurm*. Der Wurm besaß keine eigentliche Schadroutine, sondern war als Experiment gedacht, um zu überprüfen, wie schnell sich selbstreplizierender Code verbreiten kann. Durch einige Designfehler geriet dieses Experiment jedoch außer Kontrolle und der Wurm infizierte viel mehr Rechner, als Morris anfangs angenommen hatte: Morris implementierte den Wurm zwar so, dass dieser einen Rechner nicht mehrfach infizieren sollte, jedoch funktionierte diese Beschränkung nicht wie gewünscht. Aufgrund von *Race Conditions* [Whe03] kam es immer häufiger zu einer mehrfachen Kompromittierung ein und derselben Rechner. Auch die Verfügbarkeit der Rechner, die eigentliche nicht anfällig für den Morris-Wurm waren, wurde durch die hohe Last beeinträchtigt, die durch die Kompromittierungsversuche bereits infizierter Rechner entstand [Orm03].

Der Morris-Wurm besaß drei Verbreitungsroutinen, in denen jeweils eine bestimmte Anwendung zur Verbreitung beitrug. Bei zwei Anwendungen nutzte er eine Schwachstelle aus und bei der dritten Anwendung machte er sich eine schlechte Konfiguration der Anwendung zunutze [Orm03]:

1. **Unix sendmail**: Laut den Entwicklern des Mail Transfer Agent *sendmail* nutzte der Morris-Wurm zwei Eigenschaften/Funktionen von sendmail aus, die einzeln betrachtet keine Schwachstelle aufwiesen, deren Kombination aber anfällig für eine Remote Code Ausführung waren. Dies war zum einen der *DEBUG* Modus von sendmail und zum anderen eine Funktion zur Ausführung eines beliebigen Kommandos, das für Remote-Benutzer aber nur zur Verfügung stand, wenn der DEBUG Modus aktiviert war. Zur damaligen Zeit hatten viele sendmail-Instanzen diesen Modus unglücklicherweise aktiviert, so dass der Morris-Wurm dies ausnutzte, um entsprechende Rechner zu infizieren.
2. **fingerd**: Mit Hilfe der Server-Applikation des Finger-Protokolls *fingerd* (finger deamon) lassen sich hinterlegte Informationen über die Benutzer des Servers abfragen. Dazu gehören beispielsweise der Benutzername, der reale Name des Benutzers oder der Anmeldezeitpunkt. Der Morris-Wurm nutzte einen Buffer-Overflow (Verwendung der Funktion `gets`) in dieser Anwendung aus, um eine Remote-Shell zu starten. Über diese bekam der Wurm Zugriff auf die kompromittierten Rechner und konnte sich so weiter verbreiten.
3. **rsh/rexec**: Mittels der beiden Dienste *rsh* (remote shell) und *rexec* (remote process execution) ist es einem Benutzer möglich, nicht-interaktive Anwendungen auf einem entfernten Rechner auszuführen. Diese Dienste können so konfiguriert werden, dass in Konfigurationsdateien die Benutzer und Rechner angegeben werden, die sich über die beiden Dienste verbinden dürfen. Der Morris-Wurm machte sich zunutze, dass die meisten Benutzer, die auf mehreren Computern arbeiteten, auf den verschiedenen Rechnern das gleiche Passwort verwendeten. Auf diese Weise konnte sich der Wurm auch auf Computer kopieren, auf denen nicht sendmail oder fingerd lief.

Jede der drei beschriebenen Verbreitungsroutinen läuft vollständig autonom ab, d. h. es ist kein Eingreifen eines Benutzers erforderlich. Somit ist auch keine vom Morris-Wurm verwendete Verbreitungsroutine auf eine Benutzerinteraktion angewiesen. Daraus folgt, dass Teil (b) der Definition von Social Malcode nicht erfüllt ist und somit der Morris-Wurm keinen Social Malcode darstellt.

3.3.2. Code Red

Das nächste Beispiel eines Schädlings, der nicht als Social Malcode bezeichnet werden kann, ist der Computerwurm *Code Red*. Dieser wurde das erste Mal im Juli 2001 gesichtet und seine Schadroutine bestand darin, DDoS-Angriffe gegen verschiedene IP-Adressen durchzuführen – unter anderem gegen die IP-Adresse eines Webservers des Weißen Hauses. Neben diesem DDoS-Angriff verunstaltete Code Red die Internetseiten, die sich auf einem befallenen Rechner befanden. So wurde den Index-/Hauptseiten der Internetauftritte der Spruch *Hacked By Chinese* hinzugefügt. Im August des Jahres 2001 wurde eine neue Variante des Code Red entdeckt. Diese zweite Version veränderte keine Internetseiten mehr und führte auch keinen DDoS-Angriff mehr aus, sondern öffnete lediglich eine Hintertür, über die ein kompromittierter Rechner anschließend ferngesteuert werden konnte.

Code Red verbreitete sich über die Schwachstelle CVE-2001-0500 [CVE01] der Webserver-Anwendung *Internet Information Services* (IIS) von *Microsoft*. Dabei wur-

de ein Buffer-Overflow im Index-Server ausgenutzt, so dass beliebiger Code zur Ausführung gebracht werden konnte. Sobald Code Red einen Rechner kompromittiert hatte, begann er damit zufällig IP-Adressen zu generieren und diese anschließend auf die gleiche Art und Weise anzugreifen. Dabei wurde vorher nicht überprüft, ob auf diesen Rechnern ebenfalls eine verwundbare Version des IIS läuft, sondern es wurde direkt die Anfrage gestellt, die die gerade beschriebene Schwachstelle ausnutzt. Folglich konnte nur ein Bruchteil der Computer, die sich hinter den generierten IP-Adressen befanden, infiziert werden. Des Weiteren wurde der Zufallsgenerator immer mit demselben Startwert initialisiert, so dass stets die gleichen IP-Adressen generiert wurden. Die zweite Variante von Code Red behob diesen Fehler und erreichte folglich eine deutlich höhere Verbreitung als die erste Variante. Unabhängig von der Version des Code Red war die Verbreitung lediglich davon abhängig, ob auf den Computern mit den generierten IP-Adressen eine verwundbare Version des IIS läuft [MSC02,ZGT02]. In keinem Schritt der Verbreitungsroutine ist deren weiterer Verlauf von einer Benutzerinteraktion abhängig, so dass Code Red nicht als Social Malcode bezeichnet werden kann.

3.3.3. Blaster

Das letzte Gegenbeispiel für Social Malcode ist der Computerwurm *Blaster*. Dieser auch unter dem Namen *Lovesan* oder *MSBlast* bekannte Wurm wurde das erste Mal im August 2003 registriert. Die einzige Schadroutine des Wurms bestand in einem DDoS-Angriff gegen die Internetseite von *Microsoft*. Durch diesen Angriff sollte am 16. August 2003 die Domain *windowsupdate.com* attackiert werden, über die unter anderem der Software-Aktualisierungsdienst der Microsoft Windows-Betriebssysteme angeboten wird. Der Schaden für *Microsoft* hielt sich in Grenzen, da sie die Namensauflösung für diese Domain abstellten. Diese Gegenmaßnahme war problemlos möglich, da der interne Update-Mechanismus der Betriebssysteme die Domain *windowsupdate.microsoft.com* verwendete.

Blaster ist nicht als Social Malcode zu bezeichnen, da auch dessen Verbreitung vollkommen unabhängig von einer Benutzerinteraktion ablief: Blaster verbreitete sich über eine Schwachstelle, die in Windows an einer DCOM-Schnittstelle (Distributed Component Object Model) mit RPC (Remote Procedure Call) bestand. Durch diese Schwachstelle konnte über Port 135 ein Buffer-Overflow erzeugt werden, um somit beliebigen Code auf dem entfernten Rechner zur Ausführung zu bringen. *Microsoft* veröffentlichte einen Monat vor dem Auftreten von Blaster einen ersten Patch (MS03-026), dennoch infizierte Blaster zeitweise mehr als 100 000 Rechner pro Stunde. Anfällig für eine Infektion mit Blaster waren die Betriebssysteme Windows 2000 und Windows XP [BCJ+05, Bra05].

3.4. Diskussion

Nachdem in den beiden vorangegangenen Abschnitten einige Beispiele und Gegenbeispiele für Social Malcode vorgestellt wurden, wird im Folgenden die Definition von Social Malcode genauer beleuchtet. Dabei gehen wir in diesem Abschnitt auf die Unterschiede zwischen der Definition dieses Kapitels und der vorläufigen Beschreibung

von Social Malcode aus dem ersten Kapitel ein. Zudem beschreiben wir die nun bereits mehrfach verwendeten Begriffe der Schad- und Verbreitungsroutine detaillierter. Abschließend werden Probleme der Definition aufgezeigt, die deutlich machen, dass die Definition noch weiter verfeinert werden muss.

3.4.1. Vom Social Engineering zur Benutzerinteraktion

Der Unterschied zwischen Definition 3.5 und der Beschreibung von Social Malcode aus Abschnitt 1.2 besteht im Wesentlichen darin, dass in der Definition für die Weiterverbreitung nun statt Social Engineering eine Benutzerinteraktion gefordert wird.

Betrachtet man beide Formulierungen genauer, fällt auf, dass die beiden Forderungen recht ähnlich sind und die Benutzerinteraktion der Definition aus dem Social Engineering der vorläufigen Beschreibung folgt: Ein Social Engineer nutzt psychologische Tricks und Täuschungen, um sein Opfer dazu zu bewegen, ihm Information I zukommen zu lassen, in deren Besitz er normalerweise nicht sein darf. Da ein Angreifer in der Regel seine Schadprogramme (in diesem Fall Social Malcode) nicht persönlich seinen Opfern übergibt, muss sich der durch die Beschreibung in Abschnitt 1.2 geforderte Social Engineering Aspekt innerhalb des Social Malcode selbst oder dem Transportmedium befinden, über das der Social Malcode zum Opfer transportiert wird. In der Regel ist dies ein elektronischer Kanal, wie zum Beispiel das Medium E-Mail. Aus dem gleichen Grund ist der Rückkanal, der die vom Angreifer angestrebte Information I zu diesem überträgt, häufig auch ein elektronischer Kanal – etwa das Internet. Folglich muss die Information I vom Opfer in einer Anwendung eingegeben werden, damit diese über den elektronischen Kanal verschickt werden kann. Dies wiederum ist nach Definition 3.4 auf Seite 49 eine Benutzerinteraktion.

Es gilt also, dass aus dem Social Engineering, das in der vorläufigen Beschreibung von Social Malcode gefordert wurde, stets eine Benutzerinteraktion resultiert. Die allgemeine Form dieser Aussage, also dass jede Form von Social Engineering immer eine Benutzeraktion hervorruft, gilt jedoch nicht: Beispielsweise erhält ein Social Engineer bei dem in Abschnitt 2.3.4.2 beschriebenen Dumpster Diving ebenfalls unberechtigt Informationen, jedoch ist in diesem Prozess keine Benutzerinteraktion enthalten. Allerdings verbreitet sich Social Malcode auch nicht ausschließlich durch Dumpster Diving, so dass dieses Beispiel nicht der spezielleren Aussage widerspricht.

3.4.2. Unterschied zwischen Schad- und Verbreitungsroutine

Bei Schadprogrammen unterscheidet man zwischen der Verbreitungs- und der Schadroutine eines Schadprogramms. Die Verbreitungsroutine ist dafür zuständig, dass das Schadprogramm neue Rechner infiziert und ist üblicherweise unabhängig von der Schadroutine. Diese enthält die eigentliche Funktionalität des Schadprogramms und ist dafür zuständig, dass der Autor des Schadprogramms seine durch das Schadprogramm beabsichtigen Ziele erreichen kann.

3.4.2.1. Verbreitungsroutine

Betrachtet man die Verbreitungsroutinen von Schadprogrammen, unterscheiden wir zwei verschiedene Arten: Zum einen sich autonom verbreitende Schadprogramme und zum anderen Schadprogramme, die sich nicht-autonom verbreiten.

Sich autonom verbreitende Schadprogramme werden definitionsgemäß als *Wurm* bezeichnet. Der Verbreitungsprozess eines Wurms besteht aus mehreren Phasen, die nachfolgend kurz erläutert werden [SZ04]:

1. **Exploit**: Der Wurm nutzt eine oder mehrere Schwachstellen auf einem verwundbaren System aus, wie etwa einen Buffer Overflow oder fehlkonfigurierte Dateifreigaben. Durch diesen Schritt bringt der Wurm auf dem Zielsystem seine ersten Instruktionen zur Ausführung.
2. **Nachladen**: Diese Instruktionen laden den Rest des Wurmes auf den kompromittierten Rechner und führen diesen aus. Im ersten Schritt können üblicherweise, aufgrund der beschränkten Payload-Größe der meisten Exploits, nur wenige Instruktionen ausgeführt werden, weshalb dieser zweite Schritt separat durchgeführt wird.
3. **Zielgenerierung**: Nachdem der Wurm ausgeführt wird, beginnt er damit nach neuen Rechnern zu suchen, die er infizieren kann. Dabei kann er auf dem befallenen Rechner in Dateien nach IP-Adressen suchen oder diese zufällig oder auch systematisch generieren [SPW02].
4. **Zielüberprüfung**: Anschließend überprüft der Wurm, ob die neuen, potentiellen Opfer der Liste des dritten Schritts ebenfalls anfällig für die Exploits aus dem ersten Schritt sind. Ist dies der Fall, beginnt der komplette Verbreitungsprozess von vorn: Die Schwachstelle wird ausgenutzt, der Rest des Wurms wird nachgeladen und auf dem neu infizierten Rechner ausgeführt, eine Liste mit neuen Opfern wird generiert, überprüft und abschließend gegebenenfalls kompromittiert.

Diese vier Schritte laufen vollautomatisch ab, weshalb diese Art der Verbreitung *autonom* genannt wird. Zudem beziehen sich alle vier Schritte ausschließlich auf die Verbreitung des Schadprogramms. Die eigentliche Schadroutine kommt in einem weiteren Schritt zur Ausführung, der irgendwann nach dem zweiten Schritt erfolgt, aber in der obigen Aufzählung der Verbreitungsschritte eines Wurms nicht enthalten ist.

Ist der Verbreitungsprozess zu irgendeiner Stelle von einer menschlichen Interaktion abhängig, spricht man hingegen von einer *nicht-autonomen* Verbreitung. Alle vorgestellten Schadprogramme aus Abschnitt 3.3 sind Beispiele für solch eine Art von Verbreitung. Folgt ein Empfänger einer Waledac- oder Storm-E-Mail nicht dem Verweis in der E-Mail, so wird sein Rechner auch nicht mit dem entsprechenden Schadprogramm infiziert. Folgt er dem Verweis zu einem späteren Zeitpunkt, so wird auch der Rechner erst zu diesem Zeitpunkt infiziert. Durch diese menschlich verursachte Verzögerung im Verbreitungsprozess ergibt sich, dass sich autonom verbreitende Schadprogramme in der Regel schneller verbreiten als sich nicht-autonom verbreitende Schadprogramme.

Dadurch, dass der Verbreitungsprozess von sich autonom verbreitenden Schadprogrammen vollautomatisch abläuft, kann diese Art von Schadprogrammen nicht als

Social Malcode nach Definition 3.5 bezeichnet werden. Der Grund liegt in der fehlenden Benutzerinteraktion im Verbreitungsprozess. Wäre diese für den Verbreitungsprozess erforderlich, so würde die Definition zwar zutreffen, jedoch verbreitet sich das Schadprogramm dann nicht mehr autonom, sondern nicht-autonom.

3.4.2.2. Schadroutine

Neben einer möglichen Verbreitungsroutine besitzen Schadprogramme auch eine Schadroutine. Diese ist zwar nicht zwingend erforderlich, jedoch ist sie, wie bereits erwähnt, dafür verantwortlich, dass der Autor seine durch das Schadprogramm beabsichtigten Ziele erreichen kann. Die Anzahl möglicher Schadroutinen ist unbegrenzt und der Autor kann die Schadroutine beliebig festlegen. Nachfolgend sind einige Schadroutinen exemplarisch aufgelistet, die häufig bei Schadprogrammen verwendet werden. Bei jedem dieser Beispiele sind ebenfalls die Schutzziele eines Informationssystems [EBB04] (*Vertraulichkeit*, *Integrität* und *Verfügbarkeit*) angegeben, die durch die entsprechende Schadroutine verletzt werden.

- **Heimliche Einnistung im System**: Damit ein Schadprogramm auch nach einem Neustart des befallenen Systems noch aktiv bleibt, wird das System häufig so modifiziert, dass das Schadprogramm mit dem System gestartet wird. Unter Windows können dazu zum Beispiel bestimmte Schlüssel der Registrierungsdatenbank verwendet werden. Häufig wird das System auch noch so verändert, dass herkömmliche Systemanwendungen nichts von der Existenz des Schadprogramms bemerken. Durch diese Modifikationen wird die Integrität des Systems verletzt.
- **Extraktion sensibler Informationen**: Ein finanziell motivierter Angreifer ist an Objekten/Ressourcen interessiert, die er verkaufen kann. Nach dem Befall eines Rechners kann eine Schadroutine zum Beispiel Passwörter, Softwarelizenzen, CD-Keys oder auch vertrauliche Dokumente extrahieren und dem Angreifer zukommen lassen [HEF09]. Das dadurch verletzte Schutzziel ist die Vertraulichkeit der gestohlenen Informationen.
- **Denial of Service**: Die Schadroutine eines *Denial-of-Service*-Angriffs (kurz DoS) bezweckt die Verfügbarkeit des Angriffsziels durch Überlastung zu stören. Die verteilte Variante dieser Angriffsart nennt sich *Distributed Denial of Service* (kurz DDoS) und wird häufig von Botnetzen ausgeführt. Dabei belasten beispielsweise die Bots eines Botnetzes einen Webserver so stark durch gleichzeitige Anfragen, dass reguläre Anfragen nicht mehr bearbeitet werden können [MR04, PLR07].

Weitere Schadroutinen können etwa der Versand von Spam-Mails, die Kompromittierung zusätzlich installierter Sicherheitseinrichtungen, damit das Schadprogramm schwerer oder gar nicht erkannt wird, die Kommunikation mit dem Autor des Schadprogramms und die Ausführung dessen Befehle oder auch die Modifikation einer Internetseite sein, die sich möglicherweise auf einem befallenen Rechner befindet.

3.4.3. Probleme der Definition

Eine exakte Definition des Begriffs Social Malcode ist aus verschiedenen Gründen schwierig. Teil (a) der Definition 3.5 fordert von einem Code eine schadhafte Funktionalität, wie etwa die oben beschriebenen Schadroutinen. Damit diese Funktionalität greifbarer wird, verlangt die Definition die Möglichkeit eines nicht-autorisierten Zugriffs auf eine Ressource des Opfers durch den Angreifer. Nun stellt sich jedoch die Frage, wer und wann jemand ein Zugriffsrecht auf eine Ressource besitzt. Dies liegt im Ermessen des Besitzers der Ressource und somit ist die Frage nicht allgemeingültig zu beurteilen. Ebenfalls kann ein und dasselbe Programm, abhängig von seinem Benutzer, als schadhaft oder als gutartig bezeichnet werden. Die beiden nachfolgenden Beispiele verdeutlichen diese Schwierigkeiten und zeigen, dass Code, der für eine Person eine schadhafte Funktionalität besitzt, nicht automatisch auch von anderen Personen als schadhaft beurteilt werden muss.

Angenommen auf den Rechnern zweier Personen A und B wird ein Browser Helper Object (BHO) installiert. Dies sind Windows-Bibliotheken, mit denen sich der Funktionsumfang des Internet Explorer erweitern lässt. Dadurch, dass die BHOs kompletten Zugriff auf den Internet Explorer und die Struktur der angezeigten Internetseiten haben, können die BHOs das Surfverhalten von A und B aufzeichnen und an eine dritte Partei verschicken – beispielsweise zur Realisierung personalisierter Werbung. Person A steht diesem Verhalten eher gleichgültig gegenüber und ihn stört es nicht, dass sein Surfverhalten aufgezeichnet wird. Ganz im Gegenteil: Er freut sich vielmehr über die auf seine Interessen abgestimmte Werbung. Benutzer B hingegen ist sehr am Schutz seiner Privatsphäre interessiert und nicht mit dem Aufzeichnen seines Surfverhaltens einverstanden. Demnach enthält das BHO für Benutzer A keine schadhafte Funktionalität. Im Gegensatz dazu steht Benutzer B, der das BHO sehr wohl als Schadprogramm betrachtet.

Ein anderes Beispiel sind Netzscanner, mit deren Hilfe sich die offenen und geschlossenen Ports eines Rechners ermitteln lassen. Werden diese Scanner von Angreifern benutzt, um festzustellen, über welchen Port eines Rechners ein Angriff erfolgreich sein kann, so ist dies zweifelsohne eine schadhafte Funktionalität bzw. um genau zu sein, eine schadhafte Verwendung der Funktionalität des Netzscanners. Wird ein Netzscanner hingegen von einem Administrator benutzt, um unnötig offene Ports an Rechnern seines Netzes zu finden und anschließend zu schließen, ist die Funktionalität des Netzscanners nicht mehr als schadhaft zu bezeichnen.

Eine andere Frage, die sich beim Betrachten der Definition 3.5 stellt, ist: Was passiert, wenn man Social Malcode als Payload eines sich autonom verbreitenden Wurmes benutzt? Handelt es sich dann immer noch um Social Malcode oder ist diese Bezeichnung durch die autonome Verbreitung des Wurms dann nicht mehr gerechtfertigt? Eine genaue Beantwortung dieser Fragen ist zum jetzigen Zeitpunkt noch nicht möglich.

Oftmals ist auch nicht klar zwischen Schad- und Verbreitungsroutine zu trennen. Der Morris-Wurm besitzt beispielsweise keine explizite Schadroutine. Sein einziges Ziel besteht darin, sich möglichst schnell und effizient zu verbreiten. Obwohl der Wurm nicht die Vertraulichkeit und in manchen Fällen auch nicht die Verfügbarkeit der befallenen Rechner beeinträchtigt, ist deren Integrität auf jeden Fall nicht mehr gewährleistet: Dadurch, dass sich der Wurm auf den Rechner nachladen und ausführen

muss, werden Daten auf dem befallenen System verändert. Aufgrund verschiedener Designfehler des Wurms, wurde zudem in vielen Fällen die Verfügbarkeit einzelner Dienste oder ganzer Systeme verletzt [Orm03]. Obwohl dies die Folgen der Verbreitungsroutinen des Wurms sind, ist diese Aktivität als nicht-erwünschenswert und somit auch als schadhaft zu erachten.

Auch kann ein Teil der Schadroutine für die Verbreitung des Schadprogramms verwendet werden: Bots des Botnetzes *Waledac* haben keine explizite Verbreitungsroutine und im Wesentlichen nur eine Schadroutine. Diese besteht daraus, Befehle des Botmasters entgegenzunehmen und diese auszuführen. Recht häufig wird einem Teil des Botnetzes jedoch der Befehl gegeben, Spam-Mails mit einem Verweis auf die aktuelle Version des Bot zu verschicken. Obwohl dies ein Teil der Schadroutine ist, dient es der Verbreitung von Waledac [SGE+09].

Als letzten Aspekt dieser Diskussion verweisen wir auf das Thema Phishing. Bei einer Phishing-Seite handelt es sich wegen nachfolgenden Argumenten um bösartigen Code.

1. In der Regel wird von einem Angreifer eine Kombination aus HTML und PHP verwendet, um eine Phishing-Seite zu erstellen. Der HTML-Teil dient der Imitation der Internetseite eines Online-Dienstleisters, dessen Kunden die Opfer des Phishing-Angriffs sind. Die Benutzereingaben der Opfer werden hingegen durch den PHP-Teil der Phishing-Seite abgespeichert. Da die beiden Teile in Form eines Textes spezifiziert werden und da HTML und PHP eine Auszeichnungs- und eine Skriptsprache sind, ist die Phishing-Seite nach Definition 3.2 als Code anzusehen.
2. Ebenfalls entspricht eine Phishing-Seite dem Teil (a) der Definition 3.5. Die gefährdeten Ressourcen sind in dem Fall die eingegebenen Zugangsdaten eines Opfers. Da ein Angreifer erst mit Hilfe der Imitation einer originalen Internetseite ein Opfer dazu bewegen möchte, ihm die Zugangsdaten preiszugeben, ist nicht davon auszugehen, dass der Angreifer Zugriffsrechte auf die Zugangsdaten besitzt. Folglich ist der Zugriff auf diese Daten über einen Phishing-Angriff nicht-autorisiert.

Eine Phishing-Seite ist also Code, der einem Angreifer einen nicht-autorisierten Zugriff auf eine oder mehrere Ressourcen ermöglicht. Dennoch ist Definition 3.5 für eine Phishing-Seite eher unpassend. Dies liegt daran, dass sich eine Phishing-Seite üblicherweise nicht oder nicht im herkömmlichen Sinn verbreitet. Eine Verbreitung im herkömmlichen Sinn ist die ständige Infektion noch nicht befallener Rechner. Es lässt sich somit keine Aussage darüber treffen, ob für Phishing Teil (b) der Definition erfüllt ist oder nicht. Obwohl Phishing intuitiv ein Paradebeispiel für einen auf Social Engineering beruhenden Angriff ist, handelt es sich bei Phishing (bezüglich der Definition von Social Malcode) um eine Grauzone.

3.5. Zweite Definition

Vor allem die letzten Aspekte der vorangegangenen Diskussion verdeutlichen, dass die Definition 3.5 noch unklare Formulierungen enthält. In diesem Abschnitt werden wir

weitere Begriffe definieren, um schließlich eine zweite Definition von Social Malcode zu geben, die diese Problematik behebt.

3.5.1. Schadcode

In Abschnitt 3.4 wurde bereits bösartiger Code angesprochen. Aus diesem Grund definieren wir als nächstes den Begriff Schadcode, um somit eine formale Definition für diese Sorte Code zu erhalten. Die Definition unterscheidet explizit zwischen Verbreitungs- und Schadroutinen, und zwar in einer Art und Weise, wie es bereits für Schadprogramme in der vorangegangen Diskussion in Abschnitt 3.4.2 vorgestellt wurde. Der Begriff der Verbreitungsroutine wird später noch genauer definiert.

Definition 3.6: Schadcode
Ein *Schadcode* ist ein Tupel $M = (V, S)$, bestehend aus einer Menge von Verbreitungsroutinen V und einer Menge von Schadroutinen S. Die Vereinigung dieser beiden Mengen darf nicht leer sein, d. h. es gilt $V \cup S \neq \varnothing$. Des Weiteren müssen die Elemente der beiden Mengen V und S folgende Kriterien erfüllen:

(a) Jedes $v_i \in V$ ist Code, der dafür verantwortlich ist, dass sich der Schadcode M weiterverbreitet.

(b) Jedes $s_i \in S$ ist Code und ermöglicht einem Angreifer, nicht-autorisierten Zugriff auf eine oder mehrere Ressourcen eines Opfers zu erhalten.

Beim ersten Lesen der Definition könnte nicht sofort klar sein, warum Schadcode $M_{noMal} = (V, \varnothing)$ ohne eine Schadroutine, dennoch als *Schad*code bezeichnet werden sollte. Nach obiger Definition muss M_{noMal} mindestens eine Verbreitungsroutine besitzen, da nicht beide Mengen V und S leer sein dürfen. In der Diskussion wurde am Beispiel des Morris-Wurms gezeigt, dass die Verbreitungsroutinen mindestens die Integrität eines Systems verletzten und manchmal sogar die Verfügbarkeit einschränken. Somit haben Verbreitungsroutinen ebenfalls schadhafte Nebeneffekte und M_{noMal} ist auch sprachlich als schadhafter Code zu bezeichnen.

Bei den Verbreitungsroutinen eines Schadprogramms unterscheidet man, wie bereits in der Diskussion erläutert, zwischen *autonom* und *nicht-autonom* arbeitenden Routinen. Diese Unterteilung übernehmen wir auch für die Verbreitungsroutinen V aus der gerade gelieferten Definition von Schadcode. Im Folgenden bezeichnet V_{aut} die Verbreitungsroutinen, die autonom arbeiten, und $V_{\neg aut}$ die Routinen, die eine menschliche Interaktion erfordern und somit nicht-autonom sind. Durch die Einteilung der Verbreitungsroutinen in V_{aut} und $V_{\neg aut}$, entsteht eine 2-Partition der Gesamtmenge der Verbreitungsroutinen V, so dass $V = V_{aut} \dot{\cup} V_{\neg aut}$ gilt. Durch diese Partitionierung der Verbreitungsroutinen, lassen sich Schadcodes bezüglich ihrer Verbreitung in die vier nachfolgenden Klassen einteilen:

1. $(\varnothing, \boldsymbol{S \neq \varnothing})$: Schadcodes, die sich nicht verbreiten
   Beispiel: Phishing oder der *Bundestrojaner*
2. $(\boldsymbol{V_{aut}}, \boldsymbol{S})$: Schadcodes, die sich ausschließlich autonom verbreiten
   Beispiel: *Code Red*

3. $(\boldsymbol{V_{\neg aut}}, \boldsymbol{S})$: Schadcodes, die sich ausschließlich nicht-autonom verbreiten
   Beispiel: *LoveLetter*
4. $(\boldsymbol{V_{aut}} \cup \boldsymbol{V_{\neg aut}}, \boldsymbol{S})$: Schadcodes, die sowohl autonom arbeitende als auch nicht-autonom arbeitende Verbreitungsroutinen verwenden
   Beispiel: *Nimda*

Da sich Schadcodes der ersten Klasse nicht weiterverbreiten und die Vereinigung der Schad- und Verbreitungsroutinen eines Schadcodes nicht leer sein darf, müssen die Schadcodes dieser Klasse zwingend mindestens eine Schadroutine besitzen. Das Schadprogramm Nimda ist ein Beispiel für die vierte Klasse von Schadcode. Da wir dieses nicht näher beschrieben haben, verweisen wir für eine ausführliche Beschreibung und Analyse des Schadprogramms auf eine Arbeit [MRRV01] von Mackie et al. Für die vorliegende Arbeit ist ohnehin nur interessant, dass Nimda fünf Verbreitungsroutinen besitzt. Davon arbeiten drei nicht-autonom und zwei autonom. Tabelle 3.1 enthält eine Auflistung der von Nimda verwendeten Verbreitungsroutinen und gibt an, ob diese V_{aut} oder $V_{\neg aut}$ zuzurechnen sind [SPW02].

Tabelle 3.1.: Verbreitungsroutinen des Schädlings Nimda
Dies ist eine Auflistung der Verbreitungsroutinen des Schädlings Nimda. In der zweiten Spalte ist jeweils angegeben, ob es sich dabei um eine autonom oder um eine nicht-autonom arbeitende Verbreitungsroutinen handelt.

Verbreitungsroutine	**Arbeitsweise**
Ausnutzen der Schwachstelle CVE-2000-0884 des Microsoft IIS-Servers [CVE00]	autonom
Kopieren auf freigegebene Netzwerkordner	autonom
Missbrauchen der Hintertüren, die die Computerwürmer *Code Red (Version 2)* und *sadmind* eingerichtet haben	autonom
Versand von E-Mails mit einer Instanz des Schädlings als Dateianhang	nicht-autonom
Manipulation von Webservern auf kompromittierten Rechnern, damit diese einen Drive-By-Download Exploit enthalten	nicht-autonom

Abbildung 3.3 stellt die vier Klassen von Schadcode grafisch dar. Die Klasse der sich ausschließlich autonom verbreitenden Schadcodes ist grau hinterlegt. In die drei anderen Klassen fallen die Schadcodes, die im Fokus dieser Arbeit stehen. Warum dies so ist, wird in Abschnitt 3.5.3 deutlich. Der Größe der Flächen kommt in der Abbildung keine Bedeutung zu, sondern sie ist nur aus Darstellungsgründen so gewählt. Es ist sogar vielmehr davon auszugehen, dass im Moment die sich ausschließlich autonom verbreitenden Schadcodes (also die grau hinterlegte Klasse) noch den größten Teil der aktiven Schadcodes ausmacht. Jedoch ist aufgrund immer sicherer werdender Betriebssysteme und Netzdienste ein entgegengesetzter Trend festzustellen [EWH09].

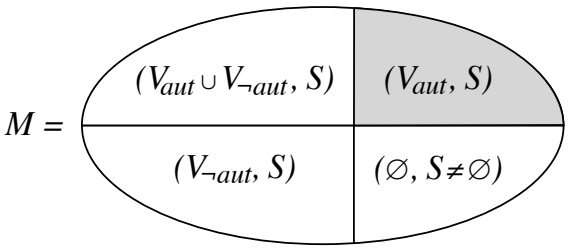


Abbildung 3.3.: Klassifizierung von Schadcodes anhand der Verbreitungsroutinen
Durch eine Partitionierung der Verbreitungsroutinen in autonom arbeitende und nicht-autonom arbeitende Routinen entstehen die vier in der Abbildung gezeigten Klassen von Schadcodes.

3.5.2. Verbreitungsroutine

In den bisherigen Definitionen und speziell im vorangegangenen Abschnitt dieses Kapitels wurde bereits mehrfach das Thema der Verbreitung von Code oder Programmen angesprochen, ohne den Begriff Verbreitung oder Verbreitungsroutine jedoch formal zu definieren. Intuitiv ist eine Verbreitungsroutine eines Schadcodes selbst wieder Code, der die Vervielfachung und Weitergabe des Schadcodes auf mindestens ein System ermöglicht, das sich von dem System unterscheidet, auf dem sich der Schadcode aktuell befindet. Nachfolgende Definition formalisiert den Begriff der Verbreitungsroutine und bildet eine Grundlage für die finale Definition von Social Malcode.

Definition 3.7: Verbreitungsroutine
Eine *Verbreitungsroutine* eines Schadcodes M ist selbst wieder Code, dessen Verhalten mittels eines endlichen, gerichteten Graphen $G = (Z, U)$ dargestellt werden kann. Dieser Graph muss die folgenden Bedingungen erfüllen:

(a) Die Elemente der Knotenmenge Z sind Zustände der am Verbreitungsprozess beteiligten Systeme.

(b) Die Elemente der Kantenmenge U sind Übergänge zwischen den Systemzuständen aus Z.

(c) Innerhalb des Graphen G muss ein Zyklus existieren, der eine ausgewiesene Kante $\hat{u} \in U$ enthält, die den Übergang des Schadcodes M von einem System zum nächsten System repräsentiert. Das bedeutet, dass der Ursprungsknoten von $\hat{u}$ ein Zustand eines Systems s_1 und der Zielknoten von $\hat{u}$ ein Zustand eines Systems $s_2 \neq s_1$ ist.

Die Zustände der Knotenmenge Z beziehen sich nur auf Konfigurationen und Eigenschaften eines Systems, die für die Verbreitung eines Codes relevant sind. Beispiele für Zustände eines Systems sind: die Anzahl der ungelesenen E-Mails in einem E-Mail-Client (wenn man die Verbreitung eines E-Mail-Wurms betrachtet), der aktuelle Patch-Level eines Browsers (bei der Verbreitung durch Drive-By-Download) oder auch die Existenz eines verwundbaren Dienstes (etwa des IIS-Servers von *Microsoft*, wenn man die Verbreitung von Code Red betrachtet). Von allen irrelevanten Zuständen

eines Systems, die keinen Einfluss auf die Verbreitung eines Codes nehmen, abstrahiert die Knotenmenge der Verbreitungsroutine aus Definition 3.7.

Die Kanten des Graphen aus der Definition stellen die Übergänge zwischen den einzelnen Systemzuständen dar. Dies können, vom System aus gesehen, interne und externe Aktionen sein, welche die Zustandsübergänge anstoßen. Interne Übergänge sind beispielsweise das Ausführen einer Berechnung oder eines Exploit. Beispiele für externe Aktionen sind jegliche Benutzerinteraktionen oder auch externe Ereignisse, wie etwa das Eintreffen einer E-Mail.

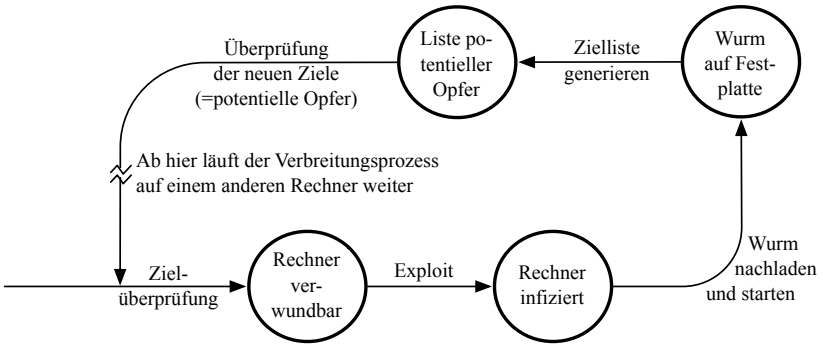


Abbildung 3.4.: Graph der Verbreitungsroutine eines Wurms
Dies ist die grafische Darstellung der Verbreitungsroutine eines Wurms. Die vier Zustandsübergänge entsprechen den vier Verbreitungsschritten, die in Abschnitt 3.4.2.1 erläutert werden.

Abbildung 3.4 zeigt exemplarisch den Graphen der Verbreitungsroutine eines Wurms. Der durch eine Doppellinie markierte Knoten gehört definitionsgemäß nicht zu der Verbreitungsroutine, erhöht aber das Verständnis des Graphen, da er den Start der Wurmverbreitung kennzeichnet. Die Zustandsübergänge entsprechen genau den vier Schritten einer Wurmverbreitung, wie sie in Abschnitt 3.4 beschrieben ist: Exploit, Nachladen, Zielgenerierung und Zielüberprüfung. Diese vier Kanten bilden zusammen den von der Definition einer Verbreitungsroutine geforderten Zyklus. Ferner beinhaltet dieser Zyklus die ausgewiesene Kante, die den Übergang von einem zum nächsten System kennzeichnet. Dies ist in dem Beispiel der Abbildung die Kante zwischen den beiden Knoten mit den Beschriftungen *Liste potentieller Opfer* und *Rechner verwundbar*. Weitere Beispiele folgen in dem nächsten Abschnitt.

3.5.3. Social Malcode

Mit Hilfe der beiden vohergehenden Definitionen liefern wir nun die endgültige Definition von Social Malcode.

Definition 3.8: Social Malcode
Social Malcode ist ein Schadcode $M = (V, S)$. Wenn dieser Schadcode $n \geq 1$ Verbreitungsroutinen $V = \{v_1, v_2, \ldots, v_n\}$ enthält, dann muss mindestens eine Verbreitungsroutine $v_i = (Z_i, U_i)$ eine Sequenz von drei Zustandsübergängen der Form

$u_{i_1} \rightarrow u_{i_2} \rightarrow u_{i_3}$ enthalten. Die Pfeile symbolisieren eine kausale Folgerung und die Zustandsübergänge $u_{i_1}, u_{i_2}, u_{i_3} \in U_i$ haben dabei folgende Bedeutung:

- u_{i_1}: Eine Herausforderung oder Provokation einer Benutzerinteraktion
- u_{i_2}: Eine Benutzerinteraktion
- u_{i_3}: Eine Code-Ausführung, welche die Infektion des zugrunde liegenden Systems verursacht

Die in der Definition beschriebene Sequenz von Zustandsübergängen einer Verbreitungsroutine v_i enthält eine Benutzerinteraktion (u_{i_2}). Somit gilt nach Abschnitt 3.5.1, dass $v_i \in V_{\neg aut}$, d. h. v_i ist eine nicht-autonom arbeitende Verbreitungsroutine. Damit ein Opfer die Eingabe durchführt, die bei der Benutzerinteraktion erforderlich ist, erhält das Opfer vom Autor des Schadcodes eine Art Herausforderung (u_{i_1}). Diese soll die Benutzerinteraktion des Opfers provozieren, üblicherweise durch Social Engineering. Aus der Eingabe resultiert dann eine Code-Ausführung (u_{i_3}), welche die eigentliche Infektion des neuen Systems verursacht – etwa die Ausführung eines Drive-By-Download Exploit, nachdem eine Internetseite geladen wurde, oder das Öffnen einer heruntergeladenen Datei.

Die Sequenz $u_{i_1} \rightarrow u_{i_2} \rightarrow u_{i_3}$ von Zustandsübergängen muss keineswegs zusammenhängend sein. Sowohl zwischen den Übergängen u_{i_1} und u_{i_2} als auch zwischen u_{i_2} und u_{i_3} können sich weitere Zustandsübergänge befinden. Abbildung 3.5 verdeutlicht dies anhand einer schematischen Darstellung: Damit eine Sequenz aus Zustandsübergängen der geforderten Sequenz entspricht, ist es lediglich wichtig, dass der kausale Zusammenhang zwischen den drei geforderten Übergängen existiert und nicht, ob diese unmittelbar aufeinanderfolgend auftreten. Dabei ist es auch nicht erforderlich, dass die zusätzlichen Übergänge ebenfalls eine *Herausforderung*, *Benutzerinteraktion* oder eine *Code-Ausführung* darstellen.

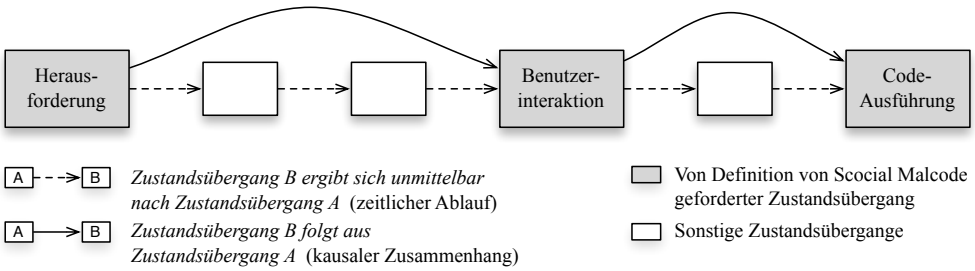


Abbildung 3.5.: Durch die Definition geforderte Sequenz von Zustandsübergängen
Die innerhalb der Definition von Social Malcode geforderte Sequenz von Zustandsübergängen muss keinesfalls zusammenhängend sein – sie kann auch durch andere Zustandsübergänge unterbrochen sein.

Nun lassen sich auch die nicht beantworteten Fragen der Diskussion aus Abschnitt 3.4.3 klären. Es stellte sich die Frage, ob man einen gegebenen Social Malcode $M = (V_m, S_m)$ immer noch als Social Malcode bezeichnen kann, wenn man ihn als Payload eines Wurms $W = (V_w, S_w)$ verwendet. Da sich der Wurm autonom verbreitet, gilt $v_w \in V_{aut}$ für alle Verbreitungsroutinen $v_w \in V_w$. Dies gilt auch noch für den Schadcode $(V_w, S_w \cup (V_m, S_m))$, der entsteht, wenn man M dem Payload des

Wurms W hinzufügt. Somit ist Social Malcode verpackt in einem Wurm nach Definition 3.8 nicht mehr als Social Malcode zu bezeichnen. Des Weiteren handelt es sich bei der Kombination der beiden Schadcodes um einen vollständig neuen Schadcode, der insbesondere auch nicht mehr dem Social Malcode M entspricht.

Es gibt nach Definition 3.8 genau zwei Bedingungen, von denen eine erfüllt sein muss, damit Schadcode als Social Malcode bezeichnet werden kann:

1. Entweder enthält der Schadcode keine Verbreitungsroutinen,
2. oder der Schadcode enthält mindestens eine nicht-autonom arbeitende Verbreitungsroutine, welche die in der Definition beschriebene Sequenz von drei Zustandsübergängen aufweist.

Die erste Bedingung ist dafür verantwortlich, dass Phishing keine Grauzone mehr bezüglich der Definition von Social Malcode darstellt. Da sich eine Phishing-Seite nicht verbreitet, entspricht sie der neuen Definition von Social Malcode. Dies trifft auf jeglichen sich nicht verbreitenden Schadcode zu. Auf den ersten Blick könnte dies zu allgemein beziehungsweise zu allumfassend wirken, was es jedoch nicht ist. Erstens existiert naturgemäß wenig Schadcode, der sich nicht verbreitet und zweitens fällt dieser intuitiv unter die Kategorie Social Malcode: Angenommen ein Autor erstellt Code mit einer schadhaften Funktionalität, der sich nicht verbreitet, aber dennoch für einen Angriff verwendet werden soll. Nach Definition 3.7 verlässt der Code nicht das System, auf dem er erstellt wurde. Somit muss der Angreifer ein potentielles Opfer auf irgendeine Art und Weise zu dem Code „locken“ – unter der Voraussetzung, dass kein Benutzer freiwillig Opfer eines Angriffs wird. Und genau diese beeinflussende Handlung des Lockens oder Köderns ist wiederum ein Akt des Social Engineering, weshalb die Einstufung sich nicht verbreitender Schadcodes als Social Malcode gerechtfertigt ist.

Die zweite Bedingung umfasst alle Schadcodes, die unter anderem nicht-autonom arbeitende Verbreitungsroutinen verwenden. Drei Beispiele solcher Verbreitungsroutinen werden in den folgenden Abschnitten genauer vorgestellt.

3.5.3.1. Drive-By-Download

Das erste Beispiel ist die Verbreitungsroutine eines Social Malcode M, die folgendermaßen arbeitet: Nach der Infektion eines Rechners werden E-Mail-Adressen generiert oder das kompromittierte System danach durchsucht. An diese Adressen wird eine E-Mail geschickt, die einen Verweis auf eine Internetseite enthält. Diese ist so präpariert, dass beim Besuch der Seite ein Drive-By-Download Exploit ausgeführt wird. Voraussetzung hierfür ist ein Browser, der eine entsprechende Schwachstelle aufweist. Sollte der Exploit erfolgreich sein, so wird der Social Malcode M heruntergeladen und ausgeführt, wodurch der Rechner des E-Mail-Empfängers ebenfalls infiziert ist.

Abbildung 3.6 zeigt die grafische Darstellung der gerade beschriebenen Verbreitungsroutine. Die drei in der Definition von Social Malcode geforderten Zustandsübergänge sind durch einen Kommentar in Klammern gekennzeichnet: Die Herausforderung ist in diesem Beispiel die E-Mail, die den Benutzer dazu verleiten soll, dem in ihr enthaltenen Verweis zu folgen. Da dies nur durch eine Eingabe per Maus oder Tastatur

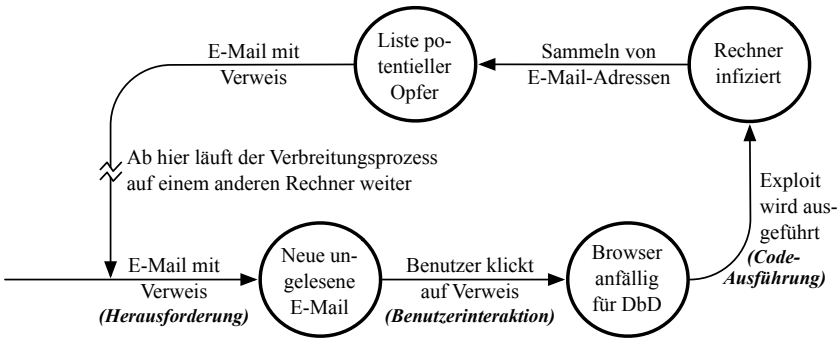


Abbildung 3.6.: Graph der Verbreitung per Drive-By-Download
Nach dem Erhalt einer E-Mail (Herausforderung), hängt es vom Verhalten des Benutzers (mögliche Benutzerinteraktion) ab, ob sich der Social Malcode weiterverbreitet (Code-Ausführung).

möglich ist, ist dies die geforderte Benutzerinteraktion. Weist der Browser auch noch eine entsprechende Schwachstelle auf, d. h. er ist anfällig für einen Drive-By Download Exploit (in der Abbildung durch *DbD* abgekürzt), so wird dieser ausgeführt. Dies ist wiederum die geforderte Code-Ausführung.

In der Abbildung ist auch der Zustandsübergang markiert, der den Übergang von einem System zum nächsten darstellt. Dies passiert bei der vorgestellten Verbreitungsroutine nach dem Versand der E-Mails (Zustand oben links): Wenn eine E-Mail abgeschickt wird, hat der Benutzer des sendenden Rechners keinen Einfluss mehr auf die Verbreitung des Social Malcode. Diese hängt nur noch von den E-Mail-Empfängern und deren Verhalten ab. Folgt ein Empfänger dem Verweis mit einem anfälligen Browser, so ist auch dessen Rechner infiziert. Wenn er dem Verweis jedoch nicht folgt, ist die Verbreitungskette des Social Malcode an dieser Stelle unterbrochen und der Schadcode kann sich nur noch über die anderen Empfänger und deren Rechner verbreiten.

3.5.3.2. Bösartig präparierter Dateianhang

Das nächste Beispiel ist eine Verbreitungsroutine, die genauso mit einer Herausforderung per E-Mail arbeitet. Jedoch findet hier die Infektion nicht über eine Internetseite statt, sondern über den Dateianhang der E-Mail. Nach der Infektion eines Rechners werden wie im ersten Beispiel E-Mail-Adressen generiert. An diese werden E-Mails versandt, denen eine Instanz des Schadcodes angehängt ist. Der Text der E-Mail muss so gestaltet sein, dass der Empfänger der Nachricht dazu bewegt wird, den Anhang auf seinem Rechner abzuspeichern und zu öffnen. Dadurch wird dieser Rechner ebenfalls infiziert. Der nächste Verbreitungszyklus wird dann durch das Generieren von E-Mail-Adressen auf dem neu infizierten Rechner initialisiert.

Abbildung 3.7 zeigt die grafische Darstellung der Verbreitungsroutine des zweiten Beispiels. Auch hier sind wieder die drei geforderten Zustandsübergänge durch Kommentare hervorgehoben. In der Abbildung ist zu erkennen, dass die Benutzerinteraktion in diesem Beispiel aus zwei separaten Benutzerinteraktionen besteht: Da beim Abspeichern eines Dateianhangs dieser üblicherweise nur auf der Festplatte abgelegt

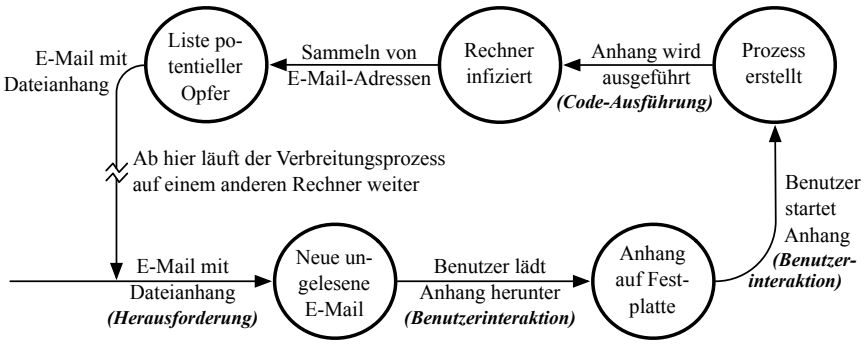


Abbildung 3.7.: Graph der Verbreitung per E-Mail-Anhang
Nach einer Herausforderung durch eine E-Mail, besteht die Benutzerinteraktion in diesem Beispiel aus zwei Schritten: Der Benutzer muss einen Anhang herunterladen und ausführen, damit es zu einer Infektion kommt.

wird, muss ein Benutzer neben dem Abspeichern des Dateianhangs, diesen auch noch explizit starten.

Dieses Beispiel zeigt, dass zwischen den in der Definition 3.8 geforderten drei Zustandsübergängen noch weitere Zustandsübergänge enthalten sein können. Der Zustandsübergang, in dem ein Benutzer einen Dateianhang herunterlädt, ist eine abstrahierte Sichtweise auf mehrere feingranularere Zustandsübergänge: Im Prinzip muss der Benutzer zuerst dem E-Mail-Client deutlich machen, dass er den Anhang herunterladen möchte. Dies funktioniert etwa durch einen Rechtsklick mit der Maus auf den Anhang und einem anschließenden Linksklick auf den Kontextmenüpunkt *Speichern unter* – also durch eine Benutzerinteraktion. Danach speichert der E-Mail-Client den Anhang auf der Festplatte, was wiederum eine Code-Ausführung ist. Anschließend kann der Benutzer durch eine Benutzerinteraktion, beispielsweise durch einen Doppelklick mit der Maus, den heruntergeladenen Anhang ausführen.

3.5.3.3. Waledac

Das letzte Beispiel modelliert eine etwas komplexere Verbreitungsroutine. Abbildung 3.8 zeigt die grafische Darstellung der Verbreitung von Waledac [SGE+09, BCF09]. Auch in diesem Beispiel ist das Medium, das zur Verbreitung genutzt wird, die E-Mail. Innerhalb der E-Mails befindet sich ein Verweis. Inhalt dieser E-Mails waren zum Beispiel Hinweise, in denen den Empfängern mitgeteilt wurde, dass eine Grußkarte für sie hinterlegt worden sei, oder erfundene Nachrichtenmeldungen, dass sich Barack Obama plötzlich weigert, neuer US-Präsident zu werden. In den E-Mails wurde behauptet, dass sich die Grußkarten oder genauere Informationen zu den Nachrichtenmeldungen auf der Internetseite befinden, auf die der Verweis innerhalb der E-Mail zeigt.

Folgt ein Benutzer dem Verweis innerhalb der E-Mail (Benutzerinteraktion), so gelangt er auf eine Internetseite, die dem Inhalt der E-Mail entspricht und auch der in der E-Mail angepriesene Inhalt ist dort zu finden. Jedoch müssen die Grusskarten erst

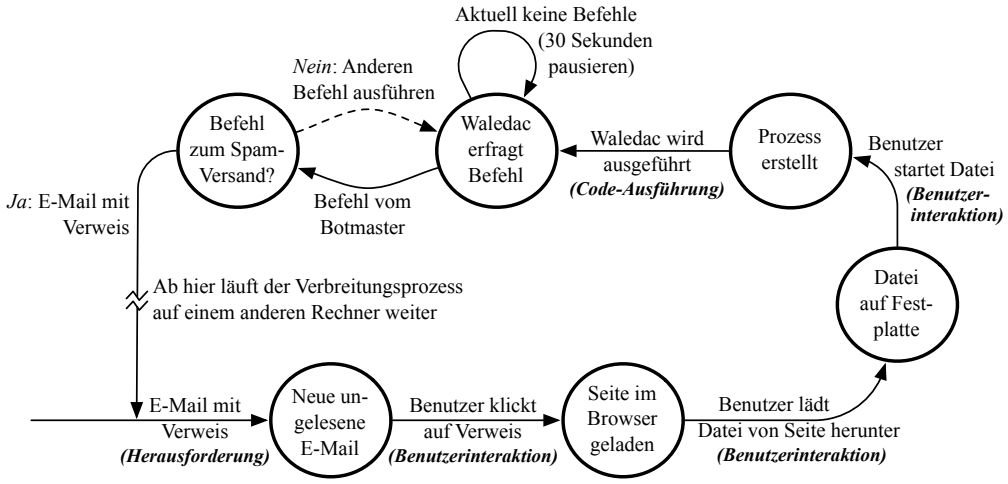


Abbildung 3.8.: Graph der Verbreitungsroutine von Waledac
Waledac verbreitet sich unter anderem über E-Mails, in denen sich ein Verweis befindet. Die Internetseite hinter dem Verweis soll den Benutzer dazu verleiten, eine Datei herunterzuladen und auszuführen – unwissend installiert er so eine Waledac-Instanz. Nach der Installation fragt der Bot den Botmaster nach Befehlen und führt diese aus.

heruntergeladen werden. Auch die Nachrichtenmeldungen lassen sich angeblich erst mit der neusten Version des Adobe Flash Player betrachten, der ebenfalls heruntergeladen werden muss. Beides ist eine Benutzerinteraktion, jedoch lädt sich der Benutzer anstatt der versprochenen Inhalte eine Instanz von Waledac auf den eigenen Rechner, so dass beim Öffnen der Dateien der Bot installiert wird (Code-Ausführung).

Nach der Installation läuft die Verbreitungsroutine vereinfacht beschrieben folgendermaßen ab: Waledac verbindet sich zu dem Botmaster und fragt nach Befehlen, die ausgeführt werden sollen. Sind keine Befehle vorhanden, pausiert die Instanz für 30 Sekunden und fragt dann erneut nach. Erhält die Waledac-Instanz Befehle vom Botmaster, führt sie diese aus und fragt dann ein weiteres Mal nach neuen Befehlen. Sollte dies ein Befehl zur Weiterverbreitung sein, werden neue E-Mails generiert und an weitere Empfänger versandt. Das Ausführen der Befehle, die nicht zur Weiterverbreitung dienen, ist in der Abbildung durch einen gestrichelten Pfeil symbolisiert, da dies nicht direkt zur Verbreitungsroutine von Waledac gehört, sondern eher einer Schadroutine zuzuordnen ist.

3.6. Zusammenfassung

Dieses Kapitel liefert die Herleitung der Definition des für diese Arbeit zentralen Begriffs *Social Malcode*. Da die Intention eines Menschen bezüglich einer Handlung nur sehr schwer festzustellen ist und die Intention beim Social Engineering eine wichtige Rolle spielt, lautet eine erste Definition folgendermaßen: Social Malcode ist Code, der einem Angreifer nicht-autorisierten Zugriff auf eine oder mehrere Ressourcen eines Opfers ermöglicht und der zur Weiterverbreitung Benutzerinteraktion erfordert. Die

erste Bedingung drückt den schadhaften Aspekt von Social Malcode aus und die zweite Bedingung bewirkt, dass nun die Intention einer Person bezüglich der Verbreitung von Social Malcode unbedeutend ist.

Für diese Definition werden anschließend einige Beispiele und Gegenbeispiele aufgezeigt. Dabei wird ersichtlich, dass bei den Beispielen für Social Malcode stets ein Eingreifen einer Person, wie zum Beispiel das Herunterladen und Ausführen eines E-Mail-Anhangs, für die Verbreitung der Schadprogramme notwendig ist. Genau diese Handlung entspricht der geforderten Benutzerinteraktion der Definition. Die Gegenbeispiele hingegen bestehen ausschließlich aus Schadprogrammen, die sich autonom verbreiten – in den meisten Fällen durch das Ausnutzen einer Schwachstelle. Diese Schadprogramme verbreiten sich vollkommen unabhängig von einer menschlichen Interaktion und fallen somit nicht unter den Begriff Social Malcode.

Eine Diskussion zeigt noch bestehende Probleme und Schwächen der Definition auf. So ist Phishing intuitiv als Paradebeispiel eines Social-Engineering-Angriffs anzusehen. Jedoch verbreitet sich der Code einer Phishing-Seite nicht und passt somit auch nicht auf die erste Version der Defintion von Social Malcode.

Aus diesem Grund wird, nach der Definition der Begriffe *Schadcode* und *Verbreitungsroutine*, die für die vorliegende Arbeit endgültige Definition von Social Malcode präsentiert: Als Social Malcode wird Schadcode bezeichnet, der sich entweder nicht verbreitet oder der sich verbreitet und mindestens eine Verbreitungsroutine enthält, in der sich eine Provokation/Herausforderung, eine Benutzerinteraktion und eine Code-Ausführung kausal bedingen. Durch die geforderte Sequenz von Zustandsübergängen sind immer noch all jene Schadprogramme in der Definition enthalten, bei der die Verbreitung auf einer Benutzerinteraktion beruht. Zusätzlich fällt nun aber auch Schadcode unter den Begriff Social Malcode, der sich nicht verbreitet – wie etwa eine Phishing-Seite.

KAPITEL IV

Bestimmung von Parametern und zusätzliche Datenquellen

Das Verhalten des Menschen nimmt definitionsgemäß wesentlichen Einfluss auf die Verbreitung von Social Malcode (siehe Abschnitt 3.5). Um die Verbreitung von Social Malcode zu modellieren, muss demnach auch das Verhalten der in den Verbreitungsprozess involvierten Benutzer modelliert werden. Die Fragestellung, die sich somit ergibt, ist:

> **Wie verhält sich ein Benutzer in Situationen, die möglicherweise zu einer Infektion durch Social Malcode führen?**

In diesem und den folgenden Abschnitten bezeichnet der Begriff *Benutzer* ein potentielles Opfer von Social Malcode – etwa einen Empfänger einer E-Mail oder einen Teilnehmer einer Konversation per Instant Messaging System.

Die in der Fragestellung erwähnten Situationen, die möglicherweise zu einer Infektion eines Benutzerrechners mit Social Malcode führen, sind sehr vielseitig und können auch ineinander verschachtelt sein: Erhält beispielsweise ein Benutzer eine E-Mail mit einem Dateianhang, so hängt eine Infektion des Benutzerrechners davon ab, ob der Benutzer den Dateianhang herunterlädt. Falls er dies getan hat, ist eine weitere Bedingung für eine Infektion, dass der Benutzer diesen Dateianhang auch ausführt. Statt eines Dateianhangs kann sich im Text der E-Mail aber auch ein Verweis auf eine Internetseite befinden, die bösartig präpariert ist und deren Aufruf mit einem verwundbaren Browser bereits ausreicht, um sich zu infizieren. Eine andere Situation, in der eine Infektion über einen Verweis stattfinden kann, ist die Kommunikation mittels eines IM-Systems. Erhält ein Benutzer während einer Konversation eine Nachricht, die einen Verweis auf eine bösartig präparierte Internetseite enthält, infiziert er sich (wieder einen verwundbaren Browser vorausgesetzt), sobald er dem Verweis folgt und somit diese Seite besucht.

Die Güte bzw. die Qualität einer Modellierung von Social Malcode und dessen Verbreitung hängt demnach wesentlich von einer verlässlichen und wahrheitsgetreuen Beantwortung der oben gestellten Frage ab. Eine Möglichkeit, Antworten auf diese Frage zu erhalten, besteht darin, Umfragen oder Interviews durchzuführen. Dieser Ansatz

birgt jedoch inhärente und nicht vernachlässigbare Nachteile, die kurz in Abschnitt 4.1 diskutiert werden.

Stattdessen wurden mehrere Experimente durchgeführt, um die obige Fragestellung zu beantworten. Abschnitt 4.2 beschreibt Experimente, die in Zusammenarbeit mit einem Unternehmen aus der IT-Sicherheitsbranche durchgeführt wurden. Dabei wurden mehrere E-Mails an die Mitarbeiter verschiedener Unternehmen gesendet, um deren Verhalten auf diese E-Mails zu messen.

Ein weiterer wesentlicher Faktor bei der Verbreitung von Social Malcode spielen die Reaktionszeiten der Benutzer. Zum Beispiel ist die Zeitspanne zwischen dem Eintreffen und dem Lesen von E-Mails entscheidend für die gesamte Verbreitungsgeschwindigkeit von Schadprogrammen, die sich per E-Mail verbreiten. In Abschnitt 4.3 stellen wir ein Skript vor, mit dessen Hilfe wir diese Zeitspannen direkt am Mailserver der Universität Mannheim gemessen haben. Ebenfalls werden in diesem Abschnitt Ergebnisse von Analysen auf den ermittelten Reaktionszeiten präsentiert.

Sobald ein System von Social Malcode befallen wird, bezeichnen wir dieses System als infiziert. Jedoch ist es in der Realität nicht so, dass ein System für immer infiziert ist, sondern es kann wieder von dem Schädling befreit beziehungsweise gesäubert werden. Abschnitt 4.4 setzt sich mit dem Säuberungsverhalten der Benutzer auseinander und versucht aus einem Datensatz, der aus mitgeschnittenen Keylogger-Daten besteht, die mittleren Reparaturzeiten der infizierten Systeme zu ermitteln.

Bevor dieses Kapitel mit einer kurzen Zusammenfassung in Abschnitt 4.6 abgeschlossen wird, geht Abschnitt 4.5 näher auf einen Spam-Korpus ein. Analysen auf dessen Spam-Mails sollen die Verhaltens- und Vorgehensweisen der Angreifer näher beleuchten. Neben dem Verhalten der potentiellen Opfer ist das Verhalten der Angreifer ebenfalls ein wichtiger Bestandteil der Verbreitung von Social Malcode. Etwa ist der verwendete Angriffsvektor (Dateianhang, Verweis auf eine Datei usw.) oder auch die Häufigkeit des Versands einer E-Mail mit Social Malcode im Anhang entscheidend für dessen Verbreitung.

Wie Tabelle 2.2 auf Seite 29 zeigt, existieren viele verschiedene Verbreitungswege von Schadprogrammen, bei denen ein menschliches Eingreifen notwendig ist. In diesem und den folgenden Kapiteln beschränken wir uns jedoch ausschließlich auf eine Verbreitung per E-Mail. Dies hat vor allem zwei Gründe: Zum einen ist dies durch den zeitlichen und den Seitenumfang betreffenden Rahmen dieser Arbeit bedingt. Eine genaue Analyse aller möglichen Verbreitungswege würde den Umfang dieser Arbeit sprengen. Zum anderen ist, obwohl die Kommunikation über soziale Medien immer stärker zunimmt, die Kommunikation per E-Mail immer noch am weitesten verbreitet [Lin11].

4.1. Nachteile von Umfragen und Interviews

Eine Möglichkeit, das Verhalten der Benutzer in Situationen zu messen, die möglicherweise zu einer Infektion durch Social Malcode führen, besteht in der Durchführung von Umfragen und Interviews. Sie sind das klassische Instrument der empirischen Sozialwissenschaften zum Quantifizieren von Verhaltensweisen. Die Bewertung des Wahrheitsgehaltes der erhaltenen Antworten und die Durchführung und Planung sol-

cher Umfragen sind aber mit einigen Problemen behaftet, die in diesem Abschnitt kurz beleuchtet werden sollen.

Das Ziel einer Umfrage oder eines Interviews ist es, den *wahren Wert* bezüglich einer Fragestellung zu ermitteln. Der wahre Wert ist die tatsächliche Ausprägung der Information, die man bei einem Befragten messen möchte – also der Wert, der die Realität widerspiegelt. Jedoch müssen in der Planung und Durchführung einer Umfrage oder eines Interviews vielerlei Merkmale und Effekte berücksichtigt werden, die einen verfälschenden Einfluss auf die Messung des wahren Wertes ausüben können. Die Effekte mit dem größten Einfluss werden in den nächsten Absätzen kurz beschrieben. „Die Frage, welche Merkmale eine Messung des ‚wahren Wertes' ermöglichen und welche Merkmale einen verfälschenden Einfluß ausüben, ist, wenn überhaupt, schwer beantwortbar" [Str94]. Es ist also schwer festzustellen, ob eine bei mehrmaligen Befragungen unterschiedliche Antwort, den verfälschenden Effekten zuzurechnen ist, oder ob sich im Zeitraum zwischen den Befragungen der wahre Wert verändert hat. Des Weiteren kann es sein, dass der wahre Wert eines Befragten bezüglich einer Frage gar nicht existiert – in diesen Fall können die Antworten der Befragten eher zufallsgenerierte Reaktionen darstellen. Diese Fehlerquelle wird in der Literatur als *non-attitudes* bezeichnet [Con70].

Einer der größten Nachteile von Interviews besteht in der *Sozialen Erwünschtheit*. „Sie bewirkt, daß der Befragte nicht die tatsächliche Ausprägung des subjektiven Merkmals berichtet, sondern die Antwort so gestaltet, daß mit ihr soziale Anerkennung erzielt oder zumindest soziale Ablehnung vermieden wird" [Str94]. Angenommen man fragt eine Person, ob sie bereits einmal einem Verweis einer Spam-Mail gefolgt ist, um zu schauen, was sich hinter dem Verweis der Spam-Mail verbirgt, die beispielsweise Potenzmittel bewirbt. In diesem Fall ist es wahrscheinlich, dass der Befragte dies nicht zugeben würde, auch wenn er genau weiß, dass er schon einmal solch einem Verweis gefolgt ist. Der Grund dafür ist, dass der Befragte einen schlechten Eindruck beim Interviewer vermeiden möchte, da er genau weiß, dass dieses Verhalten eben nicht sozial angesehen ist.

Ein Problem von Umfragen stellt unter anderem der sogenannte *Reihenfolgeneffekt* dar, der sich auf die Anordnung der Fragen eines Fragebogens bezieht. „Da eine wissenschaftliche Befragung in der Regel mehr als nur eine Frage enthält, müssen die Fragen in einer bestimmten Reihenfolge angeordnet werden". Vor allem die Fragen, die unmittelbar vor der aktuell behandelten Frage stehen, haben einen wesentlichen Einfluss auf die Antwort der aktuellen Frage. Zudem kann die Antwort des Befragten sogar unwissentlich durch die vorhergehenden Fragen vom Fragebogenkonstrukteur gelenkt werden. Erhält der Befragte beispielsweise durch die vorhergehende Frage das Gefühl, er habe ein hohes Sicherheitsbewusstsein bezüglich Verbreitungswegen von Malware, so wird er wahrscheinlich die Frage nach der Häufigkeit, mit der er Dateianhänge ausführt, anders beantworten, als wenn er dieses Gefühl nicht hat. Von Umfrageforschern wird dieser Effekt als die wichtigste Störquelle bei der Fragebogenkonstruktion angesehen [Bra83].

Ein anderes Problem bei Umfragen besteht darin, ob man die Antwortmöglichkeiten mit angibt (*geschlossene Fragen*) oder nicht (*offene Fragen*). Bei angegebenen Antwortmöglichkeiten entsteht der Vorteil, dass die Umfrage statistisch besser auswertbar ist und die Antworten miteinander verglichen werden können. Leider bewirken vor-

gegebene Antwortkategorien aber auch, „dass Inhalte mit in die Antwort einbezogen werden, die vom Befragten im offenen Format, aus welchen Gründen auch immer, vernachlässigt werden“ [SS87].

Ebenfalls wirkt sich die Formulierung einer Frage auf deren Antwort aus. Oftmals kann bereits eine reine Umformulierung einer Frage, die den Inhalt der Frage unberührt lässt, zu unterschiedlichen Ergebnissen führen [Dav76]. „Einer der am häufigsten untersuchten Aspekte der *Frageformulierung* ist die Asymmetrie bei der Verwendung von Antonymen. Am bekanntesten sind die Arbeiten zur Verwendung der Antonyme ‚verbieten‘ vs. ‚erlauben‘. Wenn ‚verbieten‘ das Gegenteil von ‚erlauben‘ ist und dieser logische Aspekt die Antwort bestimmt, dann sollte die Verteilung von Ja- und Nein-Antworten spiegelbildlich für die beiden Formulierungen auftreten. [...] Dies ist jedoch nicht der Fall“ [Str94]. Im Kontext von Social Malcode würde dies etwa bedeuten, dass die beiden Phishing-bezogenen Fragen

- „Achten Sie bei einem Verweis auf die Zieldomain des Verweises?“ und
- „Ignorieren Sie bei einem Verweis dessen Zieldomain?“

nicht die genau entgegengesetzten Antworten liefern würden.

Aber nicht nur die Verwendung von Antonymen kann bei der Formulierung der Fragen Probleme bereiten. Ein Befragter kann eine Frage einfach missverstehen, da sie unklar formuliert ist. Der Fragebogenkonstrukteur weiß genau, was er fragen möchte und sieht die Probleme im Vorfeld vielleicht nicht, die zu Missverständnissen führen könnten – er kann sich nicht immer in die Lage eines Befragten versetzen und nachvollziehen, was dieser denkt, wenn er eine Frage zum ersten Mal liest.

Ein weiterer Aspekt, der gegen die Verwendung von Umfragen und Interviews spricht, ist die klinische Situation, die bei deren Durchführung entsteht. Der Befragte weiß natürlich während der Befragung, dass der Fragende einen bestimmten Sachverhalt untersuchen möchte. Dieser Umstand kann sich unmittelbar in seinen Antworten und seinem Verhalten widerspiegeln. Aus diesem Grund sollte man bei der Fragebogenkonstruktion auch darauf achten, dass der eigentlich beabsichtigte Zweck des Fragebogens (das Erkunden eines bestimmten Sachverhalts) nicht sofort ersichtlich ist [Str94].

4.2. Spam-Experimente

Um die erörterten Nachteile eines Interviews oder einer Umfrage zu umgehen und die anfangs angesprochene Fragestellung dennoch zu beantworten, wurden in Zusammenarbeit mit der IT-Sicherheitsfirma *8com GmbH & Co. KG* [8co11] im Zeitraum von Herbst 2009 bis Herbst 2010 mehrere Experimente durchgeführt. Die Experimente hatten das Ziel, das Verhalten von Benutzern beim Umgang mit E-Mails zu untersuchen. Die *8com GmbH & Co. KG* bietet für ihre Kunden, Unternehmen aus dem Mittelstand, unter anderem Awareness-Kampagnen an, durch die das Sicherheitsbewusstsein der Mitarbeiter geschult werden soll. Im Rahmen dieser Kampagnen wurde untersucht, wie die Mitarbeiter der Kunden auf unaufgefordert zugeschickte E-Mails (Spam-Mails) reagieren – also welches Verhalten sie in solch einer Situation zeigen.

Für diesen Zweck wurden mehrere Spam-Experimente durchgeführt, die durch den Text aus Anhang A beworben wurden.

Der generelle Ablauf dieser *Spam-Experimente* ist in Abbildung 4.1 dargestellt. Ein Mailserver und ein Webserver der *8com GmbH & Co. KG*, im Folgenden *Spam-Server* und *Feedback-Server* genannt, wurden dazu verwendet, das genaue Verhalten der Mitarbeiter festzustellen und aufzuzeichnen. Jedes Spam-Experiment beginnt damit, dass ausgehend vom Spam-Server den Mitarbeitern des Kunden Spam-Mails zugeschickt werden, die bei jeder Durchführung des Experiments verschiedene Methoden einer Rechnerinfektion per E-Mail simulieren (siehe Abschnitt 2.2.3). Von dem Mailserver des Kunden gelangen die E-Mails zu den Mitarbeitern[1]. Der Weg der E-Mails ist in Abbildung 4.1 durch blaue, gestrichelte Pfeile dargestellt. Sobald ein Mitarbeiter eine Spam-Mail liest, kann er, je nach durchgeführtem Experiment, unterschiedlich auf die E-Mail reagieren und etwa einem enthaltenen Verweis folgen oder einen Dateianhang herunterladen. Wenn dies geschieht, wird über Port 80, dies ist der Standardport für das *Hypertext Transfer Protocol* (HTTP), ein Rückkanal zum Feedback-Server aufgebaut, um die Reaktion des Mitarbeiters zu protokollieren. Die Verwendung von HTTP für den Rückkanal hat den Vorteil, dass ausgehende Verbindungen zu Port 80 meist nicht durch Firewalls blockiert werden, damit Mitarbeiter im Internet recherchieren können. Der Rückkanal zum Feedback-Server wird in der Abbildung durch die grünen, durchgehenden Pfeile illustriert.

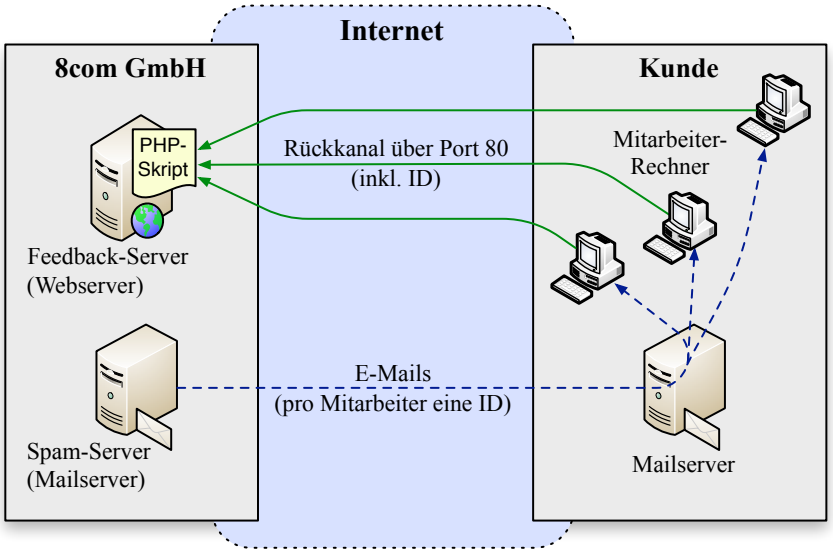


Abbildung 4.1.: Genereller Ablauf der Spam-Experimente
Ausgehend vom Spam-Server werden Spam-Mails über den Mailserver eines Kunden an dessen Mitarbeitern gesendet (blaue, gestrichelte Pfeile). Eine mögliche Reaktion eines Empfängers wird über Port 80 an den Feedback-Server übertragen (grüne Pfeile).

[1] Genau genommen ist dies nur der Fall, wenn ein Mitarbeiter seine E-Mails über POP3 abruft. Bei einem Zugriff mittels IMAP bleiben die E-Mails normalerweise auf dem Mailserver – dies wird aber, da es für die Experimente nicht relevant ist und zugunsten einer übersichtlicheren Darstellung, vernachlässigt.

Bei jeder Durchführung des Experiments wird jedem Mitarbeiter eine eindeutige Identifikationsnummer (ID) zugewiesen. Wird während des Experiments eine E-Mail an einen Mitarbeiter geschickt, so ist dessen ID in der E-Mail enthalten – beispielsweise innerhalb eines Verweises. Zudem wird diese ID bei einer möglichen Reaktion des Mitarbeiters ebenfalls über den Rückkanal zum Feedback-Server übertragen. Dort läuft ein PHP-Skript, dem die ID übergeben wird und das die Reaktion des Mitarbeiters zusammen mit dem aktuellen Zeitpunkt, dessen ID und der IP-Adresse aufzeichnet. Auf diese Weise lassen sich die aufgezeichneten Reaktionen zweier Mitarbeiter voneinander unterscheiden.

Aus Gründen des Datenschutzes wurde darauf verzichtet, die Zuordnung zwischen ID und E-Mail-Adresse eines Empfängers nach dem Versand der E-Mails aufrechtzuerhalten. Somit ist zu keinem Zeitpunkt der Experimente eine Zuordnung zwischen einer realen Person und einer aufgezeichneten Reaktion möglich.

Bei jeder Durchführung des Experiments wurden im Auftrag eines Kunden verschiedene E-Mails verschickt. Jede dieser E-Mails simuliert eine mögliche Vorgehensweise eines Angreifers, der versucht Zugriff auf das Firmennetz des Kunden zu erhalten, indem er den Mitarbeitern E-Mails sendet. Diese E-Mails enthalten entweder einen Verweis oder einen Dateianhang, die ihm bei einer entsprechenden Reaktion beziehungsweise einem für ihn günstigen Verhalten der Empfänger diesen Zugriff ermöglichen. Insgesamt wurden vier verschiedene „Arten“ von E-Mails während der Experimente verwendet:

1. E-Mails mit einem Verweis auf eine Internetseite
2. E-Mails mit einem nicht ausführbaren Dateianhang
3. E-Mails mit einem Verweis auf eine ausführbare Datei im Internet
4. Phishing-Mails, die einen Empfänger dazu auffordern, sein firmeninternes Passwort auf einer Internetseite zu ändern, auf die ein Verweis innerhalb der Phishing-Mail zeigt

Auf ein Spam-Experiment mit ausführbaren Dateianhängen wurde verzichtet, da mittlerweile die meisten Spam-Filter so konfiguriert sind, dass sie solche E-Mails sofort als verdächtig markieren oder diese filtern, so dass sie nicht mehr im Posteingang des Empfängers eintreffen. Die folgenden Abschnitte liefern eine genaue Beschreibung der verwendeten E-Mails und eine Auswertung der Experimente. Der in diesen Abschnitten verwendete Begriff eines *E-Mail-Laufs* bezeichnet dabei eine Durchführung des Spam-Experiments bezüglich einer der vier oben aufgelisteten E-Mails.

4.2.1. Verweis auf eine Internetseite

In diesem E-Mail-Lauf wird eine E-Mail verschickt, die einen Verweis auf eine Internetseite enthält. Es werden dabei keinerlei Maßnahmen durchgeführt, um den Empfänger zum Folgen des Verweises zu verleiten – etwa das Fälschen der Absenderadresse oder eine persönliche Anrede des Empfängers. Lediglich der E-Mail-Text ist so gestaltet, dass er einen möglichst großen Empfängerkreis anspricht. Bis Ende Juli 2010 war das Thema der E-Mails eine fingierte Nachrichtenmeldung zu der *Schweizer Steuer-CD*, die im Februar 2010 mehreren Landesregierungen der Bundesrepublik Deutschland zum Kauf angeboten wurde [BBvH+10]. Eine solche E-Mail zeigt Abbildung 4.2.

Der Verweis in der E-Mail, zeigt auf eine Internetseite, auf der angeblich die Namen vier hochrangiger, in den Steuerskandal verwickelter Politiker veröffentlicht sind.

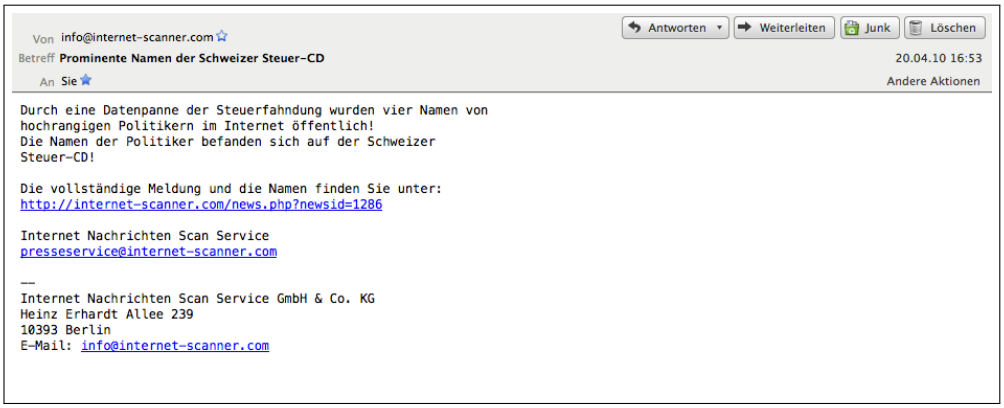


Abbildung 4.2.: Erste Spam-Mail mit dem Angriffsvektor *Verweis*
In dieser Spam-Mail zu dem Thema *Schweizer Steuer-CD* soll ein Empfänger dazu verleitet werden, einem Verweis zu folgen.

Ab August 2010 wurde diese E-Mail durch eine E-Mail zum Thema *Google Street View* ersetzt, da das Thema der Schweizer Steuer-CD als nicht mehr aktuell genug betrachtet wurde. Diese zweite Version der E-Mails beinhaltet auch wieder eine fingierte Nachrichtenmeldung, nach welcher der Kartografie-Dienst Street View des Unternehmens Google entgegen der ursprünglichen Ankündigung nicht nur in den 20 größten deutschen Städten starten wird, sondern ebenfalls in mehreren Kleinstädten Deutschlands [BEFF10]. Ein Beispiel solch einer E-Mail befindet sich in Abbildung 4.3.

Abbildung 4.3.: Zweite Spam-Mail mit dem Angriffsvektor *Verweis*
In dieser Spam-Mail zu dem Thema *Google Street View* soll ein Empfänger dazu verleitet werden, einem Verweis zu folgen.

Die E-Mails dieses E-Mail-Laufs simulieren einfache Spam-Mails, die üblicherweise nicht personalisiert sind und die an einen großen Empfängerkreis verschickt werden. Die Gefahr, die von solch einer Art von E-Mail ausgeht, liegt vor allem in

dem darin enthaltenen Verweis: Durch eine *Drive-By-Download* genannte Technik ist es einem Angreifer möglich, die hinter dem Verweis befindliche Internetseite so zu gestalten, dass lediglich der Besuch dieser Seite ausreicht, um den Client-Rechner unbemerkt mit Schadprogrammen zu infizieren [PMM+07].

Bei den Spam-Experimenten befand sich hinter dem Verweis eine harmlose Internetseite, die einen HTTP-Fehler 504 (*Gateway Timeout*) nachahmt. Diese Seite enthält ein PHP-Skript, das die ID des Empfängers entgegen nimmt, diese abspeichert und nach einer Verzögerung von wenigen Sekunden dem Benutzer die HTTP-Fehlermeldung 504 präsentiert. In den beiden Abbildungen 4.2 und 4.3 heißt das PHP-Skript *news.php* beziehungsweise *nachricht.php* und die Variablen, welche die Empfänger-ID entgegennehmen, heißen in den beiden Beispielen *newsid* und *msg*. Durch diese Namen soll ein Empfänger die ID, die eigentlich seine eigene Kennzeichnung ist, als Nachrichten-Nummer/ID interpretieren.

4.2.2. Nicht ausführbarer Dateianhang

In diesem E-Mail-Lauf wurde eine E-Mail mit einem nicht ausführbaren Dateianhang verschickt. Wie bereits bei den E-Mails des ersten E-Mail-Laufs wurden auch bei den E-Mails dieses Laufs weder die Empfänger persönlich angesprochen, noch wurde die Absenderadresse gefälscht. Lediglich der E-Mail-Text ist wieder so gestaltet, dass er einen möglichst großen Empfängerkreis anspricht. Die E-Mails behandeln das Thema *Benzinpreise*: Im März 2010 schlug der damalige Bundespräsident Horst Köhler vor, das Wachstum der deutschen Wirtschaft eventuell durch höhere Benzinpreise zu beschleunigen [Spi10]. Der Text der E-Mails ist eine Nachrichtenmeldung, laut der die Bundesregierung den Vorschlag Köhlers aufnehmen möchte und finanziell schlechter gestellte Bürger zu Ungunsten der Steuerzahler durch Subventionen unterstützen möchte. Angeblich befinden sich weitere Maßnahmen der Bundesregierung zur Wirtschaftsförderung in dem angehängten Word-Dokument, das die Empfänger herunterladen und öffnen sollen. Ein Beispiel der in diesem Lauf verschickten E-Mails befindet sich in Abbildung 4.4.

Wenn ein Empfänger den Anhang herunterlädt und ausführt, ermöglicht dies einem Angreifer im schlimmsten Fall das Ausführen von fremdem Code auf dem Rechner des Empfängers. Ein Dokument kann beispielsweise so präpariert werden, dass beim Öffnen des Dokuments in einer verwundbaren Anwendung deren Schwachstellen ausgenutzt werden, um auf diese Weise beliebigen Code auszuführen [EWH09]. Für einen Angreifer haben E-Mails mit nicht ausführbaren Dateianhängen zudem den Vorteil, dass diese von Spam-Filtern seltener als verdächtig markiert werden als E-Mails mit einem ausführbaren Dateianhang.

Für die Spam-Experimente wurden als Anhänge Word-Dokumente und Excel-Tabellen verwendet, die sich per VBA-Makro (*Visual Basic for Applications*) zum Feedback-Server verbinden. Der Inhalt der Dokumente besteht aus einem Hinweis, dass der in der E-Mail angedeutete Inhalt des Dokuments nur mit aktivierten Makros angezeigt werden kann. Ein Beispiel eines VBA-Makros aus einer Excel-Tabelle ist in Code-Block 4.1 aufgelistet. In diesem Makro sind drei Funktionen implementiert: *GetLastPos* (Zeile 1), welche die Position des letzten Vorkommens eines Zeichens in einer Zeichenkette zurückgibt, und in Zeile 21 die Funktion *FileNameToID*, die aus

Abbildung 4.4.: Spam-Mail mit dem Angriffsvektor *Dateianhang*
In dieser Spam-Mail soll ein Empfänger dazu gebracht werden, den Dateianhang herunterzuladen und diesen zu öffnen.

einem Dateinamen die ID des Empfängers extrahiert. In diesem E-Mail-Lauf, werden die Dateinamen der Anhänge genutzt, um die einem Empfänger zugewiesene ID zu diesem zu übertragen. Die Dateinamen der Anhänge sind nach dem Muster

`<beliebiger Text>-<ID>.<Dateierweiterung>`

aufgebaut. In der E-Mail aus Abbildung 4.4 wurde eine Datei mit dem Dateinamen *Massnahmenkatalog-3972.doc* verschickt, somit hat zum Beispiel der Empfänger genau dieser E-Mail die ID 3972. Die letzte Funktion des Makros ist die Funktion *Workbook_Open*. Dies ist eine von *Microsoft* ab der Office Version 97 eingeführte (Event-)Funktion, die immer dann ausgeführt wird, wenn die Excel-Tabelle geöffnet wird. Für die Spam-Experimente wurde diese Funktion so belegt, dass sie mit Hilfe der anderen beiden Funktionen beim Öffnen der Tabelle, die ID des Empfängers aus dem Dateinamen der Excel-Tabelle extrahiert und diese über den Standardbrowser des Empfängers an den Feedback-Server überträgt. Dort liegt wieder ein PHP-Skript, das die ID als Wert eines Übergabeparameters entgegennimmt. In Zeile 36 des Code-Blocks 4.1 lässt sich erkennen, dass der Name des PHP-Skriptes *aktuelles.php* lautet sowie der dazugehörige Name des Parameters *msg*.

```
1  Function GetLastPos(strSearch As String, char As String) As Long
2    Dim pos As Long
3    Dim i As Long
4    For i = 0 To Len(strSearch) - 1
5      pos = Len(strSearch) - i
6      If Mid$(strSearch, pos, 1) = char Then
7        Exit For
8      End If
9    Next i
10   If pos > 1 Then
11     GetLastPos = pos
12   Else
13     If Mid$(strSearch, 1, 1) = char Then
```

```
14        GetLastPos = 1
15      Else
16        GetLastPos = 0
17      End If
18    End If
19  End Function
20
21  Function FilenameToID(filename As String) As String
22    Dim Result As String
23    Result = filename
24    posDot = GetLastPos(Result, ".")
25    If posDot > 0 Then
26      Result = Left$(Result, posDot - 1)
27    End If
28    posDash = GetLastPos(Result, "-")
29    If posDash > 0 Then
30      Result = Right$(Result, Len(Result) - posDash)
31    End If
32    FilenameToID = Result
33  End Function
34
35  Private Sub Workbook_Open()
36    URL = "http://internet-scanner.com/aktuelles.php?msg=" & FilenameToID(
          Application.ActiveWorkbook.Name)
37    CreateObject("Shell.Application").ShellExecute URL
38  End Sub
```

Code-Block 4.1: Rückkanal zum Feedback-Server mittels VBA-Makro

Des Weiteren ist der in den Dokumenten enthaltene VBA-Code zertifiziert, damit die Empfänger beim Öffnen der Dokumente möglichst keinen Hinweis auf das Makro erhalten, um so die Unauffälligkeit der Dokumente zu erhöhen. Nicht zertifizierter Code verursacht eine Warnmeldung beim Öffnen der Dokumente, die den Benutzer darauf hinweist, dass sich ein Makro in dem Dokument befindet.

4.2.3. Verweis auf ausführbare Datei im Internet

In diesem E-Mail-Lauf wurden E-Mails verschickt, die einen Verweis auf eine ausführbare Datei im Internet enthalten. Diese Vorgehensweise, Zugriff auf das Firmennetz zu erhalten, ist gewissermaßen eine Mischung aus den Vorgehensweisen der beiden ersten E-Mail-Läufe. Beim Versenden einer E-Mail mit ausführbarem Dateianhang hat ein Angreifer oftmals das Problem, dass dieser Anhang herausgefiltert wird, bevor die E-Mail im Posteingang des Empfängers erscheint. Durch einen Verweis auf die ausführbare Datei im Text der E-Mail, kann ein Angreifer dieses Problem umgehen, da sich die ausführbare Datei auf einem Rechner im Internet befindet und nicht in der E-Mail an sich.

Die E-Mails dieses Laufs sind so gestaltet, dass sie optisch einer E-Mail der Firma *Microsoft* ähneln – ein Beispiel zeigt Abbildung 4.5. Inhaltlich stellt diese E-Mail eine Sicherheitsempfehlung dar, in der einem Empfänger mehrere Schritte präsentiert werden, um den Browser *Internet Explorer* gegen eine neu entdeckte Schwachstelle abzusichern. Dabei handelt es sich jedoch um eine fiktive Schwachstelle, die real nicht existiert. Die angegebenen Schritte sind absichtlich sehr abstrakt und unklar formuliert, so dass der Empfänger dazu verleitet werden soll, einen hinter einem Verweis liegenden Sicherheitspatch herunterzuladen und auszuführen.

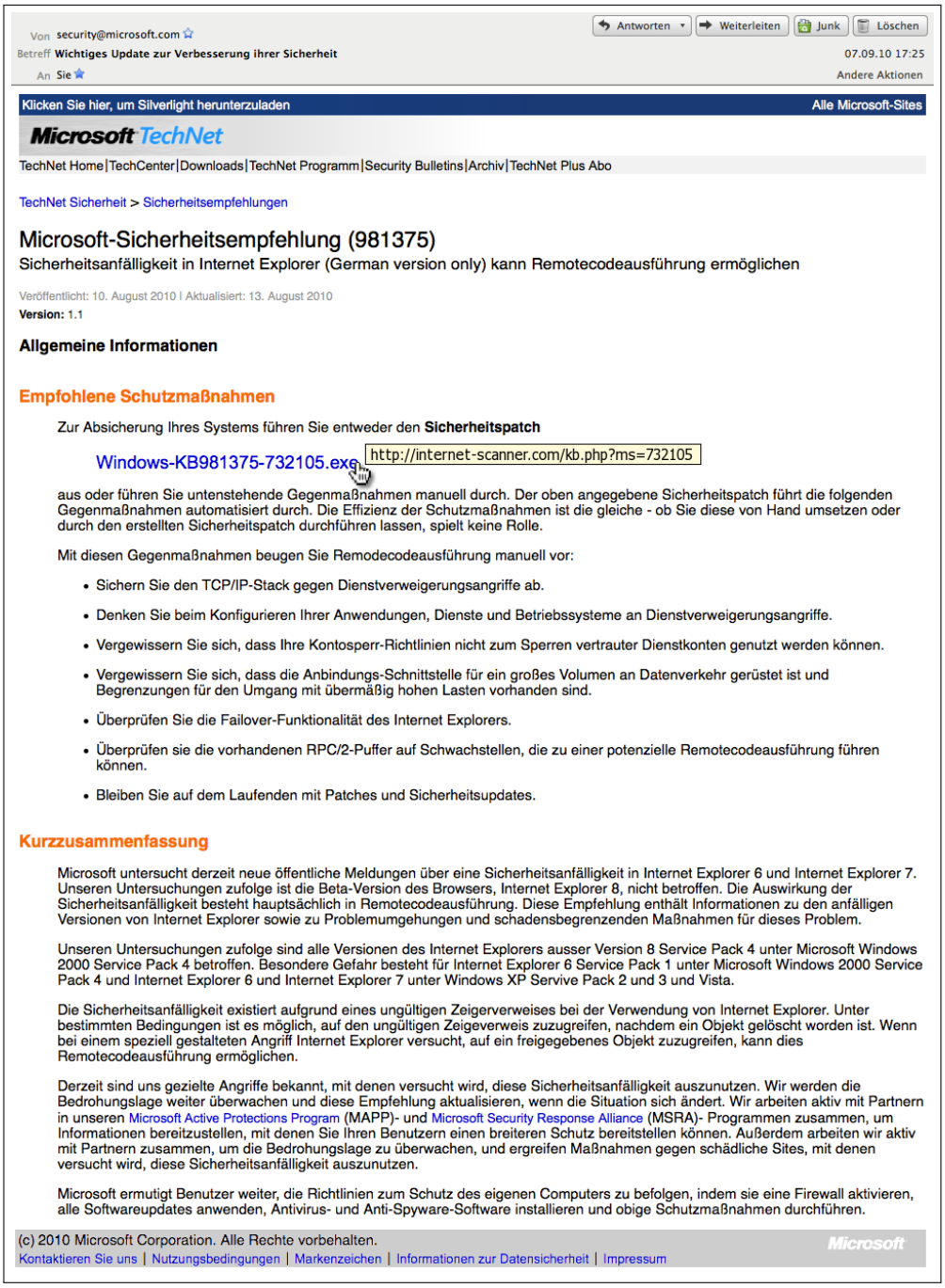


Abbildung 4.5.: Spam-Mail mit dem Angriffsvektor *Verweis auf ausführbare Datei*
Diese angeblich von *Microsoft* stammende E-Mail informiert den Empfänger über eine neue Schwachstelle, die jedoch nur für die Spam-Experimente ausgedacht wurde und real nicht existiert. *Hinweis: Der Mauscursor und das Ziel des Verweises wurden zum besseren Verständnis nachträglich eingefügt.*

Der Patch ist eine ausführbare Datei und führt vermeintlich die genannten Absicherungsschritte voll automatisch durch. Zudem wurde beim Verfassen der E-Mail die Absenderadresse gefälscht, um den Inhalt der E-Mail glaubwürdiger erscheinen zu lassen: Als Absenderadresse wurde *security@microsoft.com* verwendet. Auf eine persönliche Anrede der Empfänger wurde verzichtet.

Zum besseren Verständnis der Abbildung sind nachträglich ein Mauscursor (Hand) und das Ziel des Verweises (gelber Kasten) in die Grafik eingearbeitet worden. Die ID eines Empfängers befindet sich bei E-Mails dieses Laufs in dem Verweis: In der Beispiel-E-Mail der Abbildung hat der Empfänger die ID 732105. Folgt er dem Verweis, wird die ID einem PHP-Skript auf dem Feedback-Server übergeben und dieses bietet dem Empfänger den angeblichen Sicherheitspatch zum Herunterladen an. Das Herunterladen des Patch wird zusammen mit der ID des Empfängers und dem aktuellen Zeitstempel protokolliert. Der Patch wird einem Empfänger unter dem Dateinamen *Windows-KB981375-<ID>.exe* zum Herunterladen angeboten, so dass die ID auch innerhalb des vermeintlichen Patch zur Verfügung steht.

Bei der Durchführung des Spam-Experiments wurde auf eine selbst geschriebene, ausführbare Datei verwiesen. Diese Datei präsentiert dem Benutzer nach dem Ausführen ein Fenster, das vermeintlich die in der E-Mail aufgeführten Absicherungsschritte durchführt. Ein Bildschirmfoto des Patch ist in Abbildung 4.6 zu sehen. Genau wie die E-Mails ist auch der Patch optisch so gestaltet, dass er möglichst wenig Zweifel an seiner Echtheit bei einem Empfänger hervorruft. Aus demselben Grund wird auch die Überschrift im Fensterrahmen dynamisch an das vorhandene Betriebssystem angepasst und die Titelzeile des Fensters enthält die ID des Empfängers, damit dieser Text mit dem Dateinamen übereinstimmt. Während der Ausführung werden zeitverzögert Ausgaben generiert, jedoch werden zu keinem Zeitpunkt Änderungen am System durchgeführt. Die einzige Funktionalität, neben der grafischen Ausgabe, ist, dass eine Verbindung über Port 80 zum Feedback-Server aufgebaut wird. Über diese Verbindung nimmt dort ein weiteres PHP-Skript die ID des Empfängers ein zweites Mal entgegen und protokolliert somit die Ausführung des angeblichen Sicherheitspatch.

Zusammenfassend stellt Abbildung 4.7 die insgesamt sechs Schritte des hier beschriebenen Spam-Experiments dar: Das Experiment beginnt damit, dass Spam-Mails zu den Empfängern des Kunden geschickt werden (Schritt 1). Wenn ein Empfänger dem Verweis folgt und somit den vermeintlichen Patch herunterladen möchte (Schritt 2), wird die ID des Empfängers an den Feedback-Server übertragen (Schritt 3) und dem Benutzer der Patch zum Herunterladen angeboten (Schritt 4). Durch das Ausführen des Patch (Schritt 5) wird die ID erneut zum Feedback-Server übertragen (Schritt 6) und die Ausführung protokolliert. In der Abbildung wird der Weg der E-Mails durch den gestrichelten, blauen Pfeil symbolisiert, die gepunkteten, gestrichelten, schwarzen Pfeile sind Benutzeraktionen und die durchgehenden, grünen Pfeile stellen die Verbindung zum Feedback-Server dar.

4.2.4. Phishing

In diesem E-Mail-Lauf wurden E-Mails verschickt, die eine Aufforderung an die Mitarbeiter eines Kunden enthalten, ihre firmeninternen Zugangsdaten zu ändern. Um die E-Mails glaubwürdiger zu gestalten, wurden sie jeweils im Namen eines für solch

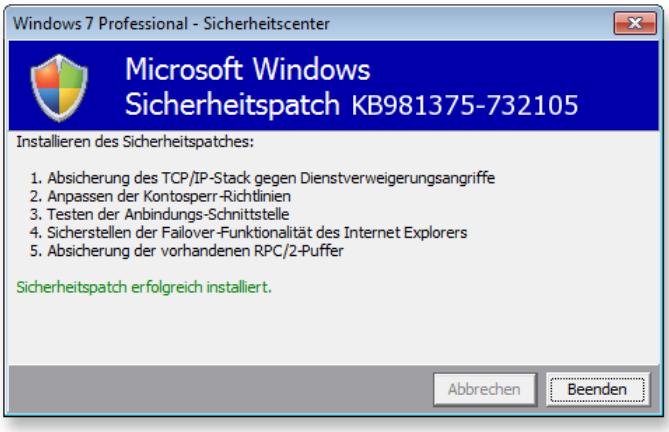


Abbildung 4.6.: Sicherheitspatch für eine angebliche Schwachstelle
Diese Abbildung zeigt die grafische Oberfläche eines vermeintlichen Sicherheitspatch. Dieser schließt angeblich die Schwachstelle, auf die in der E-Mail aus Abbildung 4.5 hingewiesen wird.

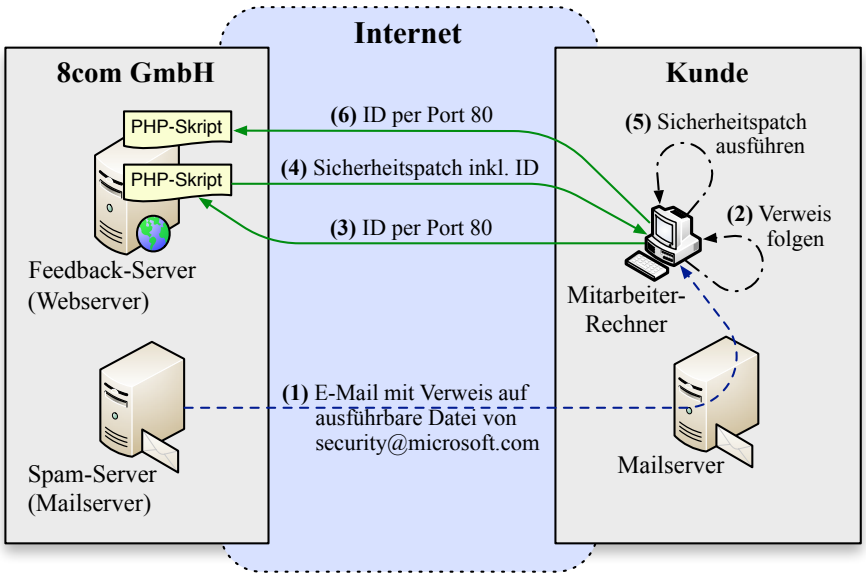


Abbildung 4.7.: Ablauf des Experiments mit einem Verweis auf eine ausführbare Datei
Vom Spam-Server werden Spam-Mails mit einem Verweis auf eine ausführbare Datei an die Mitarbeiter eines Kunden gesendet. Sowohl ein mögliches Herunterladen als auch ein Ausführen der Datei werden durch den Feedback-Server protokolliert.

administrativen Aufgaben zuständigen Mitarbeiters des Kunden versendet und dabei dessen Signatur verwendet, falls diese bekannt war. Solch ein Mitarbeiter ist beispielsweise ein Systemadministrator oder ein Mitarbeiter des für den Kunden zuständigen Rechenzentrums. Um die Glaubwürdigkeit weiter zu erhöhen, wurden zusätzlich die Empfänger mit ihrem Nachnamen angesprochen. In Abbildung 4.8 ist beispielhaft eine entsprechende E-Mail zu sehen. Aus datenschutzrechtlichen Gründen sind alle Teile der Abbildung unkenntlich gemacht, die Rückschlüsse auf den Kunden dieses konkreten Beispiels erlauben würden.

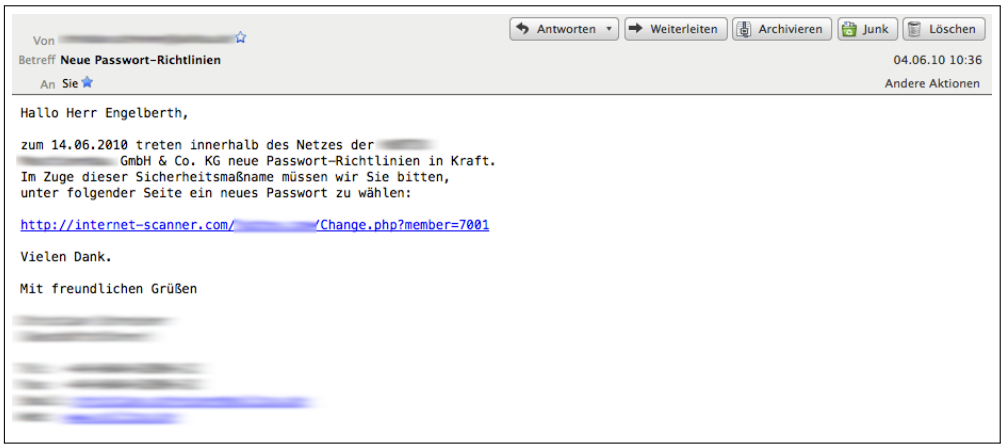


Abbildung 4.8.: Phishing-Mail mit einer Aufforderung zur Passwortänderung
Diese E-Mail fordert einen Empfänger (Kundenmitarbeiter) dazu auf, im Zuge einer neuen Sicherheitsmaßnahme sein firmeninternes Passwort zu ändern.
Hinweis: *Aus Gründen des Datenschutzes sind Teile der E-Mail unkenntlich gemacht.*

Innerhalb der E-Mails befindet sich ein Verweis auf eine Internetseite, auf der die Mitarbeiter ihren Benutzernamen, ihr altes Passwort und ein neues Passwort eingeben können. Dass der Verweis auf eine kundenfremde Domain zeigt, wurde nicht verschleiert, es wurde lediglich ein Verzeichnis angelegt, dessen Namen den Namen des Kunden enthält. Die Internetseite hinter dem Verweis besteht in allen Spam-Experimenten aus zwei Unterseiten: Eine Seite, auf der die Eingaben getätigt werden können, und eine Seite, auf der sich Tipps zur Generierung von sicheren Passwörtern und Kriterien befinden, die das vermeintlich neue Passwort erfüllen muss. Das Design und der Aufbau der Seiten ist stark an den Internetauftritt des jeweiligen Kunden angelehnt, so dass diese definitionsgemäß als Phishing-Seiten bezeichnet werden [DTH06]. Abbildung 4.9 zeigt beispielhaft die erste Unterseite eines Kunden, auf der die Mitarbeiter ihre entsprechenden Daten eingeben können. In der Abbildung sind ebenfalls alle Details unkenntlich gemacht, die Rückschlüsse auf den Kunden ermöglichen, zu dem die in der Abbildung dargestellte Phishing-Seite gehört. Nach der Eingabe der Daten und dem *Abschicken* des Formulars werden die Mitarbeiter auf die Originalseite des jeweiligen Kunden weitergeleitet.

Die zweite Unterseite teilt dem Besucher mit, dass aus sicherheitstechnischen Gründen alle 180 Tage das Passwort für seinen Account geändert werden muss. Zudem muss ein möglicherweise neues Passwort folgende Kriterien erfüllen:

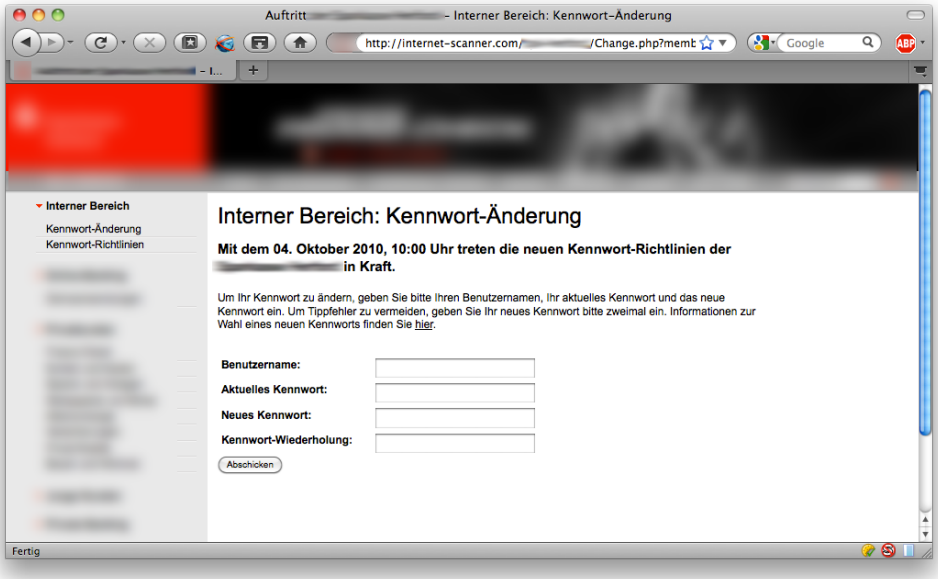


Abbildung 4.9.: Phishing-Seite zum Ändern eines Benutzerpassworts
Dies ist ein Beispiel für eine Phishing-Seite, die sich hinter dem Verweis aus einer Phishing-Mail, ähnlich der aus Abbildung 4.8, befindet. *Hinweis: Teile der Abbildung sind aus Datenschutzgründen unkenntlich gemacht.*

- Eine Mindestlänge von 9 Zeichen (Eine Ausnahme stellt Kunde 1 dar. Bei diesem Kunden wurde eine Mindestlänge von 8 Zeichen gefordert.)
- Es müssen Zeichen aus mindestens drei der folgenden Gruppen enthalten sein: Kleinbuchstaben, Großbuchstaben, Ziffern und Sonderzeichen
- Das Passwort darf weder den Vornamen, den Nachnamen noch den Benutzernamen des Mitarbeiters enthalten
- Das Passwort darf nicht in der Liste der fünf zuletzt verwendeten Passwörter enthalten sein

Die E-Mails dieses Laufs simulieren Phishing-Mails, die üblicherweise von einem Angreifer genutzt werden, um die Empfänger auf eine von ihm erstellte Phishing-Seite zu locken und so ihre Zugangsdaten zu Online-Diensten zu stehlen. Nachdem ein Angreifer zum Beispiel durch die in diesem Spam-Experiment durchgeführten Schritte die firmeninternen Zugangsdaten eines Mitarbeiters erhalten hat, kann er sich mit diesen im Firmennetz anmelden und sich dort als genau dieser Mitarbeiter ausgeben beziehungsweise alle Aktionen durchführen, zu denen der Mitarbeiter berechtigt ist.

Um bei den Spam-Experimenten möglichst realistisch einen Phishing-Angriff zu simulieren, wurden zwar Phishing-Seiten aufgesetzt, die den Internetauftritten der Kunden stark ähnelten, jedoch wurden keine sensiblen Daten oder die Passwörter der Mitarbeiter abgespeichert. Bei jeder Eingabe eines Mitarbeiters werden folgende Informationen aufgezeichnet:

1. Aktueller Zeitstempel
2. ID des Mitarbeiters

3. IP-Adresse des Rechners, von dem aus die Eingaben getätigt wurden, bzw. die IP-Adresse des dabei verwendeten Proxy-Servers
4. Eingegebener Benutzername
5. Für die Eingaben des aktuellen Passworts, des neuen Passworts und der Passwort-Wiederholung jeweils:
   - Gesalzener MD5-Hashwert der Eingabe
   - Anzahl der Kleinbuchstaben
   - Anzahl der Großbuchstaben
   - Anzahl der Ziffern
   - Anzahl der Sonderzeichen

Durch den gesalzenen Hashwert ist es sehr schwer (zumindest mit heutigen technischen Mitteln und begrenzter Zeit), auf die eingegebenen Passwörter der Mitarbeiter zurückzuschließen. Jedoch können mit Hilfe der Hashwerte die einzelnen Eingaben eines Mitarbeiters miteinander verglichen werden. Zum Beispiel ist es möglich festzustellen, ob sich ein neu eingegebenes Passwort eines Mitarbeiters von seinem aktuellen Passwort unterscheidet oder ob es das gleiche Passwort ist. Sollten die Passwörter identisch sein, so sind auch die daraus resultierenden Hashwerte identisch. Die Anzahl bestimmter Zeichen kann verwendet werden, um die Einhaltung der Passwort-Kriterien zu überprüfen.

4.2.5. Ergebnisse

In diesem Abschnitt werden die Ergebnisse der oben beschriebenen Spam-Experimente präsentiert. Aus verschiedenen Gründen sind die Ergebnisse als untere Schranken anzusehen:

- Das Ziel der Experimente war nicht, dass möglichst viele Empfänger auf die E-Mails dieser Experimente hereinfallen, sondern objektiv zu messen, wie sich die Mitarbeiter eines Kunden in Situationen verhalten, in denen sie möglicherweise zu einer Rechnerinfektion beitragen. Bei manchen Kunden gab es bereits in regelmäßigen Abständen Maßnahmen, die das Sicherheitsbewusstsein der Mitarbeiter fördern. Dies nimmt sicherlich Einfluss auf die Ergebnisse, da entsprechendes, sicherheitsrelevantes Vorwissen im Idealfall in einem antiproportionalen Verhältnis zu der Rate der Mitarbeiter steht, die einem Verweis folgen oder einen Dateianhang herunterladen und ausführen [Min09].
- Betrachtet man die Verbreitung von Social Malcode, so kann man davon ausgehen, dass der Großteil der Malware-Autoren nicht nur an einer Verbreitung innerhalb eines Unternehmens interessiert ist, sondern möglichst viele Rechner infizieren möchte. Menschen ohne das in dem ersten Punkt angesprochene Vorwissen, tragen sicherlich häufiger zu einer Infektion bei als Mitarbeiter der Unternehmen, bei denen die Experimente durchgeführt wurden.
- Bei den verschickten Dateianhängen wurden ausschließlich Office-Dokumente mit VBA-Makros verschickt. Diese verursachen normalerweise trotz Zertifikat noch eine Warnmeldung beim Öffnen der Dokumente, wenn die Sicherheitseinstellungen der entsprechenden Office-Anwendung nicht auf die niedrigste

Stufe eingestellt sind. Standardmäßig sind die Office-Anwendungen so konfiguriert, dass eine Warnmeldung auch dann angezeigt wird, wenn ein Makro mit einem nicht vertrauenswürdigem Zertifikat signiert ist. Da das verwendete *TC TrustCenter Class 2 L1 CA XII* Zertifikat kein (vertrauenswürdiges) Wurzelzertifikat ist, sondern lediglich ein SubCA-Zertifikat, wird eine Warnmeldung beim Öffnen der Dokumente angezeigt. Aus Kostengründen konnte leider kein Wurzelzertifikat oder ein von einem Wurzelzertifikat signiertes Zertifikat verwendet werden.

- Anstatt eines Office-Dokuments mit einem VBA-Makro würde ein realer Angreifer wohl eher ein Dokument mit Exploit-Code benutzen. Dies ist ein Dokument, das beim Öffnen in einer Anwendung Schwachstellen dieser ausnutzt, um so fremden Code zur Ausführung zu bringen [EWH09]. Bei diesem Vorgehen hätte der Angreifer nicht das gerade beschriebene Problem der Zertifizierung. Aus rechtlichen Gründen konnte jedoch bei den Spam-Experimenten kein Dokument verwendet werden, das eine Schwachstelle in einer vom Kunden verwendeten Anwendung ausnutzt.

- Im Laufe der Experimente stand nur eine Domain für den Feedback-Server zur Verfügung. Ein aufmerksamer Mitarbeiter wird sicherlich feststellen, dass die Verweise der Spam-Mails alle auf die Domain *internet-scanner.com* zeigen und sein Verhalten bei weiteren E-Mails dementsprechend anpassen. Folglich haben die Ergebnisse der ersten Experimente eines Kunden eine höhere Aussagekraft als die späteren Experimente.

- Schließlich ist es auch nicht auszuschließen, dass die Mitarbeiter untereinander über die Spam-Mails reden und sich so gegenseitig vor bestimmten Aktionen warnen. Vor allem bei Kunden, bei denen der Großteil der Mitarbeiter alle in einem Gebäude sitzen, ist dieser Effekt sehr wahrscheinlich. Bei zwei Kunden ist in diesem Zusammenhang auch bekannt, dass nach dem Versand der Spam-Mails etliche Telefonanrufe bei den Systemadministratoren eingingen, um auf die Spam-Mails hinzuweisen.

Tabelle 4.1 fasst die Hauptergebnisse der Spam-Experimente zusammen. Insgesamt gab es während der Zusammenarbeit mit der *8com GmbH & Co. KG* fünf Kunden, bei denen die hier beschriebenen Experimente durchgeführt wurden. Pro Kunde existiert in der Tabelle eine Zeile (farblich voneinander abgegrenzt), welche die entsprechende Anzahl der Mitarbeiter und die Ergebnisse der vier beschriebenen E-Mail-Läufe enthält. Dabei wurden bei dem ersten Kunden alle vier E-Mail-Läufe durchgeführt und bei den weiteren Kunden nur eine Teilmenge davon. Ist ein Lauf bei einem bestimmten Kunden nicht durchgeführt worden, so ist dies in der Tabelle durch einem Strich („–“) kenntlich gemacht. Für jedes durchgeführte Experiment und jeden Kunden gibt es zwei Einträge: Die obere der beiden Zahlen gibt den absoluten Anteil und die untere Zahl den relativen Anteil der Mitarbeiter des Kunden wieder, welche die in dem entsprechenden Experiment gemessene Aktion durchgeführt haben. Falls mehrere Reaktionen eines Mitarbeiters bei einem Experiment registriert wurden, beispielsweise wenn ein Mitarbeiter mehrfach dem Verweis der ersten Spam-Mail folgte, fließt dies nur einmal in die Werte der Tabelle ein.

Tabelle 4.1.: Übersicht über die Ergebnisse der Spam-Experimente
Dies ist eine Übersicht über die Ergebnisse der Spam-Experimente. Die letzten vier Spalten enthalten die gemessenen Raten der vier beschriebenen E-Mail-Läufe. Pro Kunde gibt es eine Zeile und die letzte Zeile enthält das Gesamtergebnis der Experimente.

Kunde	Anzahl der Mitarbeiter	Verweis gefolgt	Anhang geöffnet	Verweis auf ausführbare Datei		Phishing
				Heruntergeladen	Ausgeführt	
1	186	15 8,06 %	2 1,08 %	10 5,38 %	4 2,15 %	24 12,9 %
2	425	65 15,29 %	–	6 1,41 %	0 0 %	84 19,76 %
3	112	7 6,25 %	–	–	–	28 25 %
4	1 036	185 17,86 %	–	5 0,48 %	3 0,29 %	–
5	103	11 10,68 %	–	2 1,94 %	0 0 %	25 24,27 %
Gesamt	1 862	283 (von 1 862) 15,2 %	2 (von 186) 1,08 %	23 (von 1 750) 1,31 %	7 (von 1 750) 0,4 %	161 (von 826) 19,49 %

Die letzte Zeile der Tabelle 4.1 enthält das Gesamtergebnis der Experimente, dies entspricht den kumulierten Einzelergebnisse der Kunden. Auch hier gibt es für jedes durchgeführte Experiment wieder zwei Zahlen mit der gleichen Bedeutung wie bei jedem einzelnen Kunden. Für den Grundwert des prozentualen Anteils werden hier jedoch nur die Mitarbeiter der Kunden berücksichtigt, bei denen das entsprechende Experiment durchgeführt wurde. Zum besseren Verständnis ist hinter der absoluten Zahl jeweils in Klammern der Grundwert angegeben, auf den sich der entsprechende Anteil bezieht. Beispielsweise wurde in dem Experiment-Durchlauf mit dem Dateianhang nur die Mitarbeiterzahl des ersten Kunden für die Berechnung des prozentualen Gesamtergebnisses verwendet, da nur bei diesem Kunden das Experiment durchgeführt wurde.

Auffällig an den Ergebnissen ist der hohe Anteil der Mitarbeiter, die dem Verweis der ersten Experiment-Durchläufe gefolgt sind, verglichen mit den Ergebnissen des zweiten (Dateianhang) und dritten (Verweis auf eine ausführbare Datei) Experiments. Dies lag im Wesentlichen daran, dass für die Experimente nur eine Domain zur Verfügung stand. Rückmeldungen der Ansprechpartner der Kunden zeigten, dass ein großer Anteil der Mitarbeiter die Systemadministratoren, beziehungsweise die zuständigen Personen der Kunden, bereits nach dem ersten Experiment-Durchlauf auf diese Domain aufmerksam gemacht haben. Obwohl auf diese Domain auch in den E-Mails der Phishing-Experimente verwiesen wurde (siehe Abbildung 4.8), sind die Ergebnisse dieser Experimente, die bei jedem Kunden als letztes durchgeführt wurden, wieder relativ hoch. Vermutlich liegt dies an der gefälschten Absenderadresse und den optisch an die Internetauftritte der Kunden angelehnten Phishing-Seiten, so dass sich die Mit-

arbeiter der Kunden trotz der offensichtlich nicht zu den Kunden gehörigen Domain haben täuschen lassen.

Beim Protokollieren der Reaktionen wurde unter anderem auch der genaue Zeitpunkt aufgezeichnet, zu dem eine Reaktion registriert wurde. Abbildung 4.10 zeigt die Verteilung der Stunden, die seit dem Zeitpunkt des Versands einer Spam-Mail und den Zeitpunkten, zu denen eine Reaktion registriert wurde, vergangen sind (x-Achse). Die blauen Balken geben die absolute Anzahl der Mitarbeiter (Skala der linken y-Achse) und die farbige Fläche den kumulierten Anteil der Mitarbeiter (Skala der rechten y-Achse) wieder. Bei der Auswertung des zeitlichen Aspekts, werden alle 715 protokollierten Reaktionen berücksichtigt – ist ein Mitarbeiter etwa dreimal dem Verweis aus dem ersten E-Mail-Lauf gefolgt, so werden in die folgenden Auswertungen auch alle drei Reaktionen einbezogen. Aus Darstellungsgründen sind in dieser Abbildung nur die Reaktionen aufgetragen, die innerhalb einer Woche (dies entspricht 168 Stunden) registriert wurden. Außerhalb des dargestellten Bereichs wurden weitere 55 Reaktionen protokolliert.

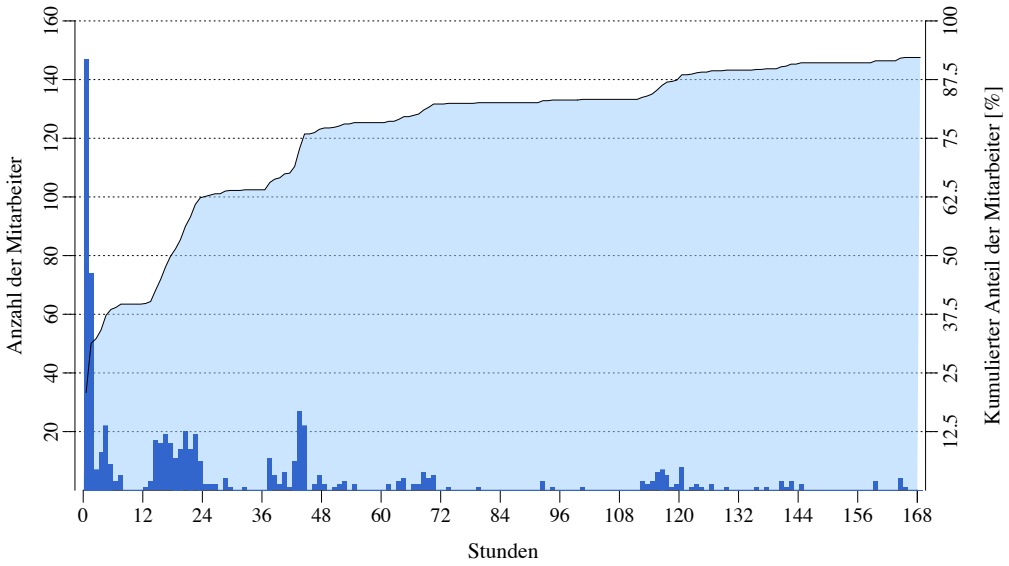


Abbildung 4.10.: Zeitliche Verteilung der Reaktionszeiten der Spam-Experimente
Die hier dargestellten Reaktionszeiten sind die Stunden, die seit dem Versand der Spam-Mails und den Zeitpunkten des Registrierens einer Reaktion vergangen sind. Berücksichtigt wurden alle aufgezeichneten Reaktionen der Mitarbeiter aller fünf Kunden.

Die längste Zeitspanne zwischen dem Versand einer Spam-Mail und dem Registrieren einer Reaktion betrug in den Experimenten 567 Stunden, dies entspricht 23 Tagen und 15 Stunden. Diese sehr späten Reaktionen liegen wahrscheinlich an krankheits- oder urlaubsbedingten Fehlzeiten der Mitarbeiter. Die Spam-Mail, bei der eine Reaktion nach 567 Stunden festgestellt wurde, ist am 20.04.2010 an einen Kunden in Rheinland-Pfalz[2] verschickt worden. Daher ist zu vermuten, dass sich in diesem Fall die lange Zeitspanne wahrscheinlich eher durch eine krankheitsbedingte Fehlzeit dieses Mitarbeiters begründen lässt, da zu dieser Zeit keine Ferien in Rheinland-Pfalz

[2]Bundesland im Südwesten Deutschlands

waren, oder er aus sonstigen Gründen nicht sein E-Mailkonto überprüfen konnte. Würden diese sehr späten Reaktionen ebenfalls in Abbildung 4.10 dargestellt, würden die Balken so schmal, dass sie kaum noch unterscheidbar wären.

Der Großteil der aufgezeichneten Reaktionen fand jedoch innerhalb kurzer Zeit nach dem Versand der Spam-Mails statt. Innerhalb der ersten 24 Stunden waren 62,32 % der Reaktionen protokolliert und nach 48 Stunden bereits 76,91 %. Das arithmetische Mittel liegt bei ca. 47,05 Stunden und die Standardabweichung beträgt ungefähr 77,13 Stunden. Mit 147 aufgezeichneten Reaktionen ist die erste Stunde, verglichen mit den restlichen Stunden, diejenige, in der am meisten Reaktionen registriert werden konnten. Das 0,5-Quantil der Verteilung, also die Grenze, welche die Anzahl der aufgezeichneten Reaktionen halbiert, liegt zwischen 17 und 18 Stunden und das 0,9-Quantil liegt zwischen 139 und 140 Stunden, d. h. dass über 90 % der Reaktionen innerhalb der ersten 140 Stunden (entspricht ca. 5,83 Tagen) registriert wurden.

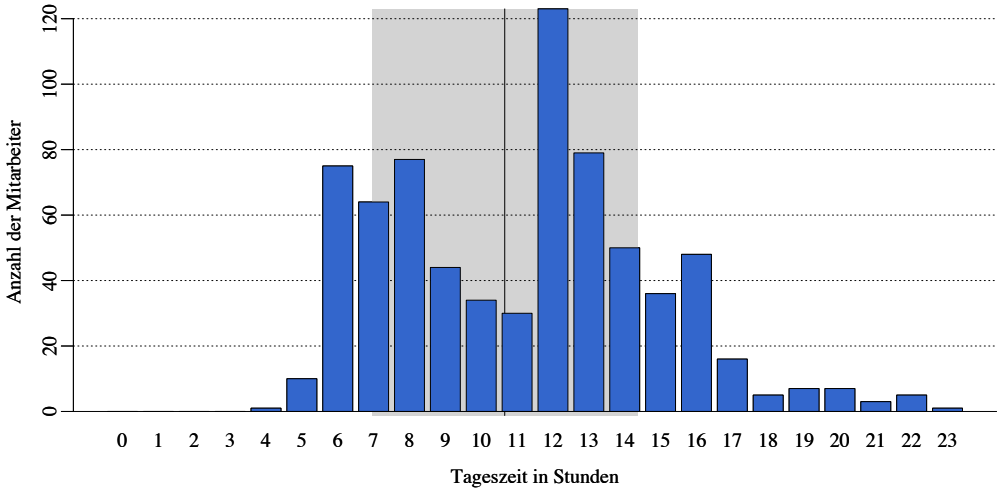


Abbildung 4.11.: Verteilung der Reaktionszeiten über einen Tag
Diese Abbildung zeigt zu welchen Tageszeiten die Reaktionen erfolgten. Die vertikale schwarze Linie ist das arithmetische Mittel und die graue Fläche stellt die Standardabweichung um dieses Mittel dar.

Über den Tag verteilt, konnten Reaktionen von 4 Uhr morgens bis 0 Uhr nachts registriert werden – 92,31 % der Reaktionen fanden zwischen 6 und 17 Uhr statt. Abbildung 4.11 zeigt die komplette zeitliche Verteilung der protokollierten Reaktionen stundengenau über einen vollständigen Tag. Das arithmetische Mittel liegt bei 11:09:36 Uhr (dies entspricht 11,16 Stunden) und ist in der Abbildung durch die vertikale schwarze Linie gekennzeichnet. Die graue Fläche um das Mittel stellt die Standardabweichung von 3,69 Stunden auf beiden Seiten des Mittels dar. Die E-Mails eines einzelnen Laufs wurden immer alle gleichzeitig versendet. Die einzelnen Läufe wurden jedoch nicht immer zu einer bestimmten Tageszeit durchgeführt, sondern verteilten sich über einen Zeitraum von ca. 8 Uhr morgens bis ca. 18 Uhr abends – je nachdem, wie dies mit dem entsprechenden Kunden abgesprochen war.

Dass alle fünf Kunden ihren Standort in Deutschland haben, stellt sich auch in der Abbildung dar: Der Großteil der Reaktionen wurde während typisch deutscher Ar-

beitszeiten gemessen. Zudem lässt sich in der Abbildung erkennen, dass viele Mitarbeiter morgens zu Beginn eines Arbeitstages ihre E-Mails lesen und dies auf die Mittagspause zugehend abnimmt. Während dieser, genau gesagt zwischen 12 und 13 Uhr, wurden mit 123 Reaktionen die meisten Rückmeldungen an den Feedback-Server gesendet und danach nimmt die Anzahl der Reaktionen wieder von Stunde zu Stunde ab. Kurz vor Arbeitsende überprüfen dann anscheinend nochmals viele Mitarbeiter ihre Postfächer, da zwischen 16 und 17 Uhr wieder ein Anstieg an registrierten Reaktionen festgestellt werden konnte. Die abends protokollierten Reaktionen wurden häufig von IP-Adressen an den Feedback-Server übermittelt, die nicht aus den IP-Adressbereichen der Kunden stammen. Es liegt die Vermutung nahe, dass diese Reaktionen von Mitarbeitern stammen, die von zu Hause aus ihre geschäftlichen E-Mails lesen.

Abbildung 4.12 zeigt die durchschnittlichen Passwortlängen, die von den Mitarbeitern auf den Phishing-Seiten eingegeben wurden. Für jeden Kunden, bei dem der Phishing-Lauf durchgeführt wurde, existieren in der Abbildung zwei Balken: Ein dunkelblauer Balken, der die durchschnittliche Länge der aktuell von den Mitarbeitern verwendeten Passwörter angibt, und ein hellblauer Balken für das neu gewählte Passwort. Die Längenangaben entsprechen der Zeichenanzahl eines Passworts.

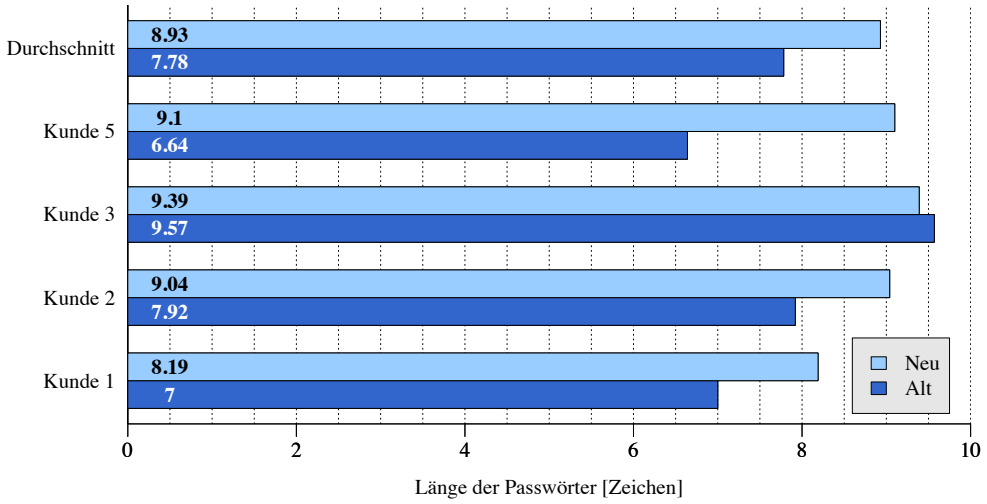


Abbildung 4.12.: Länge der eingegebenen Passwörter
Diese Abbildung zeigt die durchschnittlichen Längen der eingegebenen, aktuellen und neuen Passwörter in den Phishing-Durchläufen für jeden Kunden. Zudem sind die beiden durchschnittlichen Passwortlängen aller fünf Kunden angegeben.

Bei fast allen Kunden haben die Mitarbeiter, im Vergleich zu ihren alten Passwörtern, durchschnittlich längere, neue Passwörter gewählt. Die einzige Ausnahme sind die Mitarbeiter des Kunden 3. Bei diesem Kunden wäre bei einer echten Passwortänderung die durchschnittliche Passwortlänge um 1,88 % auf 9,39 Zeichen gesunken. Die Mitarbeiter dieses Kunden hatten jedoch auch mit Abstand die längsten Passwörter aller Mitarbeiter und wie in Abschnitt 4.2.4 beschrieben, wurden die Mitarbeiter auf der Phishing-Seite darauf hingewiesen, dass das neue Passwort eine Mindestlänge von neun Zeichen haben soll. Vermutlich haben einige Mitarbeiter des Kunden 3 dies ge-

nutzt, um ein kürzeres Passwort zu wählen, das immer noch die auf der Phishing-Seite geforderten Passwort-Richtlinien erfüllt.

Die größte Auswirkung auf die Passwortlängen der Mitarbeiter hatte die Phishing-Mail bei Kunde 5. Dort wäre die Passwortlänge im Schnitt von 6,64 Zeichen um ca. 37,05 % auf 9,1 Zeichen gestiegen. Alle Kunden betrachtet, stieg die durchschnittliche Passwortlänge um 14,78 % von 7,78 Zeichen auf 8,93 Zeichen. Die relativ kurze, neue Passwortlänge des Kunden 1 ist dadurch bedingt, dass bei diesem Kunden auf der Phishing-Seite lediglich eine Mindestlänge von acht Zeichen gefordert wurde. Deshalb wird das Kriterium der Mindestlänge auch bei diesem Kunden von den vermeintlich neuen Passwörtern erfüllt.

4.3. Zeitspanne zwischen dem Eintreffen und Lesen einer E-Mail

Mit Hilfe der Spam-Experimente des vorherigen Abschnitts haben wir erste Kennzahlen bezüglich des E-Mail-Verhaltens von Benutzern messen können. Aus diesen Messungen ging zum Beispiel der Prozentsatz der Benutzer hervor, die einem Verweis folgen oder einen Dateianhang herunterladen und öffnen (siehe Tabelle 4.1). Außerdem konnten durch diese Experimente die Zeitspannen bestimmt werden, die zwischen dem Versenden der E-Mails und dem Registrieren einer dadurch bedingten Benutzerreaktion vergingen (siehe Abbildung 4.10).

Ein Faktor konnte durch diese Experimente jedoch nicht bestimmt werden: die Zeitspanne, die zwischen dem Eintreffen einer E-Mail im Posteingang und dem Lesen durch den Benutzer vergeht. Dieser Faktor hat erheblichen Einfluss auf die Verbreitungsgeschwindigkeit von Social Malcode und ist wesentlich dafür verantwortlich, dass sich Social Malcode langsamer verbreitet als sich autonom verbreitende Malware. Definitionsgemäß ist in mindestens einer Verbreitungsroutine eines Social Malcode eine Benutzerinteraktion vorhanden. Damit es bei der Verbreitung durch E-Mails zu dieser Interaktion kommt, muss das Opfer die entsprechende E-Mail gelesen haben. Je länger sich solch eine E-Mail ungelesen im Postfach eines Benutzers befindet, desto langsamer verbreitet sich Social Malcode.

Zwar haben wir im Rahmen der Spam-Experimente des vorherigen Abschnitts die Zeiten zwischen dem Versand der E-Mails und dem Zeitpunkt einer Reaktion gemessen (siehe Tabelle 4.10), jedoch wurden diese Experimente ausschließlich in einem beruflichen Umfeld durchgeführt. Da bei fast alle Kunden die E-Mail-Adressen lediglich beruflich genutzt wurden, entstehen dadurch mehr oder weniger regelmäßige Intervalle der E-Mail-Nutzung, so dass wir diese Werte für den weiteren Verlauf der Arbeit nicht verwenden möchten.

4.3.1. Web Bugs

Die erste Idee zum Messen dieser Zeitspanne war die Verwendung von HTML-basierten E-Mails. In diese kann man Bilder einbetten, die auf einem externen Server liegen. Jedes Mal, wenn der Empfänger solch eine E-Mail öffnet, lädt der E-Mail-Client das Bild von dem Server herunter, um dieses im Inhalt der E-Mail anzuzeigen.

Dabei entsteht ein Zugriff auf den Server, über den man den Zeitpunkt bestimmen kann, zu dem die E-Mail gelesen wurde. Folglich lässt sich auf diese Weise die gesuchte Zeitspanne messen.

Leider wird bzw. wurde vor ein paar Jahren noch eine ähnliche Technik beim Versand von Spam-Mails verwendet, um zu überprüfen, ob eine E-Mail-Adresse valide ist und auch abgerufen wird. Beim Versand werden mehr oder weniger zufällig E-Mail-Adressen generiert, an die die Spam-Mails verschickt werden. Innerhalb der HTML-Mails sind 1×1 Pixel große Bilder eingebunden, die sich optisch nicht vom Hintergrund der E-Mails unterscheiden und somit auch nicht von den Empfängern wahrgenommen werden. Für jeden einzelnen Empfänger ist dessen E-Mail-Adresse in die Quelle (*src*-Attribut des HTML-Tags <*img*>) des Bildes einkodiert. Wird ein solches Bild von dem entsprechenden Server abgerufen, ist das ein Zeichen dafür, dass die entsprechende E-Mail gelesen wurde und die dazugehörige E-Mail-Adresse gültig ist. Diese unsichtbaren Bilder werden als *Web Bug* oder auf Deutsch als *Zählpixel* bezeichnet [Ben01].

Um die Privatsphäre von Benutzern zu schützen, zeigen viele aktuelle E-Mail-Clients standardmäßig keine externen Inhalte an, sondern fragen den Benutzer, ob der Inhalt nachgeladen werden soll. Das obere Fenster der Abbildung 4.13 zeigt solch einen Sicherheitshinweis (rot umrandet) der Version 8.0 des E-Mail-Clients *Mozilla Thunderbird* [Moz11]. Im unteren Fenster der Abbildung ist ein Teil des Quelltextes dieser E-Mail zu sehen. In dem Bildschirmausschnitt ist das HTML-Tag <*img*>, durch das der Web Bug realisiert ist, rot umrandet. Das Bild wird durch die Attribute *height=1*, *width=1* (Größe: 1×1 Pixel) und *border=0* (rahmenlos) nicht wahrnehmbar in die E-Mail eingebunden.

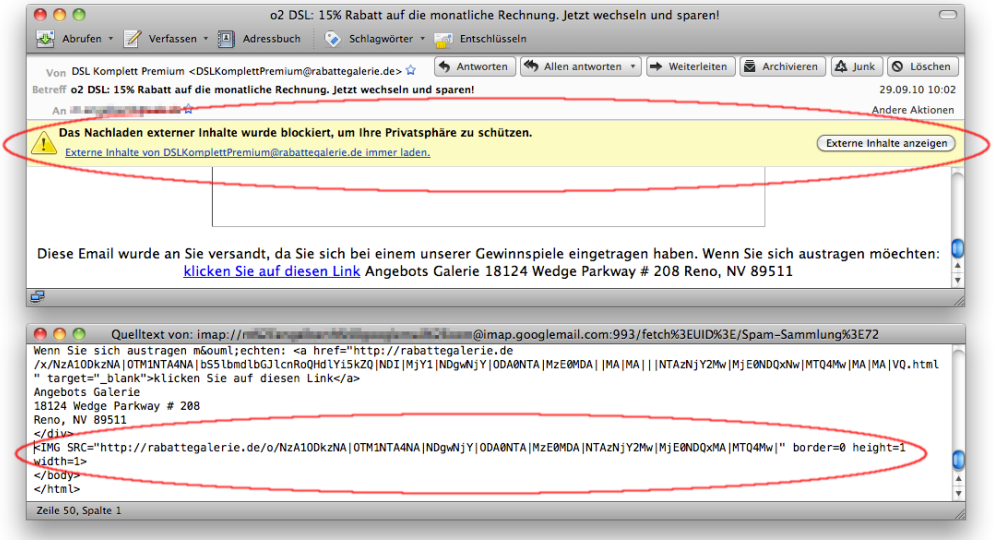


Abbildung 4.13.: Web Bug in einer Spam-Mail
Mit Hilfe von Web Bugs lässt sich die Validität einer E-Mail-Adresse überprüfen. Viele aktuelle E-Mail-Clients zeigen aus diesem Grund externe Inhalte nicht mehr ohne Zustimmung des Benutzers an.

Stimmt ein Benutzer dem Nachladen der externen Inhalte nicht zu, so finden keine Zugriffe auf den Server statt, auf dem sich die externen Inhalte befinden. Dementsprechend wird auch keine Rückmeldung generiert, ob die benutzte E-Mail-Adresse valide ist und abgerufen wird. Durch diesen mittlerweile weit verbreiteten Schutzmechanismus der aktuellen E-Mail-Clients, ist die Technik der Web Bugs nicht geeignet, um die Zeitspannen zwischen dem Eintreffen und Lesen von E-Mails zu messen. Wir würden maximal eine Rückmeldung beim Lesen der E-Mails erhalten, wenn die Benutzer dem E-Mail-Client gestatten, die externe Inhalte nachzuladen.

4.3.2. Auslesen am Mailserver

Die zweite Idee zum Messen der gesuchten Zeitspanne zwischen dem Eintreffen und dem Lesen einer E-Mail bestand darin, die benötigten Informationen direkt dort zu ermitteln, wo diese anfallen: am Mailserver selbst. Für diesen Zweck haben wir ein Skript geschrieben, das auf dem Mailserver der Universität Mannheim ausgeführt wurde. Im Gegensatz zu den Spam-Experimenten des Abschnitts 4.2 verwenden viele Studenten und Mitarbeiter der Universität Mannheim ihre E-Mail-Adresse nicht nur zu typischen, deutschen Arbeitszeiten. Teilweise werden die E-Mail-Adressen auch privat genutzt.

4.3.2.1. Cyrus

Die Universität Mannheim verwendet den Mailserver *Cyrus*, der ein Open-Source-Projekt der *Carnegie Mellon University* ist [Car11a]. In Abbildung 4.14 ist eine vereinfachte, schematische Darstellung der Funktionsweise von Cyrus abgebildet.

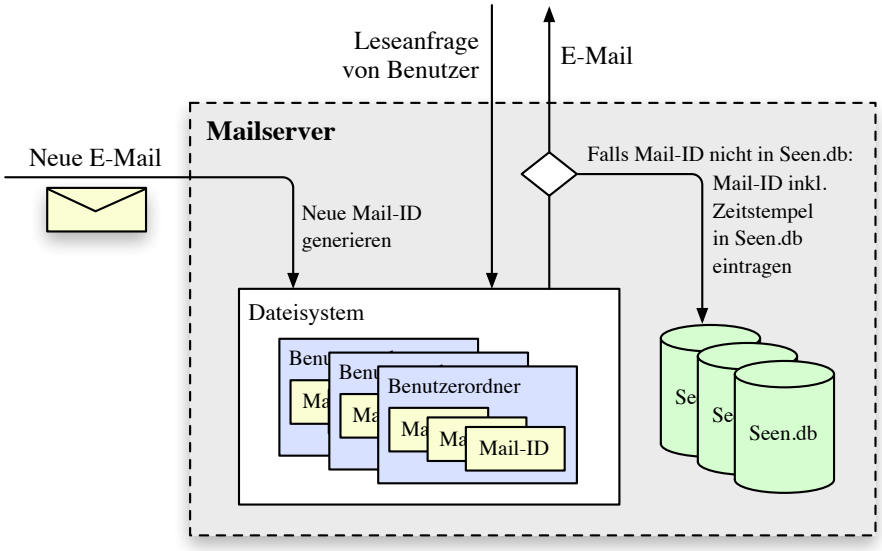


Abbildung 4.14.: Schematische Darstellung des Mailservers Cyrus
Trifft eine neue E-Mail am Mailserver ein, wird eine Mail-ID generiert und die E-Mail in einem Unterordner des Empfängers im Dateisystem abgelegt. Liest der Benutzer die E-Mail, wird dessen persönliche Datenbank der bereits gelesenen E-Mails aktualisiert.

Die Abbildung enthält nur eine abstrahierte Sicht auf die für die Messung der gesuchten Zeitspannen relevanten Teile des Funktionsumfangs. Der komplette Funktionsumfang des Mailservers ist wesentlich komplexer. Die Dateien des Datenbankmanagementsystems befinden sich in der Realität natürlich auch im Dateisystem. Aber da es sich hier um eine schematische Abbildung handelt und die Datenbanken ein eigentständiges System zur elektronischen Datenverwaltung darstellen, befinden sich diese in der Abbildung ausserhalb des Dateisystems. Der Ablauf beim Empfang und beim Lesen einer E-Mail stellt sich bei Cyrus folgendermaßen dar:

- **Empfang einer neuen E-Mail:** Für jeden registrierten Benutzer existiert ein eigener Ordner im Dateisystem des Mailservers. Erhält ein Benutzer eine neue E-Mail, so wird dieser E-Mail eine eindeutige Identifikationsnummer (Mail-ID) zugewiesen. Die Mail-ID wird gleichzeitig als Dateiname der neuen E-Mail im Dateisystem benutzt. Die Verzeichnisstruktur unterhalb des Benutzerordners entspricht der Ordnerstruktur des Postfachs.
- **Lesen einer E-Mail:** Jeder Benutzer hat eine eigene Datenbank, die Informationen darüber enthält, welche E-Mails bereits gelesen wurden und welche noch ungelesen sind. In der Abbildung sind diese Datenbanken durch die grünen Icons mit der Aufschrift *Seen.db* symbolisiert. Möchte ein Benutzer seine E-Mails angezeigt bekommen, liest der Mailserver die E-Mail-Inhalte und Header-Informationen aus dem Dateisystem und präsentiert dem Benutzer die angefragten E-Mails. Dabei wird überprüft, ob sich die Mail-ID einer angefragten E-Mail bereits in der Seen.db des anfragenden Benutzers befindet (also bereits gelesen wurde). Ist dies nicht der Fall, d. h. die E-Mail ist noch ungelesen, wird die Mail-ID und der aktuelle UNIX-Zeitstempel in Form eines neuen Eintrags der Seen.db hinzugefügt und dadurch die entsprechende E-Mail als gelesen markiert.

4.3.2.2. Skript zum Ermitteln der Zeitspannen

Das von uns in Python implementierte Skript (der komplette Quellcode befindet sich in Anhang C) ermittelt für jede E-Mail eines Benutzers den Zeitpunkt des Eintreffens in der Mailbox (Zustellzeitpunkt) und den Zeitpunkt, zu dem die E-Mail das erste Mal gelesen wurde (Lesezeitpunkt). Als Eingabe bekommt das Skript einen Benutzernamen übergeben und die Ausgabe besteht aus mehreren UNIX-Zeitstempeln, aus denen die gesuchten Zeitspannen berechnet werden.

Eine beispielhafte und gekürzte Version der Ausgabe zeigt Code-Block 4.2. Als Erstes werden Informationen über die aktuellen Zeiteinstellungen des Mailservers ausgegeben (Zeilen 1 bis 5). Diese werden zur richtigen Interpretation der weiteren Zeitstempel benötigt. Für jede E-Mail, deren ID in der Seen.db des Benutzers enthalten ist, werden anschließend ein Hashwert und fünf Zeitstempel ausgegeben.

Der SHA-1 Hashwert dient zur eindeutigen Identifikation einer E-Mail. Dieser Wert wird über den Benutzernamen, die ID des Ordners und die Mail-ID generiert. Auf diese Weise lassen sich E-Mails eindeutig voneinander unterscheiden ohne die Klartextwerte der drei genannten Eigenschaften zu kennen. Benötigt wird dies, um die Zeitstempel einer E-Mail nicht mehrmals auszuwerten, falls das Skript öfters ausgeführt wird.

```
time      : 1306151913.34
altzone   : -7200
localtime : time.struct_time(tm_year=2011, tm_mon=5, tm_mday=23, tm_hour=13, tm_min=58, tm_sec=33,
      tm_wday=0, tm_yday=143, tm_isdst=1) (1306151913.0)
ctime     : Mon May 23 13:58:33 2011
timezone  : ('CET', 'CEST')

SHA1(mailbox,folderID,mailID)   (TS_RECEIVED,  TS_DATE,      TS_CTIME)    TS_NOT_READ  TS_READ
a38900a2f5d38[...]14ff1b43b4e3b (1306099639.0, 1306099636.0, 1306150869)  1305637563   1306150869
d91120aa75fc9[...]ec2f2368540f5 (1305920125.0, 1305920120.0, 1305920125)  1305919903   1305920145
[...]
```

Code-Block 4.2: Exemplarische Ausgabe des Skripts

Die ersten drei nach dem Hashwert ausgegebenen Zeitstempel beziehen sich auf den Zustellzeitpunkt der E-Mail:

- **TS_RECEIVED:** Dieser Zeitstempel wird aus dem Header der E-Mail ausgelesen und entspricht dem Datum des zeitlich letzten *Received*-Eintrags (im Quelltext der E-Mail also der oberste Eintrag). Jeder Mailserver, der am Transport einer E-Mail vom Absender zum Empfänger beteiligt ist, fügt am Anfang des Quelltexts einen neuen Received-Header hinzu [Res08].
- **TS_DATE:** Auch der zweite Zeitstempel wird dem Header der E-Mail entnommen. Dieser Zeitstempel entspricht dem Wert des *Date*-Eintrags und beinhaltet den Zeitpunkt, zu dem der Autor die E-Mail verfasst hat [Res08].
- **TS_CTIME:** Der letzte Zeitstempel, der sich auf den Zustellzeitpunkt einer E-Mail bezieht, ist die *create time* der E-Mail-Datei im entsprechenden Benutzerordner des Dateisystems. Es ist zu beachten, dass dieser Zeitstempel unter Windows- und UNIX-artigen Betriebssystemen unterschiedlich behandelt wird: Windows setzt diesen Zeitstempel, wenn die Datei erstellt wird. UNIX-artige Betriebssysteme hingegen verändern diesen Wert immer dann, wenn sich Metainformationen der Datei ändern, wie zum Beispiel der Besitzer oder die Rechte der Datei.

Mit Hilfe dieser drei Zeitstempel wird der Zustellzeitpunkt einer E-Mail wie folgt definiert: Da die Received-Header einer E-Mail von den beteiligten Mailservern gesetzt werden, werden diese als am verlässlichsten erachtet. Dementsprechend wird als Zustellzeitpunkt der Wert TS_RECEIVED verwendet. Dadurch, dass Autoren einer E-Mail diese offline verfassen und erst zu einem späteren Zeitpunkt verschicken können, sind die Date-Header einer E-Mail etwas ungenauer. Davon abgesehen, lassen sich die Date-Header auch beliebig manipulieren. Sollte jedoch aus irgendeinem Grund der Wert TS_RECEIVED nicht zu ermitteln sein, wird auf den Zeitpunkt TS_DATE zurückgegriffen. Da der Mailserver der Universität Mannheim auf einem Rechner mit einem UNIX-artigen Betriebssystem läuft, wird der Zeitpunkt TS_CTIME aus oben genannten Gründen als am ungenausten angesehen. Sind die ersten beiden Werte jedoch nicht vorhanden, wird der Wert von TS_CTIME als letzte Alternative für die weiteren Berechnungen als Zustellzeitpunkt verwendet.

Um die gewünschten Zeitspannen zu bestimmen, werden neben den Zustellzeitpunkten der E-Mails auch Zeitstempel benötigt, die den Zeitpunkt des Lesens der E-Mails einschränken. Diese werden den einzelnen Benutzer-Datenbanken (Seen.db) entnommen, die vom Rechenzentrum der Universität Mannheim in einem binären Dateiformat betrieben werden. Für jeden Ordner einer Mailbox wird separat gespeichert,

welche E-Mails bereits gelesen wurden. Dazu werden für jeden Ordner der Mailbox Einträge in der Seen.db hinterlegt, die folgenden Aufbau besitzen [Car11b]:

```
<Version>␣<Last Read Time>␣<Last Read UID>␣
  <Last Change Time>␣<List of Read UIDs>
```

Die einzelnen Felder eines Eintrags haben folgende Bedeutung: *Version* enthält die aktuelle Datenbank-Version. *Last Read Time* entspricht dem letzten Zeitpunkt, an dem eine beliebige E-Mail des Ordners gelesen wurde. *Last Read UID* entspricht der Mail-ID der E-Mail, die zuletzt gelesen wurde. *Last Change Time* ist der Zeitpunkt, zu dem das letzte Mal eine ungelesene E-Mail gelesen wurde (also der Zustand mindestens einer E-Mail von *ungelesen* in *gelesen* übergegangen ist), und *List of Read UIDs* ist die Liste der bereits gelesenen E-Mails (genauer der Mail-IDs) des entsprechenden Ordners.

Unser Skript liest diese Einträge aus und identifiziert für jede gelesene E-Mail eines Ordners folgende zwei Zeitpunkte, die ebenfalls ausgegeben werden:

- **TS_NOT_READ:** Dies ist der maximale Wert *Last Change Time* der Seen.db-Einträge, deren *List of Read UIDs* die Mail-ID der aktuellen E-Mail noch nicht enthält. Dieser Zeitstempel entspricht also dem spätesten Zeitpunkt, zu dem die E-Mail noch ungelesen ist.
- **TS_READ:** Dies ist der minimale Wert *Last Change Time* der Seen.db-Einträge, deren *List of Read UIDs* die Mail-ID der aktuellen E-Mail bereits enthält. Dieser Zeitstempel entspricht also dem frühesten Zeitpunkt, zu dem die E-Mail bereits gelesen wurde.

Somit verfügen wir über drei Zeitstempel, die über den Zustellzeitpunkt einer E-Mail Auskunft geben, und zwei Zeitstempel, mit denen sich der Lesezeitpunkt bestimmen lässt. Dementsprechend sind wir in der Lage die Differenz aus beiden Zeitpunkten zu bilden und so die gesuchte Zeitspanne zu berechnen.

4.3.3. Diskussion

Zur Bestimmung des Lesezeitpunktes war die erste Idee, den Mittelwert aus den beiden Zeitstempeln TS_READ und TS_NOT_READ als den Lesezeitpunkt einer E-Mail zu definieren. Jedoch ergab dies einige negative Zeitspannen, da der so berechnete Lesezeitpunkt vor dem Zustellzeitpunkt der E-Mail lag. Daraus resultierend wird bei den weiteren Auswertungen der Zeitstempel TS_READ als Lesezeitpunkt verwendet und somit bilden die in Abschnitt 4.3.4 präsentierten Ergebnisse obere Schranken für die gesuchten Zeitspannen.

Zudem werden die Einträge der Seen.db immer dann geschrieben, wenn sich ein Benutzer vom Mailserver abmeldet. Dies ist ein zusätzlicher Grund dafür, dass sich durch diese Zeitstempel der Lesezeitpunkt einer E-Mail nur einschränken, jedoch nicht genau bestimmen lässt. Meldet sich ein Benutzer beispielsweise am Mailserver an, liest nacheinander zwei bisher noch ungelesene E-Mails und meldet sich dann wieder ab, weist unser Skript beiden E-Mails den gleichen Lesezeitpunkt zu, obwohl zwischen dem Lesen der beiden E-Mails eventuell noch Zeit vergangen ist.

Allerdings sind wir bei diesem Experiment nicht an einer sekundengenauen Messung der Zeitspannen interessiert, sondern möchten eine stundenweise Verteilung der Zeitspannen betrachten. Dementsprechend sind diese minimalen Ungenauigkeiten für die hier präsentierte Arbeit tolerabel.

Des Weiteren werden nicht alle jemals generierten Einträge in den einzelnen Seen.db-Datenbanken aufbewahrt. In regelmäßigen Abständen werden diejenigen Einträge gelöscht, die im Vergleich zu den anderen Einträgen keine zusätzlichen Informationen enthalten. Aus diesem Grund konnten nicht für alle E-Mails der Postfächer die beiden gesuchten Zeitpunkte berechnet werden. Für eine Vielzahl der E-Mails war zum Beispiel nur noch ersichtlich, dass diese bereits gelesen wurden, jedoch existierten keine Einträge mehr, die Informationen darüber hergeben, zu welchen Zeitpunkten die entsprechenden E-Mails noch nicht gelesen waren.

Angenommen in einem Postfach befinden sich fünf E-Mails M_1 bis M_5. Liest der Benutzer zuerst E-Mail M_1, zu einem späteren Zeitpunkt die E-Mails M_4 und M_2 und in einer letzten Session die E-Mail M_3, dann befinden sich in der entsprechenden Seen.db-Datenbank die oberen drei Einträge der Tabelle 4.2. Die Bedeutung der einzelnen Felder ist auf Seite 99 nachzulesen.

Tabelle 4.2.: Exemplarische Einträge einer Seen.db-Datenbank
Diese Tabelle enthält beispielhafte Einträge einer Seen.db-Datenbank zu zwei verschiedenen Zeitpunkten. Die Zeitpunkte sind hier durch die mittlere Linie voneinander abgetrennt.

Version	Last Read Time	Last Read UID	Last Change Time	List of Read UIDs
1	$time_1$	M_1	$time_2$	M_1
1	$time_3$	M_2	$time_4$	M_1, M_2, M_4
1	$time_5$	M_3	$time_6$	$M_1 : M_4$
1	$time_3$	M_2	$time_4$	M_1, M_2, M_4
1	$time_5$	M_3	$time_6$	$M_1 : M_4$
1	$time_7$	M_5	$time_8$	$M_1 : M_5$

Da der Mailserver Cyrus diese Einträge chronologisch anlegt, gilt folgender Zusammenhang:

$$time_i \leq time_{i+2}, \quad \text{für alle } i \in \{2, 4, 6\}$$

Aus den ersten drei Einträgen der Tabelle lässt sich also herleiten, dass die E-Mail M_2 zum Zeitpunkt $time_2$ noch nicht gelesen wurde, jedoch zum Zeitpunkt $time_4$ bereits als gelesen markiert war. Entsprechend muss die E-Mail M_2 zwischen diesen beiden Zeitpunkten gelesen worden sein.

Liest der Benutzer nun irgendwann die fünfte E-Mail M_5 und wird der älteste Eintrag der Seen.db-Datenbank gelöscht, so beinhaltet die Datenbank die unteren drei Einträge der Tabelle 4.2. In diesem Zustand der Datenbank lässt sich die gesuchte Zeitspanne (zwischen Zustell- und Lesezeitpunkt) für die E-Mail M_2 nicht mehr eingrenzen, da sich keine Aussagen mehr zum Lesezeitpunkt machen lassen: Zwar ist immer noch herleitbar, dass die E-Mail M_2 zum Zeitpunkt $time_4$ bereits gelesen wurde, jedoch findet sich in der Datenbank kein Eintrag mehr, der einen Zeitpunkt bestimmt, zu dem die E-Mail M_2 noch ungelesen war.

4.3.4. Ergebnisse

Mit Hilfe des in Abschnitt 4.3.2.2 beschriebenen Skriptes haben wir die Zustell- und die Lesezeitpunkte von E-Mails auf dem Mailserver der Universität Mannheim bestimmt. Insgesamt haben sich 33 Mitglieder der Universität Mannheim dazu bereit erklärt, dass das von uns geschriebene Skript auf ihre Postfächer angewandt werden darf. Für diese Mithilfe und das Unterschreiben der dafür notwendigen Enverständniserklärung (siehe Anhang B) möchten wir uns an dieser Stelle nochmals ausdrücklich bei den Teilnehmern bedanken.

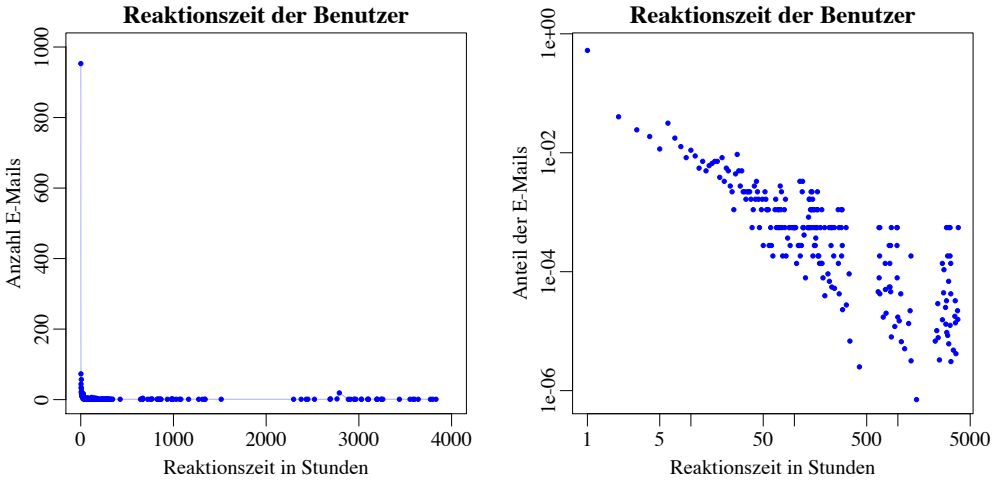
Innerhalb der 33 untersuchten Postfächer konnte auf diese Weise für 1 813 E-Mails die Dauer zwischen dem Eintreffen und Lesen der E-Mails bestimmt, beziehungsweise eingegrenzt, werden. Durch die Arbeitsweise des Skriptes erhalten wir eine sekundengenaue Messung der gesuchten Zeitspanne. Jedoch sind wir aufgrund der geringen Anzahl von E-Mails an einer stundengenauen Auswertung interessiert. Deshalb haben wir die sekundengenauen Lesezeiten in Stundenintervalle eingeteilt: Die Anzahl der E-Mails, die innerhalb der ersten Stunde nach deren Eintreffen gelesen wurden, die innerhalb der zweiten Stunde nach deren Eintreffen gelesen wurden, die innerhalb der dritten Stunden gelesen wurden usw.

Abbildung 4.15a zeigt die Verteilung dieser Zeitspannen in Form eines Histogramms. Die hellblaue Linie in diesem Diagramm dient lediglich einer leichteren Sortierung der Datenpunkte durch den Betrachter. Es lässt sich erkennen, dass mit 953 E-Mails der Großteil der E-Mails innerhalb der ersten Stunde gelesen wurden. Die sehr flach auslaufenden rechten Datenpunkte bedeuten, dass es jedoch einige wenige E-Mails gibt, die erst sehr spät gelesen wurden. Die in diesem Experiment maximal gemessene Zeitspanne beträgt 3 832 Stunden und entspricht ca. 160 Tagen.

Abbildung 4.15b zeigt ebenfalls die Verteilung der Zeitspannen. Diesmal ist jedoch der prozentuale Anteil an der Gesamtmenge der E-Mails gegen die gemessenen Zeitspannen aufgetragen. Des Weiteren sind in dieser Darstellung beide Achsen logarithmisch skaliert. In dieser Darstellung kommen die Unterschiede in den kleinen Wertebereichen beider Achsen besser zur Geltung. Auffällig ist die gerade, diagonale Grundausrichtung der Datenpunkte in dieser logarithmischen Darstellung. Dies deutet auf eine Verteilung der gemessenen Zeitspannen hin, die einem Potenzgesetz (engl. *power law*) folgt [New05].

Aufgrund der geringen Anzahl an zur Verfügung stehenden E-Mails ist das Rauschen in den Bereichen mit einer hohen Stundenzahl jedoch sehr hoch. Dies liegt vor allem daran, dass die meisten Benutzer ihre E-Mails innerhalb einer kurzen Zeit lesen und nur wenige E-Mails nach einer eher größeren Zeitspanne gelesen werden. Verdeutlicht wird dies durch das 0,75-Quantil der Verteilung, das mit 18 Stunden noch deutlich unterhalb der Ein-Tages-Grenze liegt. Ausreißer in den hohen Stundenzahlen erhalten durch die geringe Anzahl an spät gelesenen E-Mails somit relativ viel Gewicht und verursachen dadurch das Rauschen in der logarithmischen Darstellung.

Eine übliche Methode dieses Rauschen zu entfernen, ist eine geschachtelte Einteilung der gemessenen Stunden. Anstatt einer gleichmäßigen Einteilung der x-Achse, fasst man mehrere Stunden zu einem Intervall zusammen und betrachtet die Anzahl der E-Mails, die innerhalb der entsprechenden Intervalle gelesen wurden. Die daraus resultierenden Häufigkeiten werden anschließend mit Hilfe einer Division durch die



(a) Lineare Achsen **(b)** Logarithmische Achsen

Abbildung 4.15.: Stundengenaue Verteilung der Reaktionszeiten
Das Diagramm (a) zeigt die Verteilung der gemessenen Zeitspannen auf einer linearen Skala. Das Diagramm in Teil (b) hingegen ist logarithmisch skaliert und trägt den Anteil der E-Mails gegen die gemessenen Zeitspannen auf.

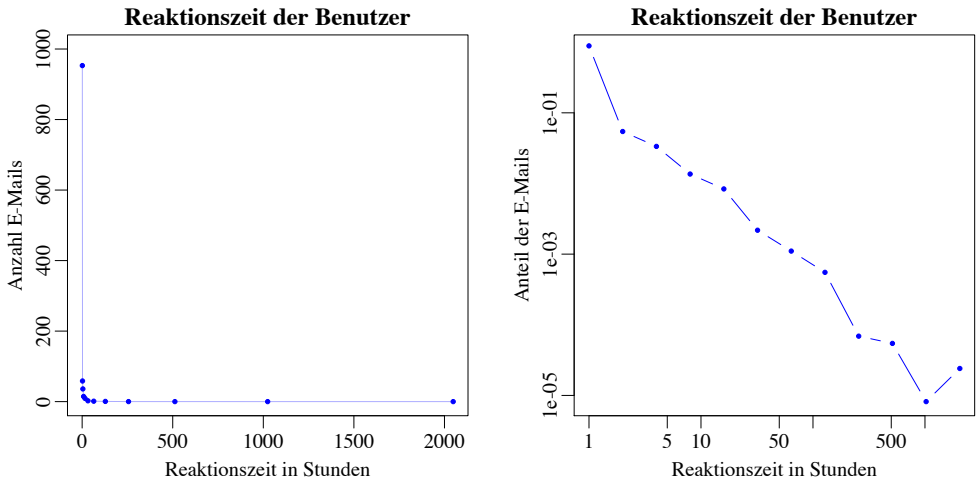
Länge der jeweiligen Intervalle normalisiert. Dadurch fallen Ausreißer nicht mehr so ins Gewicht, da diese durch die anderen Werte in ihrem Intervall *geglättet* werden. Oftmals wird eine variable Länge der Intervalle gewählt, bei der ein Intervall immer doppelt so lang ist wie das vorherige Intervall [New05].

Das folgende Beispiel verdeutlicht dieses Vorgehen: Anstatt einer gleichmäßigen Einteilung der x-Achse in die Stundenintervalle $[0-1), [1-2), [2-3), [3-4)$ usw., hat man nun eine Einteilung der x-Achse in die Intervalle $[0-1)$, $[1-3)$, $[3-7)$, $[7-15)$ usw. Bei der anschließenden Normalisierung würde man in diesem Beispiel also durch die Werte 1, 2, 4, 8 usw. dividieren.

Das Ergebnis solch einer geschachtelten Einteilung der x-Achse ist in beiden Teilen der Abbildung 4.16 dargestellt. Aufgrund der Einteilung der Stunden in logarithmische Intervalle bestehen die beiden Diagramme der Abbildung 4.16 aus weniger Datenpunkten als die beiden Diagramme der Abbildung 4.15. Die Datenpunkte befinden sich jeweils am Anfang der verwendeten Intervalle – also bei den x-Werten 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024 und 2048 (Stunden).

Abbildung 4.16a zeigt die Verteilung der gesuchten Zeitspannen bei einer gleichmäßigen Einteilung der Stunden und einer linearen Skalierung der Achsen. In Abbildung 4.16b sind die Stunden in Intervalle logarithmisch anwachsender Größe zusammengefasst, jedoch ist dieses Diagramm zusätzlich auch noch logarithmisch skaliert (weshalb in dieser Darstellung die Datenpunkte auch alle den gleichen horizontalen Abstand haben). Durch die logarithmische Einteilung der gemessenen Zeitspannen wird das Rauschen herausgefiltert und die diagonale Grundausrichtung kommt nun noch wesentlich besser zur Geltung als in Abbildung 4.15b.

Zur Überprüfung, ob die gemessenen Zeitspannen auch tatsächlich einem Potenzgesetz unterliegen, haben wir die freie GNU-Statistik-Software *R* [JMR09] eingesetzt.



(a) Lineare Achsen **(b)** Logarithmische Achsen

Abbildung 4.16.: Verteilung der Reaktionszeiten mit einer geschachtelten Einteilung
Das Diagramm (a) zeigt die Verteilung der gemessenen Zeitspannen auf einer linearen Skala. Das Diagramm in Teil (b) hingegen ist logarithmisch skaliert und trägt den Anteil der E-Mails gegen die gemessenen Zeitspannen auf. Im Gegensatz zur Abbildung 4.15 sind hier nun in beiden Teilen der Abbildung die gemessenen Stunden in normalisierte Intervalle mit logarithmisch zunehmender Größe zusammengefasst.

Für diese Software hat der amerikanische Professor Rick Wash[3] ein Zusatzpaket geschrieben, mit dessen Hilfe sich gemessene Werte auf eine Verteilung abbilden lassen, die einem Potenzgesetz folgt. Solch eine Verteilung besitzt folgende Wahrscheinlichkeitsdichte:

$$p(x) = C \cdot x^{-\alpha},\ \alpha \in \mathbb{R}_{>0}$$

Dabei wird der Parameter α der *Exponent* der Verteilung genannt. Der Vorfaktor C dient lediglich zur Erfüllung der Bedingung, dass der Inhalt der gesamten Fläche unterhalb der Dichtefunktion den Wert 1 annimmt.

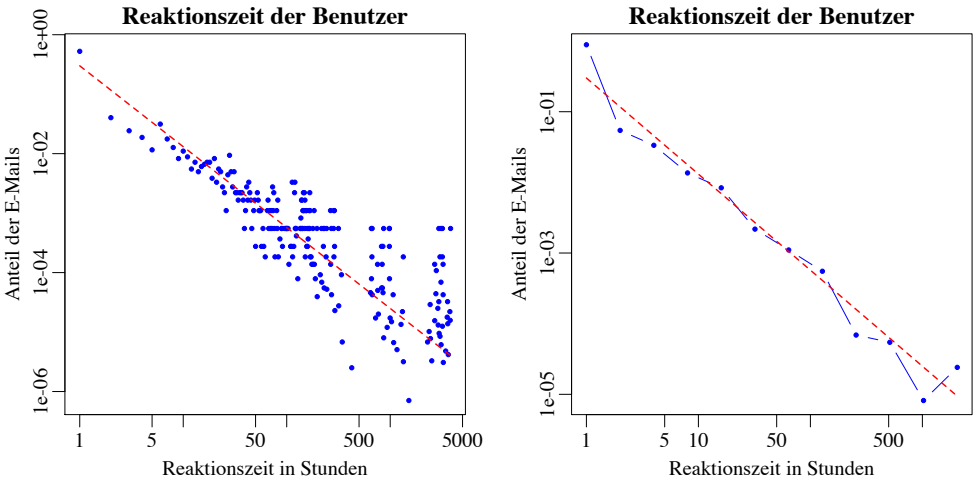
Mit Hilfe des Zusatzpaketes von Wash lässt sich bestimmen, dass die Verteilung der gemessenen Zeitspannen einer Verteilung nach dem Potenzgesetz mit den Parametern

$$\alpha = 1{,}2821 \quad \text{und} \quad C = 0{,}312$$

entspricht. Des Weiteren bietet das Zusatzpaket die Möglichkeit, einen *Kolmogorov-Smirnov-Test* durchzuführen [Mas51, GMY04]. Dieser gibt unter anderem Auskunft darüber, ob Messwerte einer zuvor angenommenen Wahrscheinlichkeitsverteilung folgen. Für die Nullhypothese „*Die gemessenen Daten entsprechen einer Verteilung nach einem Potenzgesetz*“ ergibt dieser Test einen *p-Wert* von ca. 0,2521. Mit einer üblichen Festlegung des Signifikanzniveaus auf 5 % [Hor77] bedeutet dies, dass die Nullhypothese nicht verworfen werden kann. Somit kann aus den gemessenen Zeitspannen nicht gefolgert werden, dass diese keiner solchen Verteilung folgen. Abbildung 4.17 zeigt abschließend die logarithmisch skalierten Diagramme der beiden vorherigen Ab-

[3] *http://www.rickwash.com/*

bildungen, bei denen zusätzlich die Dichtefunktion einer Verteilung eingezeichnet ist, die einem Potenzgesetz mit den beiden berechneten Parametern folgt.



(a) Lineare Einteilung der Stunden

(b) Logarithmische Einteilung der Stunden

Abbildung 4.17.: Verteilung der Reaktionszeiten inkl. Verteilung nach dem Potenzgesetz Teil (a) zeigt die Verteilung der gemessenen Zeitspannen mit einer linearen Einteilung der Stunden und das Diagramm in Teil (b) mit einer Einteilung der Stunden in logarithmische Zeitintervalle. In beiden Teilen der Abbildung ist zusätzlich die Dichtefunktion einer Power-Law-Verteilung mit den berechneten Parametern α und C eingezeichnet.

4.4. Reparaturzeiten eines infizierten Systems

Ein weiterer wichtiger Faktor, der bei der Verbreitung von Social Malcode eine Rolle spielt, ist die Geschwindigkeit, mit der ein Schädling wieder von einem infizierten System entfernt wird. Diesen Vorgang bezeichnen wir im Folgenden als *Säuberung* oder *Reparatur* eines Systems. Vor allem bei Botnetzen, deren Bots den Befehl zur Weiterverbreitung erhalten, nimmt dieser Faktor einen wesentlichen Einfluss auf die Verbreitung des Botnetzes: Je schneller die befallenen Rechner wieder gesäubert werden, desto weniger Bots erhalten im Schnitt den Befehl zur Weiterverbreitung und desto langsamer kann sich das Botnetz ausbreiten. Ein Schädling, der sofort nach der Infektion eines Rechners weitere Rechner angreift, wird durch schnelle Reparaturzeiten nicht in seiner Verbreitung verlangsamt. Dennoch ist es natürlich so, dass zu einem beliebigen Zeitpunkt weniger Rechner mit einem Schädling infiziert sind, wenn befallene Rechner schnell gesäubert werden, als zum gleichen Zeitpunkt, unter der Annahme, dass eine Säuberung nur sehr zeitverzögert vonstattengeht. Diese Beispiele verdeutlichen die Wichtigkeit der Reparaturzeiten eines befallenen Systems im Bezug auf die Verbreitung eines Schädlings.

Um Aufschluss über das Verhalten von Computerbenutzern bei der Säuberung eines infizierten Systems zu erhalten, greifen wir auf Daten einer im Jahr durchgeführten Studie 2008 [HEF09] zurück. Von April bis Oktober haben wir diese durchgeführt, um die finanziellen Schäden zu untersuchen, die durch eine neue Art von Keyloggern (sie-

he Seite 12) verursacht werden. Im Speziellen handelt es sich dabei um Keylogger, die für die Kommunikation mit dem Angreifer von sogenannten *Dropzones* Gebrauch machen. Dies sind verborgene Speicherorte im Internet – oftmals öffentlich beschreibbare Verzeichnisse eines über das Internet erreichbaren Servers.

Genutzt werden die Verzeichnisse zum Datenaustausch zwischen Keylogger und Angreifer, indem ein Keylogger dort die aufgezeichneten Daten ablegt. Später verbindet sich ein Angreifer ebenfalls zu der Dropzone, liest das entsprechende Verzeichnis aus und erhält so die vom Keylogger mitgeschnittenen Daten. Innerhalb der Verzeichnisse ist häufig eine Webanwendung installiert, über die unter anderem auch ein rudimentärer Zugangsschutz implementiert ist und die eine Oberfläche anbietet, um bequem durch die gestohlenen Daten zu navigieren.

Bei den von uns untersuchten Keylogger-Familien liegen die Daten im Klartext vor und für jedes eindeutig identifizierbare Opfer existieren eine oder mehrere Textdateien, die die mitgeschnittenen Tastatureingaben enthalten. Jede Textdatei ist wiederum in verschiedene Datensätze unterteilt, die in der Regel einer Internetsitzung (oftmals beziehen sich die Datensätze auch nur auf eine einzelne Internetseite) des entsprechenden Opfers gleichkommen. Neben den aufgezeichneten Tastatureingaben enthält solch ein Datensatz auch jeweils einen Zeitstempel. Diese Zeitstempel erlauben Rückschlüsse auf die Zeitspanne zwischen der Infektion eines Computers durch den Keylogger und seiner Säuberung – also der *mittleren Reparaturzeit* (engl.: *Mean Time To Repair*) des Computers.

In den folgenden Abschnitten wird zuerst diese spezielle Sorte der Dropzone-basierten Keylogger formalisiert, anschließend stellen wir die in der Studie verwendete Methodik und die erzielten/gefundenen Analyseergebnisse vor. Abschnitt 4.4.3 widmet sich der Funktionsweise der beiden von uns analysierten Keylogger-Familien und erläutert unter anderem auch den Aufbau und die Struktur der bereits angesprochenen Datensätze. Abschließend präsentiert Abschnitt 4.4.4 eine Auswertung der durch die Studie zur Verfügung stehenden Datensätze, mit dem Ziel, die mittleren Reparaturzeiten der betroffenen Systeme zu ermitteln.

4.4.1. Dropzone-basierte Keylogger

Der Ablauf eines Angriffs mit dieser Art von Keyloggern ist in Abbildung 4.18 dargestellt. In einem ersten Schritt infiziert ein Angreifer A die Opfer O_1, O_2 und O_3 mit der gerade beschriebenen Art von Keyloggern. Jedes Opfer O_i besitzt spezielle Zugangsdaten Z_{i_j} zur Anmeldung bei Online-Dienstleistern D_j. Beispielsweise ist der Dienstleister D_1 der Abbildung eine Bank, bei der sich die Kunden mit ihrer Kontonummer und einem Passwort zum Online-Banking anmelden können, und der Dienstleister D_2 ist ein Freemailer, auf dessen Internetseite E-Mail-Konten erstellt und E-Mails gelesen und geschrieben werden können. Die Keylogger stehlen durch Mitschneiden von Tastatureingaben die Zugangsdaten der Opfer. Anschließend senden sie die Zugangsdaten an eine Dropzone, wo diese von dem Angreifer ausgelesen werden können. In der Regel nutzen Angreifer die gestohlenen Zugangsdaten für einen Identitätsdiebstahl, d. h. Angreifer A benutzt die gestohlenen Zugangsdaten Z_{i_j} und gibt sich somit bei dem Dienstleister D_j als Benutzer O_i aus. Des Weiteren verkaufen sie die gestohlenen Zu-

gangsdaten auch auf einer Art Untergrundmarkt, der sich speziell durch den Handel mit digitalen Diebesgütern etabliert hat [Fin08, Sec07, Sec08, Sta07].

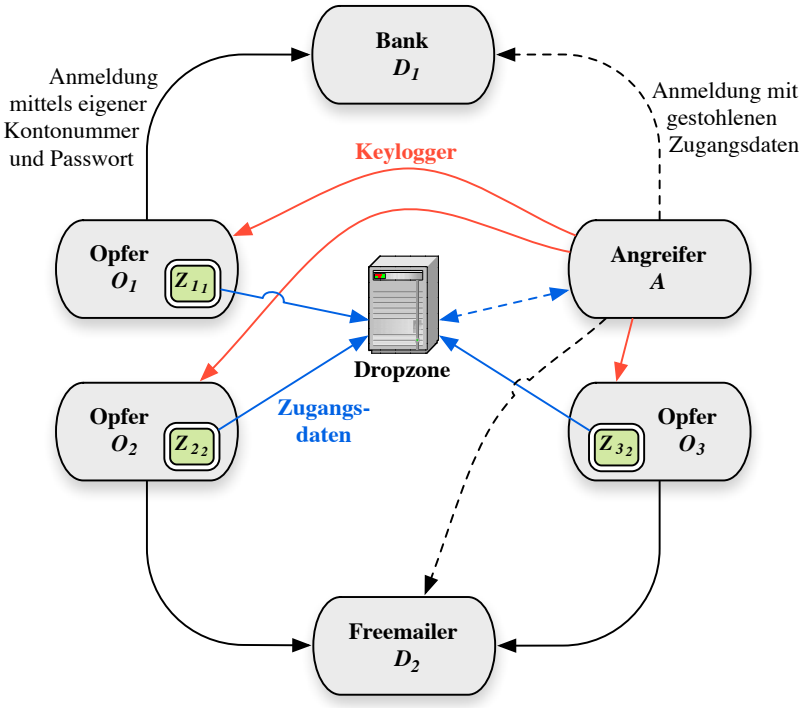


Abbildung 4.18.: Angriffsablauf eines Dropzone-basierten Keyloggers
Nachdem ein Angreifer den Opfern die Keylogger hat zukommen lassen, zeichnen diese Schadprogramme die Tastatureingaben an den infizierten Systemen auf und übermitteln die so gestohlenen Daten an eine Dropzone.

4.4.2. Analysemethodik und -resultate

Zur Analyse solcher Keylogger-basierten Angriffe haben wir zuerst mit Hilfe von verschiedenen Techniken, wie zum Beispiel *Honeypots* [Pro04] und Spamtraps, mehrere Keylogger-Instanzen gesammelt und diese anschließend innerhalb einer kontrollierten Umgebung ausgeführt und dynamisch analysiert [WHF07]. Durch die dynamische Analyse war es uns möglich, die URLs der verwendeten Dropzones zu identifizieren. In einem letzten Schritt haben wir die extrahierten URLs benutzt, um die auf den Dropzones hinterlegten Daten auszulesen – genauso wie dies auch ein regulärer Angreifer machen würde. Die so extrahierten Daten haben wir auf ein lokales Analysesystem kopiert und dort ausgewertet.

Auf diese Weise konnten wir in dem siebenmonatigen Zeitraum insgesamt ca. 33 GB mitgeschnittene Keylogger-Daten sammeln, die bei mehr als 70 verschiedenen Dropzones hinterlegt waren. Diese Daten setzen sich aus gestohlenen Zugangsdaten von mehr als 173 000 kompromittierten Rechnern zusammen. Beispielsweise befinden sich in den Daten mehr als 10 700 Zugänge für Online-Banking und mehr als 149 000 Anmeldedaten zu Freemail-Konten. Basierend auf einer von *Symantec* durch-

geführten Studie [Sym08], die sich speziell mit Untergrundmärkten für digitale Handelsgüter und deren Preise beschäftigt, berechnet sich der maximale Gesamtwert der durch uns gesammelten Daten auf ca. 16,5 Millionen US-Dollar.

4.4.3. Keylogger-Familien

Im Rahmen der Studie haben wir mit der gerade beschriebenen Methodik zwei verschiedene Keylogger-Familien genauer analysiert, die zum Zeitpunkt der Studie weitverbreitet waren: *Nethell* und *ZeuS*. In diesem Abschnitt liefern wir einen kurzen Überblick über die Arbeitsweise dieser beiden Keylogger-Familien und zeigen anhand von exemplarischen Datensätzen, in welcher Form die beiden Familien ihre aufgezeichneten Daten an die Dropzones übermitteln.

4.4.3.1. Nethell

Keylogger der Familie *Nethell* (auch bekannt unter dem Namen *Limbo*) verbreiten sich üblicherweise über Drive-By-Downloads. Potentielle Opfer werden durch Techniken des Social Engineering dazu gebracht, die entsprechenden Internetseiten zu besuchen. Nethell ist als Browser Helper Object implementiert und hat somit über eine vom Internet Explorer zur Verfügung gestellte Schnittstelle Zugriff auf das Document Object Model der aktuell geladenen Internetseite. Auf diese Weise kann Nethell sehr leicht Eingabefelder in Web-Formularen identifizieren, die zur Eingabe von sensiblen Daten, wie etwa Passwörtern oder Kreditkartennummern, verwendet werden. Des Weiteren arbeiten Keylogger der Nethell-Familie mit Konfigurationsdateien. Diese befinden sich ebenfalls auf den Dropzones und können zur Laufzeit von dort nachgeladen werden. Innerhalb der Konfigurationsdateien kann ein Angreifer zum Beispiel eine Liste von Internetseiten angeben, so dass der Keylogger nur noch auf diesen Seiten aktiv ist. Auf diese Weise verringert sich die Datenmenge und der Angreifer erhält zudem keine Daten mehr, die auf für ihn uninteressanten Internetseiten aufgezeichnet wurden.

Nach dem Aufzeichnen der Daten, überträgt Nethell diese über HTTP-Anfragen zur Dropzone. Auf dieser befinden sich PHP-Skripte, die die Anfragen entgegennehmen und die Daten auf dem Server ablegen. Zum Beispiel wird die Anfrage *http://example.org/datac.php?userid=21102008_110432_2025612* dazu benutzt, ein neu infiziertes Opfer auf der Dropzone zu registrieren. Der Parameter *userid* encodiert neben einer zufälligen Opfer-ID (letzter Zahlenblock) im ersten Zahlenblock ebenfalls das Infektionsdatum (in diesem Beispiel der 21.10.2008) und im mittleren Zahlenblock die Uhrzeit der Infektion (in diesem Beispiel 11:04:32 Uhr). Durch eine dynamische Analyse des Keyloggers lässt sich anhand dieser HTTP-Anfragen auf eine einfache Art und Weise die Dropzone lokalisieren.

Eine Nethell-Dropzone ist als einfache Webanwendung implementiert und ermöglicht dem Angreifer unter anderem eine komfortable Verwaltung der aufgezeichneten Daten. Auch lassen sich die Daten in der Webanwendung bequem durchsuchen oder nach bestimmten Kriterien filtern und sortieren. Über die Webanwendung kann der Angreifer die Keylogger der kompromittierten Maschinen zudem dazu veranlassen, bestimmte Dateien aus dem Internet herunterzuladen und auszuführen.

Abbildung 4.19 zeigt beispielhaft drei Arten von Datensätzen, wie sie von Keyloggern der Nethell-Familie zu Dropzones übermittelt werden. Alle drei Datensätze sind anonymisiert, damit keine Rückschlüsse auf die beteiligten Opfer möglich sind und somit deren Identität und Privatsphäre geschützt bleiben.

```
(a) 01 *******PROTECTED STORAGE*******
    02 Resource: https://ssologon.bankofamerica.com/
    03 Description: AutoComplete Passwords
    04 Username: XXXk8p3
    05 Password: kissXXX
    06
    07 Resource: http://www.myspace.com/
    08 Description: AutoComplete Passwords
    09 Username: cutelaXXX@live.com
    10 Password: alleXXX

(b) 01 Timestamp:08.07.2008 22:12:25
    02 [http://login.live.com/login.srf?wa=wsignin1.0[...]
    03 login=KEYLOGGED:leear2XXX.edu KEYSREAD:leear@XXX.edu
    04 [http://login.live.com/login.srf?wa=wsignin1.0[...]
    05 passwd=KEYLOGGED:04uciXX KEYSREAD:04uciXX
    06 [https://login.live.com/ppsecure/post.srf?wa=wsignin1.0[...]
    07 Sign In
    08 PPSX=Pass
    09 PwdPad=IfYouAreReadingThisYouHaveTooMuchF
    10 login=leear@XXX.edu
    11 passwd=04uciXX
    12 LoginOptions=2
    13 PPFT=Bzy3u0imy4JP6WjmRQISrSb3fwF7WDxCHfhHEHZTg74Hzktqk
    14 IP=XXX.181.104.92
    15 ID=08052008_155842_32185906

(c) 01 Timestamp:08.07.2008 18:07:54
    02 ************COOKIES************
    03 http://www.google.com/ig
    04 PREF=ID=a0567315adb4edXX:TB=5:TM=1137730467:LM=1208466064:
    05 S=4RBtzgEjOVgwH9wP; NID=12=IsSRM11qplYNeeFneEaVGt1F5LUF[...]
```

Abbildung 4.19.: Von Nethell aufgezeichnete Datensätze (anonymisiert)
Teil (a) zeigt extrahierte Zugangsdaten aus dem Protected Storage, Teil (b) mitgeschnittene Zugangsdaten bei der Anmeldung auf der Kommunikationsplattform *live.com* und Teil (c) ein gestohlenes Cookie der Domain *www.google.com*. Alle drei Datensätze sind anonymisiert, damit keine Rückschlüsse auf die beteiligten Opfer möglich sind.

Teil (a) der Abbildung zeigt einen Datensatz, der extrahierte Daten aus dem *Protected Storage* des Betriebssystems enthält. Dies ist ein geschützter Speicherplatz für vertrauliche Daten wie Passwörter oder private Schlüssel. Teil (b) der Abbildung zeigt die aufgezeichneten Daten einer speziellen Internetseite – in diesem Fall die Anmeldedaten der Kommunikationsplattform *live.com*. Der Benutzername wurde in diesem Beispiel in das Eingabefeld *login* (Zeile 10) eingegeben und das Passwort in das Eingabefeld *passwd* (Zeile 11). Der letzte Teil der Abbildung zeigt ein Beispiel für ein

gestohlenes Cookie der Domain *www.google.com*. Bei den Datensätzen der beiden Teile (b) und (c) ist jeweils in der ersten Zeile der Zeitpunkt angegeben, zu dem die entsprechenden Daten mitgeschnitten beziehungsweise ausgelesen wurden. Dies sind die Zeitstempel, die es uns später erlauben, die gesuchten mittleren Reparaturzeiten der betroffenen Systeme zu berechnen.

4.4.3.2. ZeuS

Keylogger der Familie *ZeuS* (auch bekannt unter den Namen *Zbot* und *Wsnpoem*) verbreiten sich über Dateianhänge in E-Mails. Auch hier wird wieder versucht, die Opfer durch Techniken des Social Engineering zum Öffnen des Dateianhangs zu bewegen – beispielsweise indem in der E-Mail behauptet wird, dass es sich bei dem Dateianhang um eine elektronische Rechnung handelt. Im Gegensatz zu den eher simplen Techniken der Nethell-Familie, nutzen Keylogger der ZeuS-Familie technisch anspruchsvollere und raffiniertere Techniken, um die Zugangsdaten der Opfer auszulesen: ZeuS injiziert Code in jeden laufenden Benutzerprozess und verbirgt durch verschiedene Techniken seine Existenz auf den befallenen Rechnern. Sobald sich ZeuS erfolgreich in den Windows-Standardbrowser *Microsoft Internet Explorer* injiziert hat, fängt er abgesetzte HTTP-Post-Anfragen ab, um aus diesen eventuell vorhandene Zugangsdaten auszulesen. Ebenfalls unterstützen Keylogger der ZeuS-Familie das Auslesen von gespeicherten Cookies und Zugangsdaten aus dem Windows-eigenen Protected Storage. Genau wie Nethell-Keylogger schicken auch Keylogger der Familie ZeuS die ausgelesenen Daten in regelmäßigen Abständen über HTTP-Anfragen an eine Dropzone. Diese ist ebenso als Webanwendung implementiert, kann die Daten aber sowohl im Dateisystem des Dropzone-Servers als auch in einer Datenbank ablegen.

Auch die dynamische Konfiguration des Keyloggers zur Laufzeit ist beiden Keylogger-Familien gemein: Nach der Ausführung verbinden sich ZeuS-Keylogger zur Dropzone, um von dort die aktuelle Konfiguration zu beziehen. In den Konfigurationsdateien kann ein Angreifer sowohl Internetseiten angeben, von denen der Keylogger die Zugangsdaten stehlen soll, als auch Seiten, deren Benutzereingaben explizit ignoriert werden sollen. Des Weiteren sind Keylogger der ZeuS-Familie auch in der Lage, bei einem Mausklick Bildschirmfotos der Größe 50×50 Pixel rund um die Position des Mauszeigers aufzunehmen und diese ebenfalls an die Dropzone zu senden. Dadurch wird ein Schutzmechanismus umgangen, der es einem Benutzer ermöglicht, sensible Daten nicht über die normale Tastatur, sondern über eine visuelle Bildschirmtastatur mit Hilfe der Maus einzugeben. Solch eine Technik der Benutzereingabe wird auf verschiedenen Online-Banking-Plattformen angeboten und sichert die eingegebenen Daten gegen das Mitlesen durch einen einfachen Keylogger ab – Keylogger der ZeuS-Familie registrieren durch die Bildschirmfotos jedoch auch diese Form der Dateneingaben. In den Konfigurationsdateien kann ein Angreifer auch Internetseiten angeben, auf die ein *Man-in-the-middle*-Angriff (MITM) durchgeführt werden soll: Jedes Mal, wenn ein Opfer solch eine Seite besucht, wird der Datenverkehr transparent über einen vom Angreifer kontrollierten Server übertragen. Auf diesem befindet sich in der Regel eine Phishing-Seite. Dadurch ist der Angreifer in der Lage, weitere Zugangsdaten der Opfer zu stehlen, die in der Eingabemaske der Originalseite nicht abgefragt werden. Darüber hinaus können Keylogger der ZeuS-Familie unter anderem die Na-

mensauflösung der befallenen Rechner modifizieren und beeinflussen oder auch das Aktualisierungsverhalten der installierten Software und des Betriebssystems ändern.

Abbildung 4.20 zeigt drei Arten gestohlener Datensätze, die durch einen ZeuS-Keylogger ausgelesen wurden. Auch in dieser Abbildung sind die Daten wieder anonymisiert, um Rückschlüsse auf die entsprechenden Opfer zu verhindern. Die Reihenfolge der gestohlenen Daten entspricht der vorherigen Abbildung: Teil (a) zeigt ausgelesene Zugangsdaten aus dem Protected Storage und Teil (b) enthält im Internet Explorer abgefangene Benutzereingaben auf der Seite *wachovia.com*. Der Benutzername und das Passwort zur Anmeldung wurden bei diesem Beispiel in den HTTP-Post-Parametern *userid* (Zeile 14) und *password* (Zeile 15) übertragen und ausgelesen. Der letzte Teil (c) der Abbildung zeigt ein durch ZeuS ausgelesenes Cookie der Domain *google.com*.

```
(a) 01 [0002] VERSION: 1548 ------------------------------------------------
    02 System time: 09.05.2008 09:37:12, GMT: -4.00, Login time: 00:02:03
    03 Version: 5.1.2600 SP1, Language: 1033
    04 Process: C:\WINDOWS\system32\services.exe
    05 -
    06 Protected Storage:
    07 https://login.comcast.net/login = acesharXXX|downdXXX
    08 https://my.screenname.aol.com/_cqr/login/login.psp = biglamXX|laXX
    09 http://www.myspace.com = XXcardXX@students.rowXX.edu||yveXX83

(b) 01 [0739] VERSION: 1289 ------------------------------------------------
    02 System time: 18.06.2008 03:07:55, GMT: -4.00, Login time: 00:05:43
    03 Version: 5.1.2600 SP2, Language: 1033
    04 Process: C:\Program Files\Internet Explorer\iexplore.exe
    05 -
    06 https://onlineservices.wachovia.com/auth/AuthService
    07 Referer: -
    08 Keys: www.wachovia.com  julXXX  mckinXXX  minimXXX  july2XXX
    09 Data:
    10 action=uidLogin
    11 bi=version%3D1%26pm_fpua%3Dmozilla%2F4.0 %28compatible%3B msie 7.0
       windows nt 5.1%3B .net clr 1.0.3705%3B .net clr 1.1.4322%3Bmedia
       center pc 4.0%29%7C4.0 %28compatible%3B MSIE 7.0%3B Windows[...]
    12 requestTimestamp=1213758461593
    13 homepage=yes
    14 userid=julXXXmckinXXX
    15 password=july2XXX
    16 systemtarget=gotoOSH
    17 -

(c) 01 IE Cookies:
    02 Path: google.com/
    03 PREF=ID=d9361e071b3ebeXX:TM=1214675597:LM=1214675597:S=COFeRz[...]
    04
    05 Path: yahoo.com/
    06 B=3eju00t46cv1v&b=3&s=au
```

Abbildung 4.20.: Von ZeuS aufgezeichnete Datensätze (anonymisiert)
Teil (a) zeigt extrahierte Zugangsdaten aus dem Protected Storage, Teil (b) mitgeschnittene Zugangsdaten bei der Anmeldung auf der Domain *wachovia.com* und Teil (c) ein gestohlenes Cookie der Domain *www.google.com*. Alle drei Datensätze sind anonymisiert, damit keine Rückschlüsse auf die beteiligten Opfer möglich sind.

Auch in den gestohlenen Zugangsdaten dieses ZeuS-Beispiels ist in zwei von drei Datensätzen ein Zeitstempel (in Teil (a) und (b) der Abbildung jeweils die zweite Zeile) vorhanden, zu dem die entsprechenden Daten ausgelesen wurden. Da sich ZeuS in die Benutzerprozesse der befallenen Rechner injiziert, befindet sich in den Datensätzen auch ein Eintrag, der den Prozess angibt, innerhalb welchem ZeuS die Daten ausgelesen hat. Diese Angabe befindet sich in den Beispielen der Abbildung in der vierten Zeile der Teile (a) und (b).

4.4.4. Ergebnisse

Aus allen Datensätzen der über 173 000 Opfer haben wir, falls vorhanden, die Zeitstempel extrahiert, zu denen der entsprechende Datensatz aufgezeichnet wurde. Mit Hilfe dieser Zeitstempel haben wir anschließend die Reparaturzeit des Computers eines Opfers O_i berechnet, indem wir als Infektionsdauer die Zeit zwischen dem ersten Zeitstempel $t_{i_{first}}$ und dem letzten Zeitstempel $t_{i_{last}}$ seiner Datensätze festgelegt haben. Im Folgenden verwenden wir die Begriffe *Infektionsdauer* und *Reparaturzeit* synonym, da die Zeitspanne zwischen einer Infektion und der Säuberung eines Rechners der Reparaturzeit entspricht, während derer der Computer mit dem entsprechenden Schädling infiziert ist.

Wie bereits in der Einleitung zu diesem Abschnitt erwähnt, erhält man auf diese Weise nicht die genauen Reparaturzeiten, vielmehr lassen sich Rückschlüsse ziehen, die diese Zeiten eingrenzen. Dies hat verschiedene Gründe:

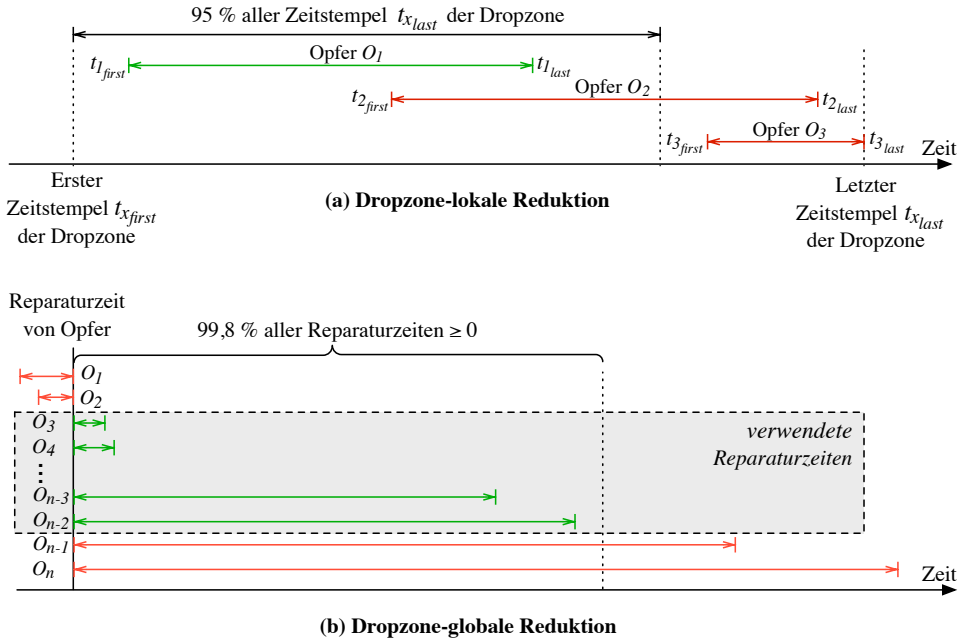
- Der erste Zeitstempel entspricht zwar genau dem Zeitpunkt der Infektion, jedoch ist das Ende der Infektion nicht mit absoluter Gewissheit durch den letzten Zeitstempel zu bestimmen. Dieser besagt lediglich, dass ein Computer zu diesen Zeitpunkt noch infiziert war und die Säuberung des Computers erst nach diesem Zeitpunkt stattgefunden hat. Wie zeitnah dies nach dem zuletzt gefundenen Zeitstempel passierte, lässt sich anhand der vorliegenden Daten nicht herleiten.
- Ebenfalls muss die Tatsache, dass nach dem zuletzt übermittelten Datensatz eines Opfers keine weiteren Datensätze des Opfers mehr zur Dropzone gesendet wurden, nicht dem Umstand geschuldet sein, dass der entsprechende Rechner gesäubert wurde. Dies kann auch daran liegen, dass die Dropzone nicht mehr erreichbar war oder der entsprechende Rechner (unabhängig von der Infektion) nicht mehr benutzt wurde.
- Opfern, von denen nur ein Datensatz mit einem Zeitstempel zur Dropzone übertragen wurde, wird eine Reparaturzeit von null Stunden zugewiesen.
- Des Weiteren entsprechen die übermittelten Zeitstempel der lokalen Systemzeit eines infizierten Rechners. Wurde diese während der Dauer der Infektion verändert oder konnte nicht richtig vom Keylogger ausgelesen werden, so entspricht die von uns ermittelte Infektionsdauer nicht der tatsächlichen Infektionsdauer. Zum Beispiel existieren in den ZeuS-Datensätzen einige Zeitstempel, die als Jahresangabe das Jahr 1601 oder auch das Jahr 2120 enthalten.

Um diesen Problemen entgegenzuwirken, haben wir die extrahierten Reparaturzeiten einer Filterung unterzogen. Ziel der Filterung ist es, die Reparaturzeiten zu identifizieren und zu ignorieren, die den gerade beschriebenen Ungenauigkeiten unterliegen.

Somit wird die Anzahl der Reparaturzeiten, die in die Berechnung der mittleren Reparaturzeiten einfließen, reduziert. Zur Filterung der Daten haben wir drei verschiedene Reduktionen durchgeführt:

1. In den weiteren Berechnungen haben wir nur Datensätze von Opfern betrachtet, von denen mindestens zwei Datensätze zu einer Dropzone übertragen wurden. Diese Beschränkung hat den Grund, dass für Opfer mit nur einem Datensatz die gesuchte Reparaturzeit nicht zu bestimmen ist. Entsprechenden Rechnern würde automatisch eine Reparaturzeit von null Stunden zugewiesen, d. h. die Opfer infizieren sich mit dem Keylogger und im selben Augenblick würde der infizierte Rechner auch wieder gesäubert. Da dies ein realitätsfremder Zustand ist (beziehungsweise müsste diskutiert werden, ob eine Infektionsdauer der Länge *Null* überhaupt als Infektion zu betrachten ist), werden diese Opfer bei den weiteren Berechnungen vernachlässigt.
2. Durch Dropzone-lokale Reduktionen haben wir die Reparaturzeiten von Opfern herausgefiltert, deren letzter Zeitstempel nah am zeitlichen Ende einer Dropzone liegt. Als zeitliches Ende einer Dropzone erachten wir den allerletzten Zeitstempel eines Datensatzes, der an die entsprechende Dropzone übermittelt wurde. Es werden also nur die Zeitstempel einer einzelnen Dropzone miteinander verglichen. Aus diesem Grund haben wir diese Reduktion *Dropzone-lokal* genannt. Grafisch dargestellt ist diese Reduktionsart im Teil (a) der Abbildung 4.21. Die beiden Zeitstempel $t_{x_{first}}$ und $t_{x_{last}}$ der Abbildung beziehen sich nicht unbedingt auf ein einzelnes Opfer, sondern beziehen sich auf alle ersten und letzten Zeitstempel der Opfer der entsprechenden Dropzone. Somit wird durch die Dropzone-lokale Reduktion der Einfluss (auf die gesuchte mittlere Reparaturzeit) derjenigen Opfer verringert, deren Reparaturzeiten lediglich durch eine mögliche Abschaltung der Dropzone sehr gering ausfallen. Wie in der Abbildung dargestellt, haben wir alle Reparaturzeiten vernachlässigt, deren letzter Zeitstempel innerhalb der letzten 5 % aller letzten Zeitstempel der Dropzone liegt. In der Abbildung sind exemplarisch die Reparaturzeiten von drei Opfern O_1, O_2 und O_3 dargestellt. Von diesen drei Opfern würde dementsprechend nur die Reparaturzeit des ersten Opfers O_1 in die Berechnung der mittleren Reparaturzeit einfließen. Diese Dropzone-lokale Reduktion haben wir für jede einzelne Dropzone durchgeführt.
3. Nach den Dropzone-lokalen Reduktionen führen wir zusätzlich eine Dropzone-globale Reduktion durch, die in Teil (b) der Abbildung 4.21 dargestellt ist. Dabei werden sowohl die ermittelten, negativen Reparaturzeiten (aller Dropzones) als auch mögliche Ausreißer nach oben gestrichen. Als Ausreißer nach oben definieren wir die Reparaturzeiten, die länger andauern als das Quantil $Q_{0,998}$ der Verteilung aller nicht negativen Reparaturzeiten (aller Dropzones). Durch diese Form der Reduktion beseitigen wir den verzerrenden Effekt, der durch eine falsch ausgelesene oder veränderte Systemzeit entsteht. Alle übrig gebliebenen Reparaturzeiten werden verwendet, um die mittlere Reparaturzeit zu berechnen beziehungsweise um deren Verteilung zu analysieren.

Nach den drei beschriebenen Reduktionen bleiben 109 141 Reparaturzeiten übrig, die in die Berechnung der mittleren Reparaturzeiten einfließen. Aus diesen haben wir



(a) Dropzone-lokale Reduktion

(b) Dropzone-globale Reduktion

Abbildung 4.21.: Qualitative Reduktion der Keylogger-Datenmenge
Um den Effekt von Reparaturzeiten zu verringern, die lediglich aufgrund einer nicht mehr erreichbaren Dropzone sehr gering sind, führen wir pro Dropzone eine Dropzone-lokale Reduktion durch, die in Teil (a) dargestellt ist. Teil (b) zeigt die Dropzone-globale Reduktion, bei der negative und außergewöhnlich lange Reparaturzeiten herausgefiltert werden.

ein Histogramm erstellt, das in Abbildung 4.22 durch die durchgehende schwarze Linie dargestellt ist. Auf der x-Achse ist die Dauer der Reparaturzeiten in Stunden aufgetragen und die y-Achse enthält die Wahrscheinlichkeit für die jeweilige Reparaturzeit in den uns vorliegenden Keylogger-Daten. Die gestrichelte blaue Linie entspricht einer Exponentialverteilung mit der Rate $\lambda = 0,01486432$, die durch eine Maximum-Likelihood-Schätzung auf den gemessenen Zeiten bestimmt wurde. Der Erwartungswert der mittleren Reparaturzeit beträgt somit:

$$\frac{1}{\lambda} \approx 67,28 \text{ (Stunden)}$$

Bei den anlysierten Keylogger-Daten ist ein Computer also durchschnittlich ca. 2,8 Tage mit einem Keylogger infiziert, bevor er wieder gesäubert wird. Der Median der Verteilung liegt bei 25 Stunden und die maximale Reparaturzeit betrug 1 711 Stunden.

Auffällig ist das schwingende Muster in der linken, unteren Ecke des Histogramms. Dieser Bereich ist in einer Vergrößerung in Abbildung 4.23 zu sehen, bei der die x-Achse auf Werte im Intervall $[0, 240]$ (Stunden) und die y-Achse auf Werte im Intervall $[0,\ 2 \times 10^{-2}]$ beschränkt wurde. Die vertikalen Linien entsprechen jeweils einem Vielfachen von 24 Stunden. In den vorliegenden Daten nutzen die Keylogger-Opfer also in regelmäßigen, zeitlichen Abständen ihre Computer – etwa immer abends nach der täglichen Arbeit. Dementsprechend findet sich diese regelmäßige Nutzung auch in Form eines schwingenden Musters mit einer Periode von 24 Stunden in den gesuchten Reparaturzeiten wieder.

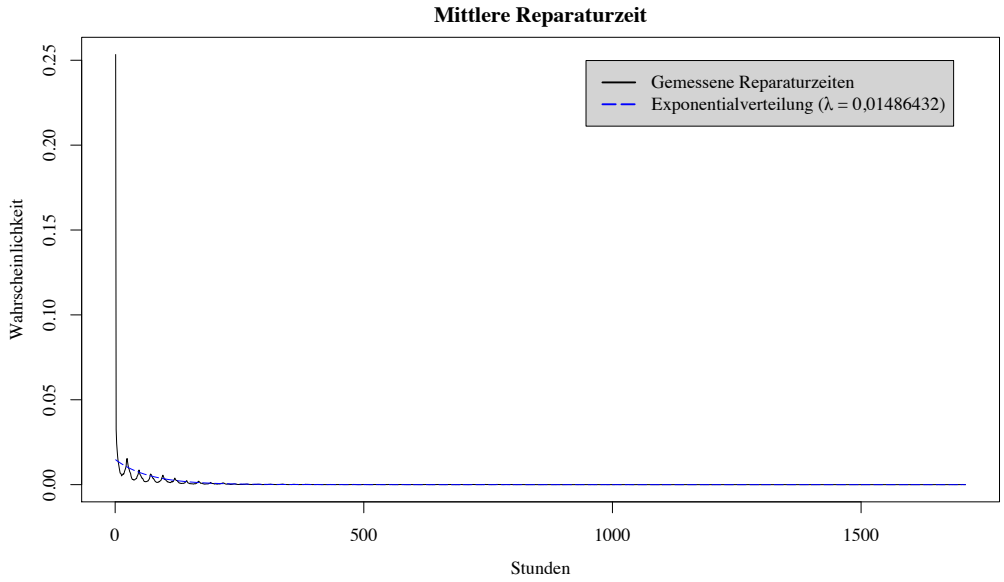


Abbildung 4.22.: Mittlere Reparaturzeiten der Keylogger-Opfer
Die schwarze Linie entspricht einem Histogramm der mittleren Reparaturzeiten der befallenen Rechnern. Die blaue, gestrichelte Linie ist die Exponentialverteilung, deren Rate sich durch eine Maximum-Likelihood-Schätzung auf den berechneten Zeitspannen ergibt.

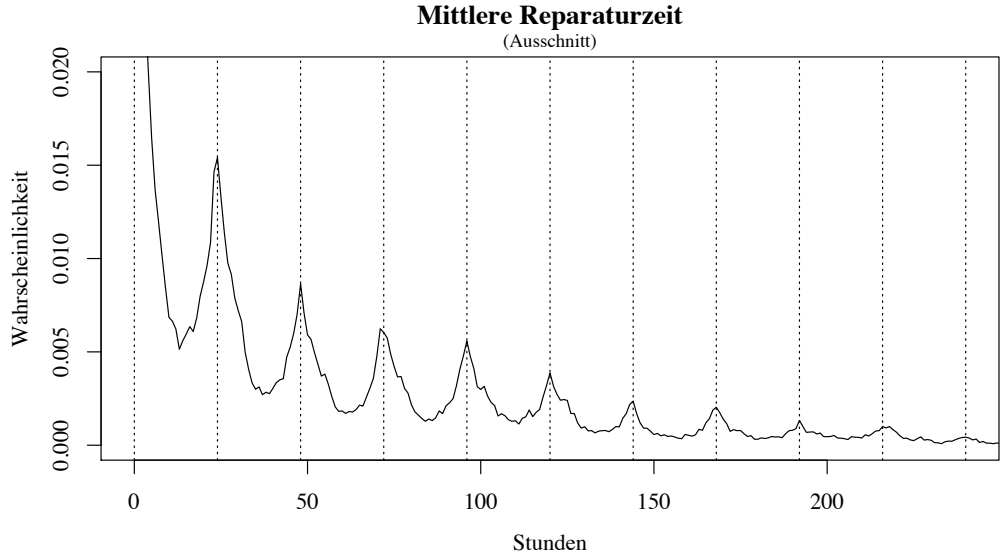


Abbildung 4.23.: Schwingendes Muster in den berechneten Reparaturzeiten
Diese Abbildung enthält einen Ausschnitt der mittlere Reparaturzeiten, wie sie in Abbildung 4.22 dargestellt sind. Die regelmäßige Nutzung der Computer spiegelt sich in einem schwingenden Muster mit einer Periode von 24 Stunden wieder.

4.5. E-Mail-Spam

Neben dem Verhalten der Opfer spielt natürlich auch das Verhalten und die Handlungen eines Angreifers eine wesentliche Rolle bei der Verbreitung von Social Malcode. Beispielsweise wirken sich die Antworten folgender Fragen auf die Verbreitung aus: Welche Angriffsvektoren werden von einem Angreifer verwendet? Wie häufig verschickt ein Angreifer Spam-Mails? Welchen Inhalt haben diese Spam-Mails?

Zur Beantwortung dieser Fragen haben wir Spam-Mails eingehender untersucht. Als Spam-Mails bezeichnet man üblicherweise E-Mails, die

a) unaufgefordert und

b) massenhaft an einen unspezifischen Empfängerkreis

verschickt werden [CL98]. Da der Versand von E-Mails beim Absender kaum Kosten verursacht, werden Spam-Mails häufig genutzt, um (meist dubiose) Produkte zu bewerben oder Schadprogramme zu verbreiten. Weitere Vorteile für den Absender sind, dass es beim Medium E-Mail kaum zu Verzögerungen kommt und er einen großen Personenkreis adressieren kann.

Dieser Abschnitt ist eine Beschreibung des Spam-Korpus, der dazu dient das Verhalten eines Angreifers zu untersuchen. Zu diesem Zweck werden einige Statistiken und Analyseergebnisse bezüglich der gesammelten Spam-Mails präsentiert, die dem Leser ein Gefühl für die Charakteristik dieser Datenquelle vermitteln sollen.

4.5.1. Verwendeter Spam-Korpus

Ursprüngliche sollten mehrere *Spamtraps* als Datenquelle dienen, die wir im Rahmen eines Forschungsprojektes aufgesetzt hatten, um Spam-Mails zu sammeln. Spamtraps sind E-Mail-Postfächer ohne produktiven Einsatzzweck, es wird also kein regulärer E-Mail-Verkehr erwartet. Somit kann jede eingehende E-Mail laut obiger Definition als Spam angesehen werden [EFG+09a]. Leider konnte dieser Datenbestand aufgrund von Hardware-Problemen nicht weiter genutzt werden, so dass eine andere Quelle verwendet werden musste. An dieser Stelle möchten wir uns nochmals bei Chris Morrow von *ops-netman.net* bedanken, der uns freundlicherweise Zugriff auf deren Spam-Korpus gewährt hat.

Gesammelt wurden die Spam-Mails dieses Korpus an 12 in den USA registrierten Domains (siehe Tabelle 4.3). Für jede dieser Domains ist ein sogenanntes *Catch-All-Postfach* eingerichtet. In diesen laufen alle E-Mails zusammen, die an eine E-Mail-Adresse dieser Domains verschickt wurden, welche jedoch in der entsprechenden Domain nicht oder nicht mehr registriert ist. Ist zum Beispiel innerhalb der Domain *as701.net* die E-Mail-Adresse *max.mustermann@as701.net* nicht eingerichtet und wird dennoch eine E-Mail an diese Adresse versandt, so landet die E-Mail im Catch-All-Postfach der Domain. Voraussetzung dafür ist natürlich, dass dieses „Auffang"-Postfach aktiviert ist. Dies ist bei allen 12 gelisteten Domains der Fall.

Tabelle 4.3 zeigt, dass über einen vierwöchigen Zeitraum insgesamt über 60 GB an Spam-Mails in diesen Catch-All-Postfächern eingetroffen sind. Über 17 Millionen Spam-Mails wurden an über 1 Million unterschiedliche Empfänger verschickt. Summiert man die Anzahl der Spam-Mails der einzelnen Domains auf, so erkennt man,

Tabelle 4.3.: Eigenschaften des Spam-Korpus
Die Tabelle enthält neben allgemeinen Charakteristiken des Spam-Korpus auch spezielle Kennzahlen bezüglich der Empfänger und den in den Spam-Mails enthaltenen Dateianhängen und Verweisen.

Allgemein	
Zeitraum des Sammelns	20.05.2011 – 16.06.2011
Datenmenge	ca. 60,19 GB
Anzahl Spam-Mails insgesamt	17 100 844
davon entfallen auf folgende Domains...	
as701.net	2 328 498
notellbooks.org	78
notellmotel.org	775 970
oe-consulting.com	2 543 894
ops-netman.net	3 074 297
rarc.net	215 213
rebaroni.com	3 215 177
reblivingston.net	1 932 401
secsup.com	1 833 212
secsup.net	975 591
secsup.org	431 157
velvet-tigers.org	196 931
Empfänger	
Anzahl Empfänger	38 901 685
Anzahl unterschiedlicher Empfänger	1 051 644
Durchschnittliche Anzahl Empfänger pro Spam-Mail	ca. 2,30
Maximale Anzahl Empfänger pro Spam-Mail	400
Durchschnittliche Anzahl Spam-Mails pro Empfänger	ca. 36,99
Maximale Anzahl Spam-Mails pro Empfänger	13 723
Dateianhänge	
Anzahl Dateianhänge	545 002
Anzahl unterschiedlicher Dateianhänge	62 059
Anzahl Spam-Mails mit mindestens einem Dateianhang	528 004
Maximale Anzahl an Dateianhängen pro Spam-Mail	28
Verweise	
Anzahl Verweise	10 571 790
Anzahl unterschiedlicher Verweise	5 454 901
Anzahl Spam-Mails mit mindestens einem Verweis	6 173 540
Maximale Anzahl an Verweisen pro Spam-Mail	221

dass diese Zahl um 421 555 Spam-Mails höher ist als die in der Tabelle angegebene Gesamtzahl der Spam-Mails. Dies liegt daran, dass manche Spam-Mails mehrere Empfänger hatten und diese aus mehr als nur einer der angegebenen Domains stammen. Solche Spam-Mails sind demzufolge auch mehreren Domains zuzuordnen.

4.5.2. Zeitliche Aspekte

Abbildung 4.24 zeigt die Verteilung der Spam-Mails über die Wochentage. Jeder der sieben Balken repräsentiert einen Wochentag und ist unterteilt in vier farbige Segmente, die jeweils den vier Wochen des Zeitraums entsprechen, in dem die Spam-Mails gesammelt wurden. Links in jedem Balken befindet sich die erste und rechts die vierte Woche. Rechts neben den Balken steht jeweils die Anzahl an Spam-Mails, die an dem jeweiligen Wochentag über den gesamten Zeitraum gesammelt wurden.

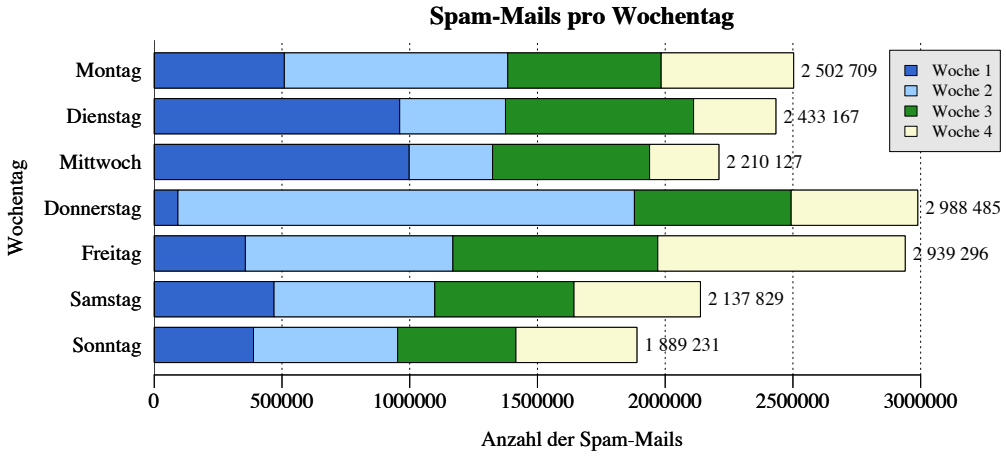


Abbildung 4.24.: Spam-Mails pro Wochentag
Diese Abbildung zeigt die Verteilung der Spam-Mails über die Wochentage. Jeder Balken besteht aus vier Segmenten, die jeweils eine Woche des Zeitraums repräsentieren, in dem die Spam-Mails gesammelt wurden.

Anhand der Segmente erkennt man, dass die Verteilung der Spam-Mails über die Wochentage (zumindest in dem von uns betrachteten vier Wochen) keiner Regelmäßigkeit unterliegt: In der ersten Woche treffen die meisten Spam-Mails mittwochs ein, in der zweiten Woche donnerstags und in den letzten beiden Wochen freitags. Über den gesamten Zeitraum betrachtet wurden an den beiden Tagen des Wochenendes die wenigsten Spam-Mails verschickt. Gegen Ende einer Arbeitswoche hingegen mit Abstand am meisten: Donnerstags trafen fast 1,6-mal so viele Spam-Mails in den Postfächern ein als sonntags – hauptsächlich bedingt durch die zweite Woche.

Die Verteilung der Spam-Mails über die Stunden eines Tages ist in Abbildung 4.25 dargestellt. Die rote, gestrichelte Linie ist der zirkulierende, gleitende Durchschnitt der Ordnung 5. An diesem erkennt man besonders gut den tendenziellen Verlauf über einen Tag: Nachts treffen die wenigsten Spam-Mails ein und über den Tag hin, bis 16 Uhr, nimmt die Anzahl zu. Danach treffen dann kontinuierlich weniger Spam-Mails in den Catch-All-Postfächern der Domains ein. Der Mittelwert der Verteilung

ist in der Abbildung durch die vertikale, schwarze Linie gekennzeichnet und liegt bei 12:08 Uhr (dies entspricht ca. 12,14 Stunden). Die relativ hohe Standardabweichung von ca. 6,71 Stunden, in der Abbildung durch die graue Fläche auf beiden Seiten des Mittelwerts dargestellt, macht jedoch deutlich, dass sich keine Tageszeit von anderen Tageszeiten bezüglich der Menge von eintreffenden Spam-Mails besonders abhebt.

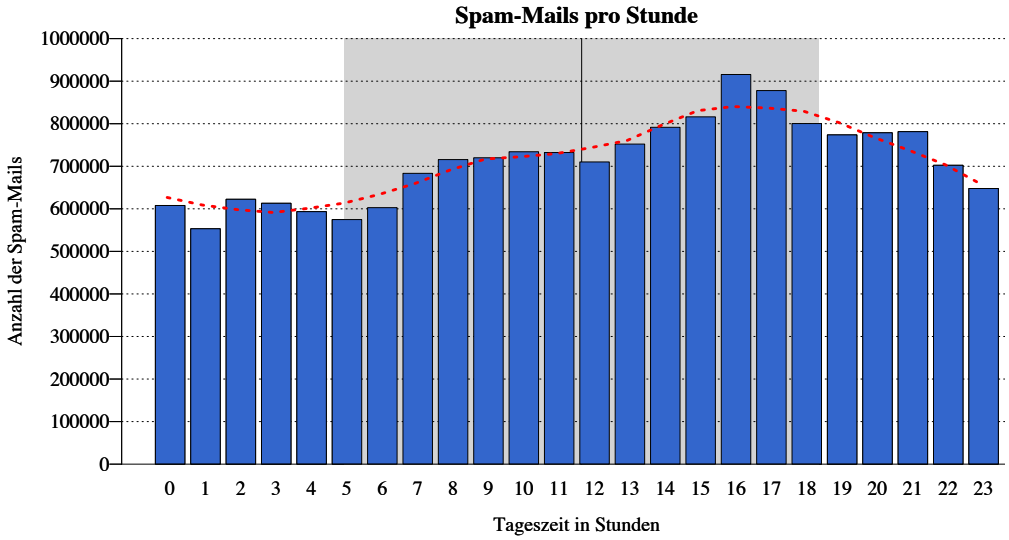


Abbildung 4.25.: Spam-Mails pro Stunde
Die Abbildung zeigt die Verteilung der Spam-Mails über die Stunden eines Tages. Die vertikale Linie stellt den Mittelwert, die graue Fläche die Standardabweichung in beide Richtungen und die gestrichelte Linie den gleitenden Durchschnitt der Zahlenreihe dar.

4.5.3. Statistiken zu den Dateianhängen

Insgesamt enthält der Spam-Korpus 528 004 Spam-Mails mit mindestens einem Dateianhang. Diese Spam-Mails wiederum enthalten 62 059 unterschiedliche Dateianhänge, wobei als Unterscheidungskriterium der SHA-1-Hashwert des Anhangs verwendet wird. Tabelle 4.4 zeigt die 11 häufigsten *Multipurpose Internet Mail Extensions* Typen (MIME) aller unterschiedlichen Dateianhänge. In der Tabelle erkennt man, dass über 93 % aller Dateianhänge GIF-Bilder sind. Der zweit- und dritthäufigste Typ der Dateianhänge sind Word-Dokumente (ca. 3,07 %) und JPEG-Bilder (ca. 2,19 %). Ungefähr 0,62 % aller Dateianhänge besitzen den MIME-Typ *application/octet-stream*, der unter anderem auch ausführbaren Dateien (*.exe) eines Windows-Betriebssystems zugeordnet wird. In den vorhandenen Spam-Mails sind alle Dateianhänge mit diesem MIME-Typ solche ausführbaren Windows-Dateien. Dateien mit anderen MIME-Typen werden eher nur sehr selten als Dateianhang verwendet.

Betrachtet man die Häufigkeit, wie oft die einzelnen MIME-Typen insgesamt in den Spam-Mails verwendet werden, so ergibt sich eine andere Verteilung, die in Abbildung 4.26 dargestellt ist. Während sich die Werte aus Tabelle 4.4 auf die 62 059 unterschiedlichen Dateianhänge beziehen, nehmen die Prozentzahlen dieser Abbildung Bezug auf die Gesamtmenge aller Spam-Mails, die mindestens einen Dateianhang be-

Tabelle 4.4.: MIME-Typen der eindeutig unterscheidbaren Dateianhänge
Diese Tabelle enthält die am häufigsten im Spam-Korpus vorkommenden MIME-Typen. Falls ein Dateianhang in mehreren E-Mails verwendet wurde, wurde dessen MIME-Typ nur einmal gezählt.

MIME-Typ	Anzahl
image/gif	58 139
application/msword	1 906
image/jpeg	1 356
application/octet-stream	387
application/x-zip-compressed	95
application/vnd.ms-excel	33
image/png	27
application/x-pkcs7-signature	26
application/zip	25
image/pjpeg	22
application/pdf	16
andere	*27*

sitzen. Die Abbildung zeigt, dass über die Hälfte der Spam-Mails mit mindestens einem Dateianhang dazu benutzt werden, ein Archiv (*application/zip*) zu verschicken. Insgesamt befinden sich nur 25 unterschiedliche Archive dieses MIME-Typs in der Menge der eindeutig unterscheidbaren Anhänge (siehe Tabelle 4.4), d. h. einige, seltene Dateianhänge werden sehr oft per Spam-Mails verschickt. Bilder kommen hingegen in weniger als zwei Fünftel aller Spam-Mails mit Dateianhang vor.

Tabelle 4.5 listet die 10 am häufigsten in den Spam-Mails verwendeten Dateianhänge auf. Der Dateianhang der am häufigsten verwendet wurde ist ein Bild, das vier angebliche Arzneimittel gegen Erektionsstörungen bewirbt (siehe Abbildung 4.27). Die folgenden Einträge der Tabelle sind der Grund für den hohen Prozentsatz des MIME-Typs *application/zip* aus Abbildung 4.26: Dem ersten Eintrag folgen sieben Archive. Eine Analyse dieser Archive mit Hilfe des Online-Analyse-Dienstes *VirusTotal* [His11] ergibt, dass diese Schadprogramme aus den Familien *Chepvil* [MC11b], *Oficla* [MC11c] und *Bredo* [MC11a] enthalten. Alle diese Schadprogramme sind sogenannte *Downloader*, die während ihrer Ausführung weitere Schadprogramme aus dem Internet auf den infizierten Rechner nachladen und diese anschließend starten.

4.5.4. Statistiken zu den Verweisen

Die Verteilung der Top Level Domains (TLD) der insgesamt ca. 5,5 Millionen Verweise, die aus den Spam-Mails extrahiert werden konnten, zeigt Abbildung 4.28. Mit 37 % und 30 % zeigen die meisten Verweise auf Internetseiten mit nicht-gesponserten Domains [Int10], die ursprünglich nur für Unternehmen (.com) und Informationsanbieter (.info) gedacht waren, mittlerweile aber frei zugänglich sind. Auch die viert und fünft meist genutzten TLDs sind nicht-gesponserte TLDs, die ursprünglich für Netz-

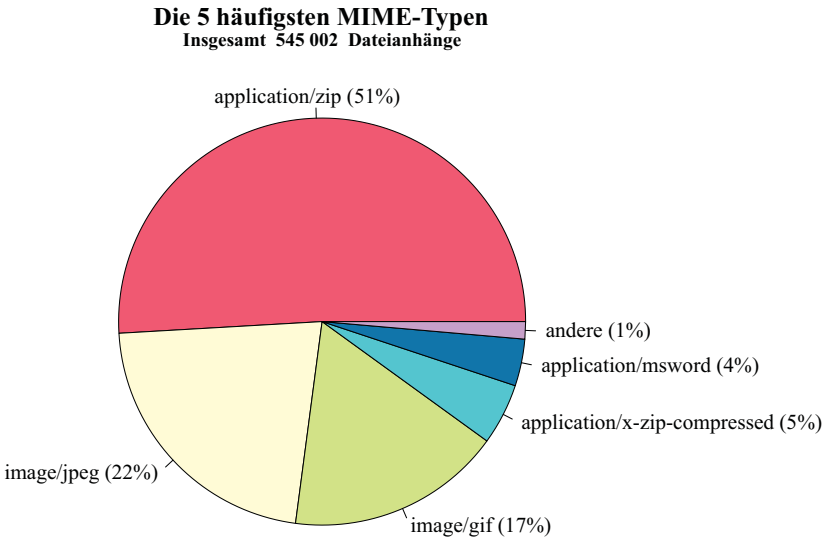


Abbildung 4.26.: Verteilung der MIME-Typen der Dateianhänge
Diese Abbildung zeigt wie häufig Dateien mit bestimmten MIME-Typen einer E-Mail angehangen sind. Die Prozentzahlen beziehen sich somit auf die Gesamtzahl aller Spam-Mails mit mindestens einem Dateianhang.

Tabelle 4.5.: Die 10 häufigsten Dateianhänge
Der am häufigsten verwendete Dateianhang ist ein JPEG-Bild. Der Großteil der verschickten Anhänge sind jedoch Archive, die einen Downloader für weitere Schadprogramme enthalten.

Anzahl Spam-Mails	MIME-Typ	Häufigster Dateiname	VirusTotal-Erkennungsrate	VirusTotal-Befund
59 902	image/jpeg	foto1.jpg	0 / 43	–
57 279	application/zip	UPS_Document.zip	35 / 42	Chepvil
42 274	application/zip	Postal_Document#54905.zip	38 / 42	Oficla
37 630	application/zip	UPS_Document.zip	34 / 40	Chepvil
34 752	application/zip	UPS_Document.zip	33 / 42	Chepvil
32 395	application/zip	UPS_Document.zip	17 / 42	Chepvil
26 570	application/zip	FedEx mail.zip	22 / 43	Bredo
23 975	application/zip	UPS_Document.zip	35 / 42	Chepvil
20 883	image/jpeg	x300.jpg	0 / 41	–
14 361	image/jpeg	x301.jpg	0 / 43	–

Abbildung 4.27.: Der am häufigsten verwendete Dateianhang
Dies ist der am häufigsten verwendete Dateianhang. Dieses JPEG-Bild wird dazu benutzt, Arzneimittel zu bewerben, die angeblich bei Erektionsstörungen helfen sollen.

verwaltungseinrichtungen (.net) und nicht-kommerzielle Organisationen (.org) vorgesehen waren. Die einzige länderspezifische TLD, die häufig in den Verweisen des Spam-Korpus zu finden ist, ist die russische TLD (.ru), die von ca. 20 % aller Verweise benutzt wird. Die Differenz zwischen der Gesamtzahl der Verweise mit einer TLD (5 433 804 – siehe Untertitel der Abbildung 4.28) und der Gesamtzahl aller unterschiedlichen Verweise (5 454 901 – siehe Tabelle 4.3) kommt daher, dass ca. 20 000 Verweise keinen Domain-Namen, sondern eine IP-Adresse zur Adressierung des Zielservers nutzen.

4.6. Zusammenfassung

Die Verbreitung von Social Malcode hängt zu einem wesentlichen Teil von dem Verhalten der beteiligten Personen ab – also von dem Verhalten des Angreifers und dem Verhalten der möglichen Opfer. Um in den späteren Kapiteln die Verbreitung von Social Malcode zu simulieren, ist es notwendig auch das Verhalten dieser Personen modellieren zu können. Dieses Kapitel stellt Methoden und Techniken vor, deren Ziel es ist, dieses Verhalten zu erfassen und zu quantifizieren.

Das klassische Instrument der empirischen Sozialwissenschaften zum Quantifizieren von Verhaltensweisen sind Umfragen und Interviews. Diese sind Jedoch mit einigen Problemen behaftet, die die Bewertung des Wahrheitsgehaltes der Antworten erschwert. Beispielsweise haben die *soziale Erwünschtheit*, der *Reihenfolgeneffekt* und

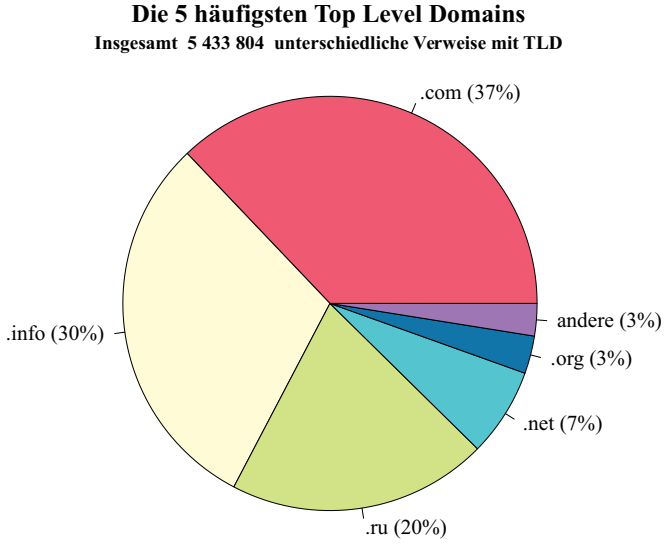


Abbildung 4.28.: Die 5 häufigsten Top Level Domains
Die Abbildung zeigt die Verteilung der am häufigsten genutzten Top Level Domains innerhalb der unterschiedlichen Verweise, die aus den Spam-Mails extrahiert werden konnten.

die Formulierung der Fragen oftmals einen verfälschenden Einfluss auf die Antworten der befragten Personen.

Um dieses Problem zu umgehen, haben wir das Verhalten der beteiligten Benutzer in mehreren Experimenten quantifiziert. In Zusammenarbeit mit einem Unternehmen aus der IT-Sicherheitsbranche haben wir mehrere Mail-Experimente durchgeführt. Dabei wurden Spam-Mails an die Mitarbeiter mehrerer Kunden (mittelständische deutsche Unternehmen) des IT-Sicherheitsunternehmens verschickt und anschließend über verschiedene Rückkanäle das Verhalten der Empfänger analysiert. Für jeden Kunden wurden jeweils Spam-Mails mit einem Verweis, mit einem Dateianhang, mit einem Verweis auf eine ausführbare Datei und eine Phishing-Mail verschickt. Daraus resultierend können wir die Wahrscheinlichkeiten für das Folgen eines Verweises, das Öffnen eines Dateianhangs, das Herunterladen und Ausführen einer ausführbaren Datei und das Preisgeben von kundeninternen Zugangsdaten ermitteln.

Neben diesen Wahrscheinlichkeiten spielt auch die Reaktionszeit der Benutzer eine große Rolle bei der Verbreitung von Social Malcode. Diese stellt gewissermaßen einen zeitlichen Puffer bei der Verbreitungsgeschwindigkeit von Social Malcode dar. Je schneller potentielle Opfer beispielsweise einen schädlichen Dateianhang ausführen, desto schneller verbreitet sich der entsprechende Schadcode. Zum Quantifizieren dieser Reaktionszeiten haben wir ein Python-Skript geschrieben, das die Reaktionszeiten direkt am Mailserver der Universität Mannheim ausliest. Es zeigt sich, dass diese einer Power-Law-Verteilung folgen: Der Großteil der bei diesem Experiment beteiligten Benutzer liest neu eintreffende E-Mails sehr schnell (innerhalb einer Stunde),

einige wenige E-Mails werden jedoch erst recht spät nach deren Eintreffen im Posteingang gelesen.

Um in der abschließenden Simulation von Social Malcode auch den Effekt einfließen zu lassen, der durch die Säuberung eines infizierten Systems entsteht, muss ebenfalls das Verhalten der Benutzer nach der Infektion eines Rechners untersucht werden. Diesbezüglich haben wir Daten ausgewertet, die durch Keylogger aufgezeichnet wurden. Da die aufgezeichneten Daten jeweils mit einem Zeitstempel markiert sind, lässt sich dadurch eine Analyse der Reparaturzeiten der infizierten Systeme vornehmen. In den uns vorliegenden Daten sind diese Zeiten exponentialverteilt.

Abschließend haben wir einen Spam-Korpus analysiert. Daraus lassen sich Verhaltensweisen der Angreifer ableiten. Beispielsweise ist für die Verbreitung von Social Malcode relevant wie häufig ein Angreifer Spam-Mails verschickt, welchen Inhalt diese Spam-Mails haben oder auch welche Angriffsvektoren von einem Angreifer verwendet werden. Der von uns verwendete Spam-Korpus besteht aus insgesamt 17 100 844 Spam-Mails, die in einem Zeitraum von vier Wochen gesammelt wurden. Die Spam-Mails wurden an 12 in Amerika registrierten Domains gesammelt und waren an insgesamt 1 051 644 unterschiedliche Empfänger adressiert. Innerhalb der Mails werden 5 454 901 unterschiedliche Verweise verwendet und es sind 62 059 unterschiedliche Dateianhänge eingebettet.

KAPITEL V

Simulative Berechnung von Verbreitungsverläufen

Im vorherigen Kapitel wurden einige Experimente und Methoden vorgestellt, deren Ziel es war, das Verhalten eines Benutzers in Situationen zu untersuchen, die möglicherweise zu einer Infektion durch Social Malcode führen können. Mit Hilfe dieser Erkenntnisse wird in diesem Kapitel die Verbreitung von Social Malcode in Form einer umfangreichen Simulation analysiert. Vor allem der Einfluss der einzelnen Simulationsparameter steht im Fokus dieses Kapitels.

Das gesamte Kapitel ist folgendermaßen strukturiert: In Abschnitt 5.1 wird zunächst die verwendete Simulationssoftware OMNeT++ vorgestellt, ehe in Abschnitt 5.2 die eigentlich Simulation von Social Malcode präsentiert wird, die mittels OMNeT++ implementiert wurde. In diesem Abschnitt werden neben dem simulierten Schadprogramm auch die Struktur und das Verhalten der konkreten Simulation präzisiert. Speziell in letzteres, also in das Simulationsverhalten, fließen unmittelbar die Ergebnisse des vorangegangenen Kapitels ein. Abgeschlossen wird das vorliegende Kapitel durch die Präsentation und Diskussion der Simulationsergebnisse in Abschnitt 5.3.

5.1. OMNeT++

OMNeT++ [VH08] ist ein objektorientiertes, modulares und diskret ereignisbasiertes Rahmenwerk zur Simulation von Rechnernetzen und Netzwerk-Protokollen. Dieses Rahmenwerk ist keine konkrete Simulation an sich, sondern stellt lediglich verschiedenste Werkzeuge und Komponenten zur Verfügung, die es einem Benutzer erlauben Simulationen zu erstellen. Die Komponenten werden in Abschnitt 5.1.3 genauer vorgestellt.

Ursprünglich wurde OMNeT++ im Jahr 1992 von András Varga entwickelt. Die momentane Weiterentwicklung wird durch die Firma *OpenSim Ltd.* vorangetrieben, welche auch die offizielle Internetseite *http://www.omnetpp.org* betreibt und an der Varga ebenfalls beteiligt ist. Auf dieser Internetseite befindet sich eine ausführliche Dokumentation, die unter anderem eine Bedienungs- und Installationsanleitung, eine Beschreibung der auf Eclipse basierenden, integrierten Entwicklungsumgebung (IDE, von engl. *integrated development environment*) und die Beschreibung einer Program-

mierschnittstelle (API, von engl. *application programming interface*) enthält. Zudem finden sich dort auch Mailinglisten, die hilfreiche Tipps zur Benutzung von OMNeT++ bieten.

Von OMNeT++ existieren englischsprachige Versionen für Windows (XP, Win2K, Vista, 7), Linux, Mac OS X (ab 10.5) und andere Unix-ähnliche Betriebssysteme – aktuell ist Version 4.3 (Stand: Mai 2013). Dadurch, dass OMNeT++ als Open-Source-Projekt unter der *Academic Public License* entwickelt wurde, ist es für gemeinnützige Organisationen und die akademische Lehre und Forschung kostenlos verwendbar. Eine kommerziell nutzbare Version von OMNeT++ wird unter dem Namen *OMNEST* angeboten.

OMNeT++ ist so konzipiert, dass es einfach möglich ist, die Funktionalität des Simulations-Rahmenwerks durch Plug-ins und externe Rahmenwerke zu erweitern. Viele dieser Plug-ins sind kostenlos über die Internetseite von OMNeT++ zu beziehen. Beispielsweise finden sich dort Plug-ins, welche die Simulation von drahtlosen Netzen oder Mobilfunknetzen ermöglichen, oder Plug-ins zur Simulation der bekanntesten Netzprotokolle wie etwa IP, TCP, UDP oder Ethernet.

5.1.1. Network Description (NED)

OMNeT++ bietet dem Benutzer eine eigene Sprache zur Beschreibung der Simulationsstruktur – die sogenannte *Network Description* (NED). Mithilfe dieser Sprache wird ausschließlich die Struktur beziehungsweise die Topologie der zu erstellenden Simulation festgelegt, das eigentliche Verhalten der einzelnen, an der Simulation beteiligten Module wird nicht innerhalb dieser Beschreibung festgelegt. Jede mittels OMNeT++ erstellte Simulation beinhaltet mindestens eine Datei, welche die Dateierweiterung `.ned` haben muss und welche die Struktur des zu simulierenden Netzes beschreibt.

Die Syntax der NED-Sprache beinhaltet verschiedene Schlüsselwörter zur Beschreibung der einzelnen, an der Simulation beteiligten Module. Jedem dieser Module können Eigenschaften zugewiesen werden und es kann festgelegt werden wie diese untereinander verbunden sind und wie diese miteinander kommunizieren. NED ermöglicht es zudem, sehr komplexe Module aus einzelnen Untermodulen zusammenzusetzen. Auf diese Weise können einmal erstellte Module auch in anderen Simulationen wiederverwendet werden. Die in Abschnitt 5.1 angesprochenen Plug-ins beinhalten beispielsweise auch Bibliotheken beziehungsweise Pakete bereits geschriebener Modulbeschreibungen.

Auch das aus objektorientierten Programmiersprachen bekannte Prinzip der Vererbung ist innerhalb der NED-Sprache anwendbar, so dass neue Module von bereits bestehenden Modulen abgeleitet werden können. Eine Auswahl der wichtigsten Schlüsselwörter wird im Folgenden kurz vorgestellt. Genauere Informationen zur Verwendung der NED-Sprache befinden sich in der Dokumentation von OMNeT++ [Var10].

simple: *Einfache Module* (engl. *simple modules*) sind die unterste Ebene der Modulhierarchie einer OMNeT-Simulation, die das Verhalten der Simulation beinhalten. Da mit Hilfe der NED-Sprache nur die Struktur einer Simulation beschrieben wird,

wird das Verhalten der einfachen Module in separaten C++ Dateien programmiert und nicht innerhalb der Netzbeschreibung. Diese einfachen Module werden über das Schlüsselwort `simple` eingeführt und können zwei optionale Sektionen haben: Zum einen die Sektion `parameters` und zum anderen die Sektion `gates`. Die erste Sektion enthält die Eigenschaften des Moduls und innerhalb der zweiten Sektion werden die Schnittstellen des Moduls angegeben. Über diese Schnittstellen, kann das Modul Nachrichten mit anderen Modulen einer Simulation austauschen.

Code-Block 5.1 zeigt ein Beispiel, in welchem ein einfaches Modul mit dem Namen `User` (Zeile 1) definiert wird. Dieses Modul, das einen Benutzer eines Rechnernetzes modellieren soll, hat zwei Eigenschaften: Einen Namen (Zeile 5) und einen Display-String (Zeile 4), der von der grafischen Ausführungsumgebung Tkenv von OMNeT++ ausgewertet wird. Des Weiteren werden diesem Modul in Zeile 7 eine Reihe von bidirektionalen Schnittstellen (vom Typ `inout`) zugewiesen, die über den Namen `gate` angesprochen werden können.

```
1 simple User
2 {
3   parameters:
4     @display("i=block/user,green;is=s");
5     string name;
6   gates:
7     inout gate[];
8 }
```

Code-Block 5.1: NED-Beschreibung eines einfachen Moduls

network: Mittels des Schlüsselworts `network` wird ein neues Simulationsnetz eingeführt. Ein solches Netz kann mehrere Sektionen haben, die beispielsweise neue Kommunikationskanäle definieren (Sektion `types`), die von dem Simulationsnetz verwendeten Module spezifizieren (Sektion `submodules`) oder die Verbindungen zwischen den verwendeten Modulen festlegen (Sektion `connections`).

Im Code-Block 5.2 wird beispielsweise ein neues Netz mit dem Namen `Social_Malcode` eingeführt (Zeile 1), welches aus drei Modulen besteht, die vom Typ `User` sind (Zeile 9). Diese sind hier kreisförmig untereinander verbunden, festgelegt in der Sektion `connections` (Zeilen 10-13) des Simulationsnetzes. In dem Beispiel ist der Rechner des ersten Benutzers mit dem Rechner des zweiten Benutzers verbunden, dieser mit dem Rechner des dritten Benutzers und dieser wiederum mit dem Rechner des ersten Benutzers. Grafisch dargestellt ist dieses Beispielnetz in der Abbildung 5.1. Dieses Beispiel verwendet zudem einen selbst definierten Kommunikationskanal (Zeilen 3-7), der eine doppelt so große Latenz aufweist, wie die beiden anderen verwendeten Kanäle.

module: Mehrere bereits bestehende Module lassen sich über sogeannte *zusammengesetzte Module* (engl. *compound modules*) zu einem einzigen Modul zusammenfassen. Diese werden durch das Schlüsselwort `modules` definiert und können über die bereits bekannten Sektionen `types`, `parameters`, `gates`, `connections` und `submodules` verfügen. In der Sektion `submodules` werden die einzelnen Module

```
network Social_Malcode
{
  types:
    channel Channel2s extends ned.DelayChannel
    {
      delay = 2s;
    }
  submodules:
    user[3]: User;
  connections:
    user[0].gate++ <--> Channel2s <--> user[1].gate++;
    user[1].gate++ <--> {delay = 1s;} <--> user[2].gate++;
    user[2].gate++ <--> {delay = 1s;} <--> user[0].gate++;
}
```

Code-Block 5.2: NED-Beschreibung eines Simulationsnetzes

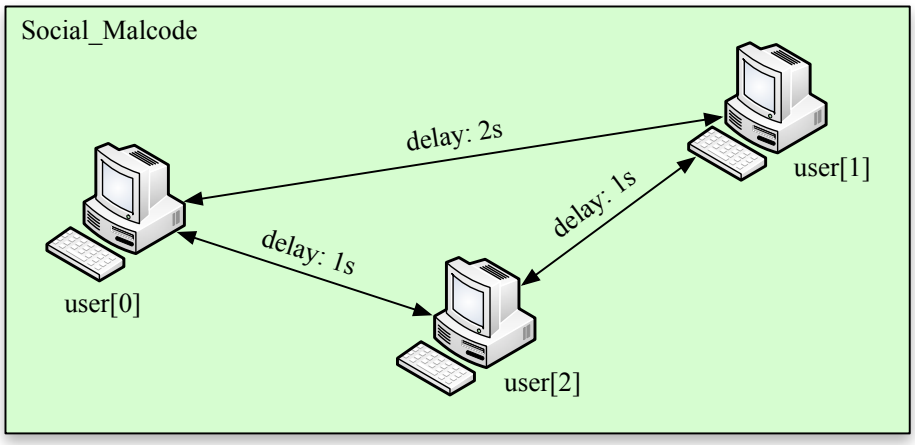


Abbildung 5.1.: Beispiel eines Simulationsnetzes mit drei Benutzerrechnern
Grafische Darstellung des Beispielnetzes, dessen Beschreibung im Code-Block 5.2 dargestellt ist. Dieses Netz besteht aus drei untereinander verbundenen Rechnern. Der Kommunikationskanal zwischen dem ersten und dem zweiten Rechner besitzt eine doppelt so große Latenz wie die beiden anderen Kanäle.

angegeben, aus denen das zusammengesetzte Modul besteht. Dabei ist es egal, ob dies einfache Module oder selbst wieder zusammengesetzte Module sind.

Code-Block 5.3 zeigt ein Beispiel eines zusammengesetzten Moduls, welches aus zwei einfachen Modulen (Zeilen 4 und 5) und einem zusammengesetzten Modul (Zeile 6) besteht. Diese Module sind untereinander durch unidirektionale Kommunikationskanäle miteinander verbunden (Zeilen 8-10).

```
1  module Compound
2  {
3    submodules:
4      simple1: SimpleModuleType;
5      simple2: SimpleModuleType;
6      subCompound: CompoundModuleType
7    connections:
8      simple1.out --> simple2.in;
9      simple2.out --> subCompound.in;
10     subCompound.in2 <-- simple1.out2;
11 }
```

Code-Block 5.3: NED-Beschreibung eines zusammengesetzten Moduls

5.1.2. Modell-Struktur und -komponenten

Eine mittels OMNeT++ erstellte Simulation besteht aus einzelnen Modulen (engl. *modules*), die hierarchisch ineinander eingebettet sein können. Die Module der untersten Hierarchieebene enthalten die eigentliche Funktionalität einer Simulation. Diese Funktionalität wird in C++ implementiert. Mittlerweile existieren auch Plug-ins, die es gestatten, die Funktionalität in Java oder C# zu formulieren.

Abbildung 5.2 zeigt den hierarchischen Aufbau einer Beispielsimulation. Das simulierte Netz besteht aus zwei einfachen Modulen und einem zusammengesetzten Modul, die über Kommunikationskanäle miteinander kommunizieren beziehungsweise Nachrichten austauschen können. In diesem Beispiel kommuniziert das einfache Modul 1 mit dem einfachen Modul 2. Dieses wiederum kann Nachrichten zu dem zusammengesetzten Modul schicken und über seine dritte Schnittstelle Nachrichten von dem zusammengesetzten Modul empfangen. Empfängt das zusammengesetzte Modul eine Nachricht, so leitet es diese intern an sein erstes Untermodul $C1$ weiter und dieses kann Nachrichten an das zweite Untermodul $C2$ verschicken. Von diesem werden Nachrichten über die zweite (in der Abbildung die untere) Schnittstelle des zusammengesetzten Moduls an das zweite einfache Modul des Simulationsnetzes versendet.

Aus welchen Komponenten beziehungsweise Modulen eine Simulation besteht und wie diese strukturell zusammenhängen und miteinander kommunizieren, wird mit Hilfe der NED-Sprache (siehe Abschnitt 5.1.1) in einer Datei mit der Dateierweiterung `.ned` spezifiziert. Innerhalb dieser Dateien wird auch hinterlegt, welche Eigenschaften die einzelnen Module der Simulation haben. Die eigentliche Aktivität der Simulation, wird durch den C++-Code der einzelnen, einfachen Module festgelegt (oder eben durch den Java- oder C#-Code, wenn ein entsprechendes Plug-in installiert ist).

Für die Simulation wird eine Konfigurationsdatei mit dem Namen `omnetpp.ini` benötigt. Innerhalb dieser Datei können mehrere Konfigurationssektionen angelegt

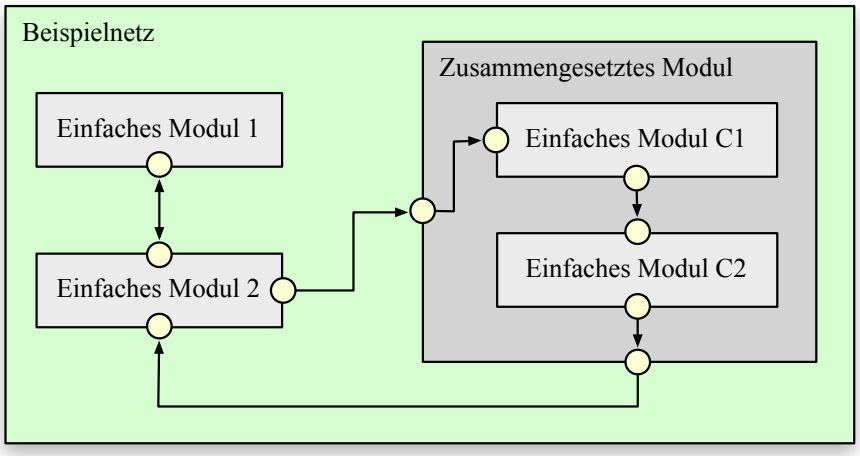


Abbildung 5.2.: Hierarchie von OMNeT++-Modulen
Dieses Beispiel-Netz besteht aus zwei einfachen Modulen und einem zusammengesetzten Modul, das wiederum aus zwei einfachen Modulen besteht. Die kleinen gelben Kreise sind die Schnittstellen der entsprechenden Module und die Pfeile zwischen diesen Schnittstellen stellen die Kommunikationskanäle zwischen den Modulen dar. [Var10]

werden, die beim Start einer Simulation einzeln aktiviert werden können. Jede einzelne dieser Sektionen kann ein unterschiedliches Verhalten der Simulation spezifizieren. So kann beispielsweise für jede Sektion eine unterschiedliche Menge an aufzuzeichnenden Daten angegeben werden oder die Eigenschaften der verwendeten Module unterschiedlich belegt beziehungsweise initiiert werden. Die Eigenschaft `name` des einfachen Moduls `User` aus dem Code-Block 5.1 kann zum Beispiel in einer Konfigurationssektion anders belegt werden als in einer davon unterschiedlichen weiteren Sektion.

Wie bereits angesprochen, ermöglicht es OMNeT++ während einer Simulation zur statistischen Auswertung verschiedene Daten und Informationen aufzuzeichnen. Diese Informationen werden nach jedem Simulationslauf, abhängig von der Art der aufgezeichneten Daten, entweder in Skalar- oder Vektor-Dateien abgelegt. Die integrierte Entwicklungsumgebung von OMNeT++ bietet dem Benutzer ein paar schlichte Werkzeuge, mit denen diese Daten analysiert und betrachtet werden können. Abbildung 5.3 zeigt beispielhaft die Ansicht von aufgezeichneten Vektordaten mit Hilfe eines durch die Entwicklungsumgebung mitgelieferten Betrachtungswerkzeuges.

5.1.3. Komponenten des Simulationsrahmenwerks

Das OMNeT++ Rahmenwerk besteht aus mehreren Komponenten, die zum einen unerlässlich zur Erstellung einer Simulation sind und zum anderen den Benutzer lediglich bei der Erstellung einer Simulation unterstützen sollen.

Die eigentliche Kernfunktionalität von OMNeT++, beispielsweise der Nachrichten-Austausch zwischen Modulen, das Erzeugen der Simulation oder auch die zeitliche Planung und Durchführung der Ablaufbehandlung einer Simulation, ist in der so-

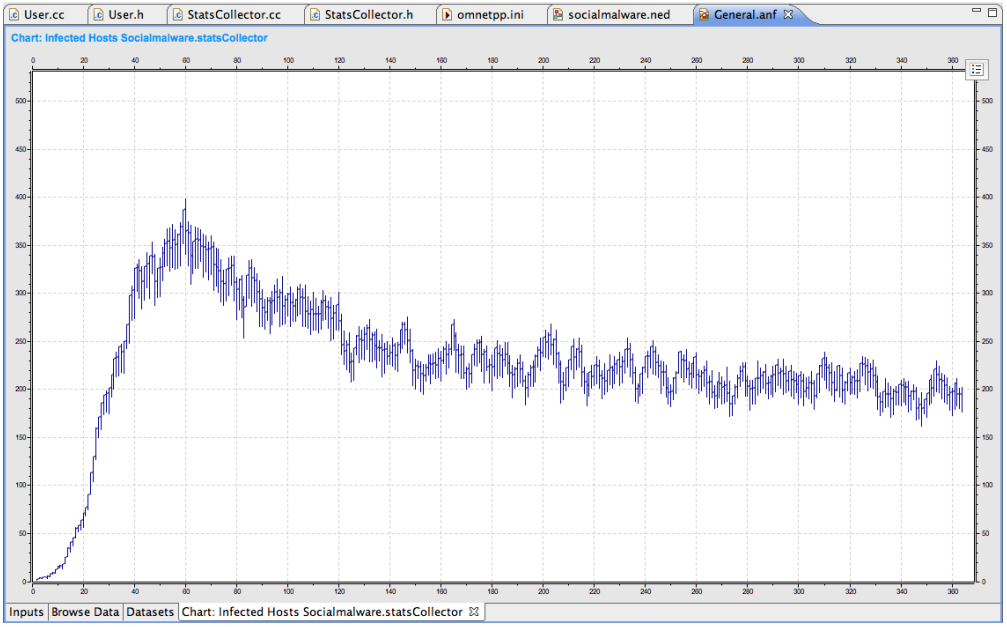


Abbildung 5.3.: Aufgezeichnete Vektordaten eines Simulationslaufes
OMNeT++ bietet dem Benutzer eine Vielzahl von Werkzeugen zum Betrachten und Auswerten der Simulationergebnisse. Das Diagramm ist eine grafische Darstellung von aufgezeichneten Vektordaten durch eines dieser Werkzeuge.

genannten *Kernel-Bibliothek* gekapselt. Dies ist eine C++-Klassenbibliothek und sie wird von den benutzerdefinierten Klassen (die das Verhalten der einfachen Module implementieren) eingebunden. Dadurch müssen beim Erstellen einer Simulation keine Änderungen am Quellcode von OMNeT++ durchgeführt werden und es besteht auch nicht die Notwendigkeit, Dateien in ein bestimmtes Verzeichnis zu kopieren, damit die Simulation lauffähig wird – wie dies bei anderen Simulations-Rahmenwerken teilweise der Fall ist. [Ras07]

Abbildung 5.4 zeigt ein Bild der auf *Eclipse* basierenden IDE. Auf der linken Seite der IDE bekommt der Benutzer eine Übersicht der aktuellen Projekte und der zu diesen Projekten gehörenden Dateien präsentiert. Im mittleren Bereich der IDE lassen sich diese Dateien bearbeiten. Neben einem einfachen Textmodus mit Syntax-Highlighting, beinhaltet die IDE auch einen grafischen Modus zum Erstellen und Bearbeiten der Netztopologie (siehe Abschnitt 5.1.1) und eine einfache, grafische Auswertung der Simulationsergebnisse. Ein Bildschirmausschnitt, der eine grafische Darstellung von aufgezeichneten Simulationsdaten zeigt, ist in Abbildung 5.3 enthalten. Zudem bietet die IDE dem Benutzer eine Übersicht über die von einer Simulation verwendeten Klassen und Funktionen (rechter Bereich der IDE) und mehrere kleine Fenster, zum Anpassen und zur Fehlerbehandlung einer Simulation (unterer Bereich der IDE).

Damit eine Simulation die vom Benutzer spezifizierte Netztopologie verwenden kann, muss diese mit Hilfe der NED-Sprache verfasste Beschreibung der Topologie in eine von OMNeT++ verständliche Form übersetzt werden – diese Aufgabe übernimmt ein mit OMNeT++ ausgelieferter *NED-Compiler*. Zudem beinhaltet die Windows-

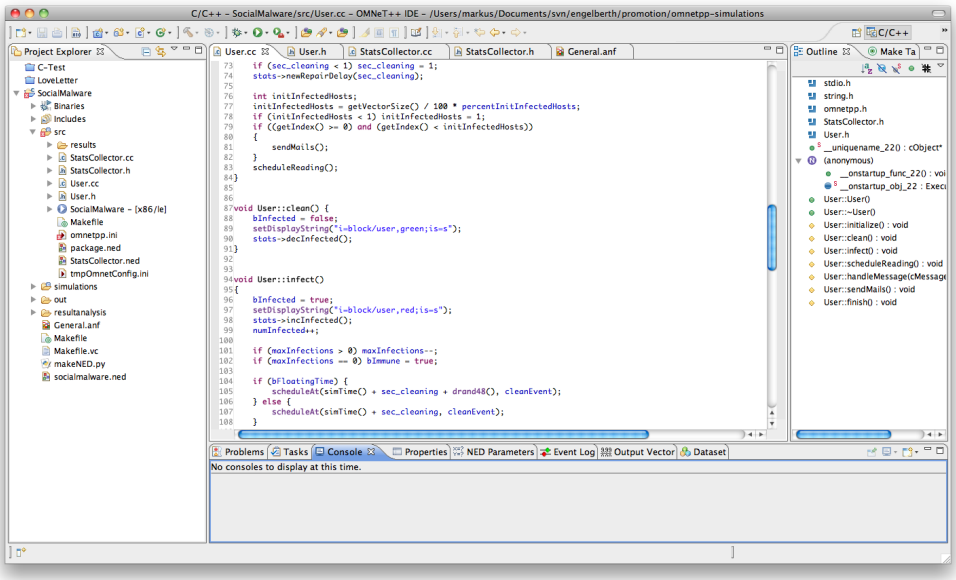


Abbildung 5.4.: Integrierte Entwicklungsumgebung von OMNeT++
Die mit OMNeT++ ausgelieferte, integrierte Entwicklungsumgebung basiert auf Eclipse. Auf der linken Seite wird dem Benutzer eine Übersicht über die verwendeten Dateien einer Simulation präsentiert, in dem mittleren Bereich lassen sich diese Dateien editieren und rechts befindet sich eine Übersicht der erstellten Klassen und Funktionen.

Version von OMNeT++ den *MinGW C++-Compiler*, um das Verhalten der einfachen Module zu übersetzen, welches in C++ hinterlegt wird.

Zur Ausführung der erstellten Simulationen werden mit OMNeT++ zwei Umgebungen ausgeliefert: Zum einen *Tkenv* und zum anderen *Cmdenv*. Dabei bietet die erste Ausführungsumgebung dem Benutzer eine grafische Oberfläche und die zweite ist eine rein textbasierte Ausführungsumgebung für die Kommandozeile. Über die grafische Oberfläche von Tkenv kann eine Simulation zur Ausführungszeit in einem gewissen Umfang beeinflusst und bereits erstmalig ausgewertet werden. So können etwa die Zustände der verwendeten Module eingesehen werden oder es kann die Ausführungsgeschwindigkeit der Simulation geändert werden. Cmdenv bietet dagegen keine Eingriffsmöglichkeiten in den Ablauf der Simulation, ist jedoch wesentlich performanter und kann dazu benutzt werden, die Simulationsausführung zu automatisieren.

5.2. Simulation des Verbreitungsverlaufs eines E-Mail-Wurms

Um den Verbreitungsverlauf von Social Malcode zu modellieren, wurde mit Hilfe von OMNeT++ eine Simulation erstellt, die eine Verbreitung eines E-Mail-Wurms simuliert. Anhand verschiedener Simulationsparameter werden die Einflüsse unterschiedlicher, äußerer Gegebenheiten auf die Verbreitung des E-Mail-Wurms nachgestellt und

analysiert – beispielsweise der Zeitdauer zwischen dem Eintreffen und dem Lesen einer E-Mail durch deren Empfänger.

Die folgenden Abschnitte beschreiben den simulierten E-Mail-Wurm, den Aufbau beziehungsweise die Struktur des simulierten Netzes, das Verhalten der einzelnen an der Simulation beteiligten Komponenten und die Parameter, welche die Simulation beeinflussen. Der letzte Abschnitt enthält abschließend die Ergebnisse der Simulation. Diese werden anschaulich anhand von dreidimensionalen Verbreitungsverläufen dargestellt.

5.2.1. E-Mail-Wurm

Tabelle 4.3 zeigt, dass in ca. 36,1 % aller Spam-Mails des untersuchten Spam-Korpus mindestens ein Verweis enthalten ist. Die Anzahl der Spam-Mails mit mindestens einem Verweis ist über 11 Mal größer als die Anzahl der Spam-Mails, die mindestens einen Dateianhang enthalten.

Diese hohe Signifikanz der Verweise gegenüber einem Dateianhang (bezogen auf die Verbreitung eines E-Mail-Wurms) liegt wahrscheinlich daran, dass Dateianhänge selbst ein Teil der E-Mail sind – also in diese eingebettet den Empfänger erreichen. Somit passieren auch die Dateianhänge an sich jegliche E-Mail Gateways und Filtersysteme, bevor die E-Mail den Posteingang der Empfänger erreicht. Dadurch ist die Erkennungsgefahr wesentlich größer als bei der Verbreitung über einen Verweis. Im Falle eines Verweises befindet sich der Schadcode auf einem entfernten Server und wird in der Regel nicht durch vorhandene E-Mail-Filtersysteme geprüft.

Dementsprechend haben wir uns dazu entschlossen, bei der Simulation von Social Malcode ein Schadprogramm zu simulieren, dessen Verbreitung ebenso durch E-Mails mit einem Verweis erfolgt. Der simulierte E-Mail-Wurm verbreitet sich durch folgende Schritte:

1. Eine E-Mail mit einem Verweis auf eine Internetseite trifft im Posteingang eines Empfängers ein.
2. Der Empfänger liest eventuell die E-Mail, d. h. er öffnet die E-Mail in einem E-Mail-Client.
3. Der Empfänger folgt eventuell dem enthaltenen Verweis, indem er mit dem Mauszeiger auf den Verweis klickt.
4. Ein Webbrowser zeigt die Internetseite an, auf die der Verweis innerhalb der E-Mail zeigt.
5. Die Internetseite ist so gestaltet, dass bei deren Besuch mit einem anfälligen Webbrowser ein Drive-By-Download ausgelöst wird. Dabei wird eine neue Instanz des simulierten E-Mail-Wurms auf den Rechner geladen und ausgeführt. Sollte der Rechner bereits mit einer Instanz des E-Mail-Wurms infiziert sein, so findet keine neuerliche Infektion statt.
6. Nach einer Infektion liest der E-Mail-Wurm die E-Mail-Adressen aus dem Adressbuch des befallenen Rechners aus. Des Weiteren werden die E-Mails des Posteingangs analysiert und die darin gefundenen Absenderadressen ebenfalls zur Verbreitung des Wurms genutzt.

7. Der E-Mail-Wurm verschickt die gleiche E-Mail, deren Eintreffen in Schritt 1 das auslösende Ereignis für die Verbreitung des Wurms war, an die gefundenen E-Mail-Adressen des vorherigen Schrittes.

Nach dem letzten Schritt setzt sich die Verbreitung des E-Mail-Wurms auf den Rechnern der neuen Empfänger fort. Die in Definition 3.8 geforderte kausale Folgerung von drei Zustandsübergängen u_{i_1}, u_{i_2} und u_{i_3} ist bei dem hier simulierten E-Mail-Wurm folgendermaßen gegeben:

u_{i_1}: Eine Herausforderung oder Provokation einer Benutzerinteraktion durch den Versand der E-Mail (Schritt 1 bzw. Schritt 7)

u_{i_2}: Zwei Benutzerinteraktionen durch das Öffnen der E-Mail in einem E-Mail-Client (Schritt 2) und dem Folgen des Verweises durch einen Mausklick (Schritt 3)

u_{i_3}: Daraufhin kommt es zu einer Code-Ausführung, welche die Infektion des zugrundeliegenden Systems verursacht (Schritt 5)

Somit entspricht der simulierte Wurm der Definition von Social Malcode aus Kapitel 3. Der beschriebene Wurm ist keine Simulation eines reales Schadprogramms, sondern simuliert lediglich einen sehr häufigen Verbreitungsweg von Social Malcode. Darüber hinaus wird in der hier durchgeführten Simulation komplett von Schadroutinen abstrahiert, da diese in der Regel für die Verbreitung eines Schadprogramms bedeutungslos sind. Selbstverständlich könnte man die Simulation auch mit einem realen oder wesentlich komplexeren Wurm durchführen, jedoch reicht die hier gewählte Variante vollkommen aus, um die Einflüsse von äußeren Gegebenheiten auf die Verbreitung des E-Mail-Wurms zu untersuchen.

5.2.2. Modell-Struktur

Da die gewählte Netztopologie starken Einfluss auf den Verbreitungsverlauf von Social Malcode nimmt, beschreiben wir in diesem Abschnitt die bei der Simulation verwendete Netztopologie – diese bildet die Modell-Struktur der Simulation.

Das simulierte Netz besteht aus N Rechnern, deren Benutzer sich gegenseitig E-Mails zuschicken können (im Folgenden als *E-Mail-Netz* bezeichnet) und entspricht einem ungerichteten Graphen

$$G = (V, E), \text{ mit } |V| = N,$$

wobei jeder Knoten $v_i \in V$ für $i \in \{1, \ldots, N\}$ einen Rechner mit einem E-Mail-Client darstellt. In jedem dieser E-Mail-Clients ist genau ein E-Mail-Konto eingerichtet, das genau eine E-Mail-Adresse a_i abruft, so dass folgende Ungleichung gilt:

$$i \neq j \Rightarrow a_i \neq a_j, \ \forall\, i, j \in \{1, \ldots, N\}$$

Zudem ist in jedem E-Mail-Client eines Knotens v_i ein Adressbuch A_i vorhanden, das aus einer Menge von zu a_i unterschiedlichen E-Mail-Adressen besteht. Jedem Rechner sind zwei Zustände zugeordnet: Entweder ist der Rechner *infiziert*, d. h. er ist durch Social Malcode befallen, oder er hat den Zustand *sauber*. Der letztere Zustand bedeutet, dass der Rechner nicht durch Social Malcode infiziert ist.

Sind zwei Knoten v_1 und v_2 durch eine Kante $e = \{v_1, v_2\} \in E$ miteinander verbunden, so bedeutet dies, dass sich die zu dem Knoten v_1 gehörige E-Mail-Adresse a_1 im Adressbuch A_2 des Knotens v_2 befindet und umgekehrt. Voraussetzung dieses Modells ist also, dass zwei Personen bidirektional miteinander kommunizieren und dies nicht einseitig geschieht. Allgemein gilt somit also:

$$i \neq j \Rightarrow \{v_i, v_j\} \in E \Longleftrightarrow a_i \in A_j \wedge a_j \in A_i, \ \ \forall\, i, j \in \{1, \ldots, N\}$$

Da durch die Simulation die Verbreitung des E-Mail-Wurms untersucht werden soll, ist diese Voraussetzung bereits dadurch erfüllt, dass der Wurm bei der Generierung neuer E-Mail-Adressen auch den Posteingang analysiert. In diesem befindet sich natürlich auch die E-Mail, die zur Infektion des Rechners geführt hat, und somit wird auch an den ursprünglichen Absender wieder eine neue Spam-Mail geschickt.

Da die in dem Adressbuch A_i eines Knotens v_i vorhandenen E-Mail-Adressen verschieden von a_i sind, gibt es im gesamten E-Mail-Netz keine Schleife. Es gilt also:

$$\{v_i, v_i\} \notin E, \ \ \forall\, i \in \{1, \ldots, N\}$$

5.2.2.1. Verzweigungsgrad

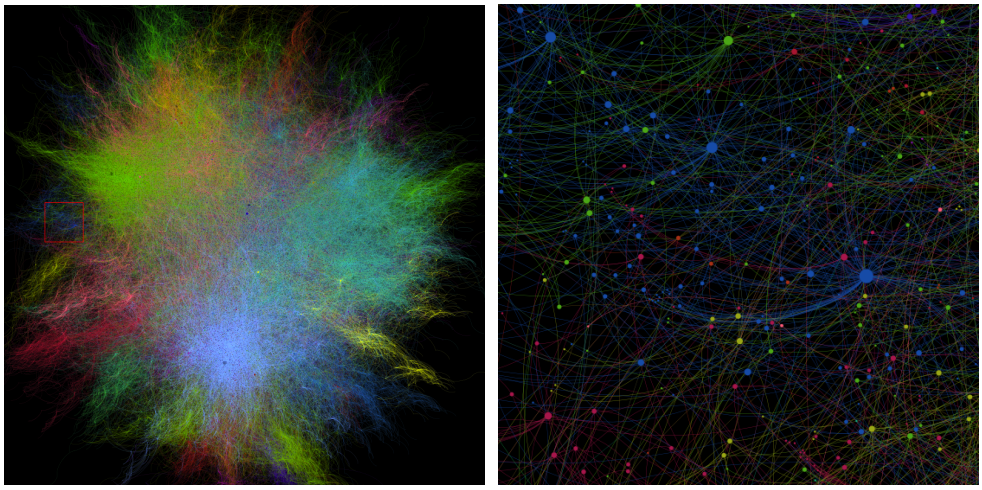
Der Verzweigungsgrad der Knoten des simulierten E-Mail-Netzes wirkt sich genauso auf die Verbreitung von Social Malcode aus. Der Verzweigungsgrad eines Knotens entspricht, übertragen auf die in diesem Kapitel beschriebene Simulation, der Anzahl der Kontakte in einem Adressbuch, d. h. der Größe des Adressbuches.

Man kann sich leicht vorstellen, dass sich Schadprogramme in einem Netz, das beispielsweise durch einen *vollständigen Graphen*[1] repräsentiert wird, wesentlich schneller ausbreiten als in einem Netz, das die Topologie eines *linearen Graphen*[2] aufweist. Dies liegt daran, dass die durchschnittliche, kürzeste Pfadlänge in einem Graphen mit hohem Verzweigungsgrad kürzer ist als in einem Graphen mit niedrigerem Verzweigungsgrad. Im Bezug auf die Simulation bedeutet dies: Benutzer mit relativ vielen Kontakten in ihrem Adressbuch tragen wesentlich stärker zur Verbreitung des E-Mail-Wurmes bei, als Benutzer, die kaum Kontakte in ihrem Adressbuch haben.

Albert-László Barabási und Réka Albert haben gezeigt, dass der Verzweigungsgrad großer Netze einer Verteilung nach einem Potenzgesetz (engl. *power law*) folgt [BA99, AB02]. Jure Leskovec und Eric Horvitz haben zudem im Jahr 2008 das Kommunikationsverhalten von Benutzern des *Microsoft Instant-Messaging*-Netzes untersucht und dabei unter anderem festgestellt, dass die Größen der Freundeslisten in diesem Netz einer Verteilung entspricht, die einem Potenzgesetz mit einem exponentiellen Abfall unterliegt [LH08]. Da die Verteilung der beiden Größen – Anzahl der Adressbuchkontakte in einem E-Mail-Netz und Anzahl der Freunde in einem Instant-Messaging-Netz – als hinreichend ähnlich anzusehen sind, wird angenommen, dass die Verteilung der Adressbuchgrößen ebenfalls einer Verteilung nach einem Potenzgesetz folgt.

[1] In einem vollständigen Graphen sind alle Knoten jeweils durch eine unterschiedliche Kante miteinander verbunden.

[2] Ein linearer Graph ist zusammenhängend und alle Knoten haben den Verzweigungsgrad 2 bis auf zwei Knoten, die den Verzweigungsgrad 1 haben.



(a) Gesamtübersicht **(b)** Ausschnitt

Abbildung 5.5.: Grafische Darstellung des Simulationsnetzes
Teil (a) der Abbildung enthält eine mit Gephi erstellte Darstellung des gesamten E-Mail-Netzes. In Teil (b) der Abbildung ist ein vergrößerter Ausschnitt dargestellt. Dieser Ausschnitt ist in der Gesamtübersicht durch ein rotes Quadrat am linken Rand kenntlich gemacht.

Zur Simulation der Verbreitung des E-Mail-Wurms haben wir ein E-Mail-Netz generiert, das aus insgesamt $N = 50\,000$ Rechnern besteht. In Abbildung 5.5 ist eine mit *Gephi* [BHJ09] erstellte Visualisierung des E-Mail-Netzes zu sehen. Teil (a) der Abbildung zeigt eine Gesamtübersicht des Netzes und Teil (b) einen vergrößerten Ausschnitt. Der Bereich des Ausschnitts ist in der Gesamtübersicht durch ein rotes Quadrat am linken Rand der Abbildung kenntlich gemacht. Je höher der Verzweigungsgrad eines Knotens, desto größer ist dieser in beiden Abbildungsteilen dargestellt.

Die Verteilung der Adressbuchgrößen ist in Abbildung 5.6 dargestellt. Da diese Verteilung einem Potenzgesetz folgt, hat sie in der vorliegenden, doppelt-logarithmischen Darstellung eine lineare Grundausrichtung. Die eingezeichnete, rote Gerade entspricht einem Potenzgesetz mit dem Parameter $\alpha = 2{,}78$. Die durchschnittliche Adressbuchgröße liegt bei 5,194 Kontakten, das größte Adressbuch enthält 1 876 Kontakte und der Median der Verteilung ist eine Adressbuchgröße von 3 Kontakten.

Zur Generierung des Netzes haben wir einen Algorithmus von Holme und Kim [HK02] verwendet, der das *Modell von Barabási und Albert* (dies ist ein Algorithmus zur Generierung von skaleninvarianten Zufallsgraphen [AB02]) um einen zusätzlichen Schritt erweitert. Der verwendete Algorithmus besteht aus folgenden Schritten, die iterativ wiederholt werden, bis die gewünschte Knotenanzahl erreicht ist:

1) Ein neuer Knoten v_{neu} wird dem bereits bestehenden Graphen hinzugefügt.

2a) Der Knoten v_{neu} wird mit einer bestimmten Anzahl bereits bestehender Knoten durch eine Kante verbunden. Je höher der Verzweigungsgrad eines bereits vorhandenen Knotens v_{alt} ist, desto höher ist die Wahrscheinlichkeit, dass der neue Knoten v_{neu} mit dem Knoten v_{alt} verbunden wird.

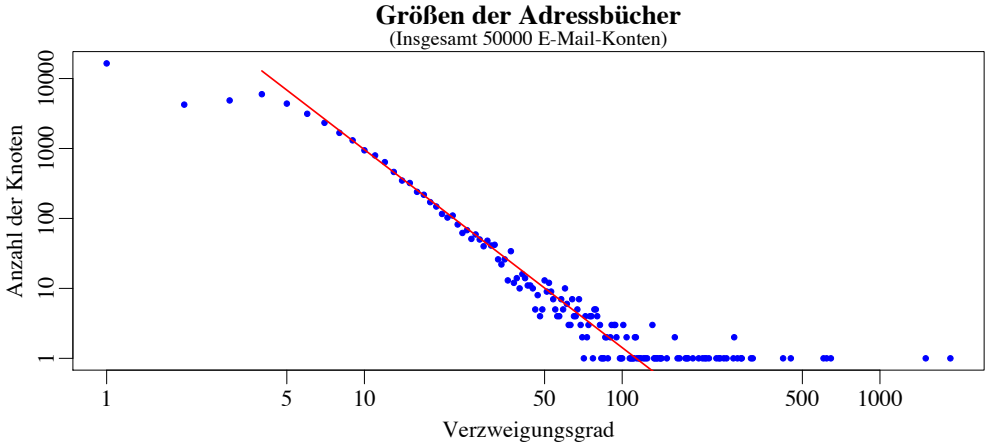


Abbildung 5.6.: Verteilung der Adressbuchgrößen
Die Adressbuchgrößen sind nach einem Potenzgesetz verteilt, d. h. wenige Adressbücher enthalten sehr viele Kontakte und der Großteil der Adressbücher enthält wenige bis durchschnittlich viele Kontakte. Die Achsen der Abbildung sind logarithmisch skaliert, so dass die Adressbuchgrößen eine lineare Grundausrichtung haben.

2b) Jedes Mal, wenn in dem vorherigen Schritt eine neue Kante hinzugefügt wurde, wird mit einer gewissen Wahrscheinlichkeit eine weitere Kante hinzugefügt, die den neu eingefügten Knoten v_{neu} mit einem Nachbarknoten $v_{alt\_neighbour}$ des Knotens v_{alt} verbindet. Auf diese Weise entsteht ein Dreieck zwischen den Knoten v_{neu}, $v_{alt\_neighbour}$ und v_{alt}.

Durch Schritt 2a) erhält der Verzweigungsgrad des entstehenden Graphen die gewünschte Form. Dies liegt daran, dass neu eingefügte Knoten bevorzugt mit einem Knoten mit hohem Verzweigungsgrad verbunden werden, so dass die Verteilung des Verzweigungsgrades des gesamten Netzes einem Potenzgesetz folgt.

Schritt 2b) ist dafür verantwortlich, dass in dem E-Mail-Netz Cluster entstehen. Vereinfacht ausgedrückt ist ein Cluster eine Teilmenge von Knoten, die untereinander stark miteinander verbunden sind, jedoch kaum Verbindungen zu Knoten anderer Cluster aufweisen. In einem realen Kommunikationsnetz entsprechen die Cluster Gruppen von stark miteinander vernetzten Benutzern. Beispiele für solch stark vernetzte Gruppen können etwa die Abonnenten einer Mailingliste, die Mitglieder eines Vereins oder etwa die Mitarbeiter eines Unternehmens sein, die sich untereinander vermehrt E-Mails zuschicken. Um die verschiedenen Cluster in Abbildung 5.5 besser kenntlich zu machen, sind diese farblich voneinander abgegrenzt, d. h. Knoten mit derselben Farbe befinden sich in demselben Cluster. Für die Abbildung wurde der Graph mittels einer Heuristik, die auf einer Optimierung der Modularität (engl. *modularity*) beruht, in verschiedene Cluster eingeteilt [BGLL, New06].

5.2.3. Modell-Verhalten

Das Verhalten der Simulation wird im Wesentlichen durch das Verhalten der (simulierten) Benutzer des E-Mail-Netzes bestimmt. Neben Modulen, die einen Benutzer simu-

lieren, wurden noch Funktionen und Module implementiert, die lediglich dem Sammeln und Auswerten von Simulationsdaten dienen. Da diese Module keinen Einfluss auf die Verbreitung des simulierten E-Mail-Wurms haben, wird in diesem Abschnitt nur das simulierte Benutzerverhalten vorgestellt.

Ein Programmablaufplan, der den globalen Ablauf der Simulation darstellt, ist in Abbildung 5.7 zu sehen. Nach dem Start der Simulation schicken sich die Benutzer der einzelnen Rechner so lange E-Mails zu, bis die Simulation beendet ist. Die verschickten E-Mails bewirken, dass sich ein Rechner mit dem simulierten Wurm infizieren kann. Die gesamte Anzahl der infizierten Rechner wird für jede simulierte Stunde festgehalten und entspricht somit dem Verbreitungsverlauf des E-Mail-Wurms. Durch die Säuberung eines infizierten Rechners, also dem Entfernen des Wurms, kann diese Zahl auch sinken.

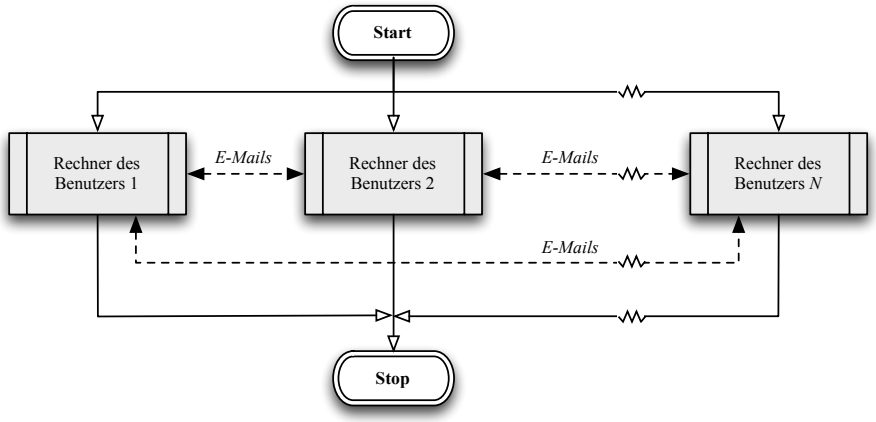


Abbildung 5.7.: Globaler Simulationsablauf
Nach dem Start der Simulation schicken sich die einzelnen Benutzer des Simulationsnetzes so lange E-Mails zu, bis die Simulation beendet ist.

Die Abbruchbedingungen der Simulation werden jeweils in den einzelnen Unterprogrammen des Programmablaufplans abgefragt. Beendet ist die Simulation, falls eins der folgenden drei Ereignisse eintritt:

1. Die Simulation hat das zeitliche Limit von 1 200 simulierten Stunden erreicht. Dies entspricht genau 50 Tagen.
2. Zu einem beliebigen Zeitpunkt der Simulation haben alle Rechner den Zustand *sauber* und in keinem Posteingang der Rechner befindet sich eine neue, ungelesene E-Mail. In diesem Fall ist der E-Mail-Wurm sozusagen *ausgestorben*.
3. Zu jedem Zeitpunkt der Simulation wird in einer globalen Statistik die Anzahl der aktuell infizierten Rechner festgehalten. Sobald ein Rechner seinen Zustand ändert (also entweder von *sauber* nach *infiziert* oder umgekehrt) wird dieser Wert angepasst und der alte Wert wird in einem FIFO-Puffer mit 150 Plätzen abgelegt. Sind alle Plätze dieses Puffers belegt, wird nach dem Einfügen eines neuen Wertes überprüft, ob die Differenz zwischen dem maximalen und dem minimalen Wert dieses Puffers geringer als 5 (Infektionen) ist. Ist dies der Fall,

wird der Verlauf der Anzahl der infizierten Rechner als konstant angesehen und die Simulation ist ebenfalls beendet.

Wie sich ein Rechner nach dem Eintreffen einer E-Mail mit dem simulierten Wurm infizieren kann und wann genau die Abbruchbedingungen überprüft werden, ist in dem Programmablaufplan der Abbildung 5.8 veranschaulicht. Dieser Plan enthält das Verhalten eines einzelnen, lokalen Benutzers und entspricht den Unterprogrammen der Abbildung 5.7, die mit *Rechner des Benutzers X* beschriftet sind.

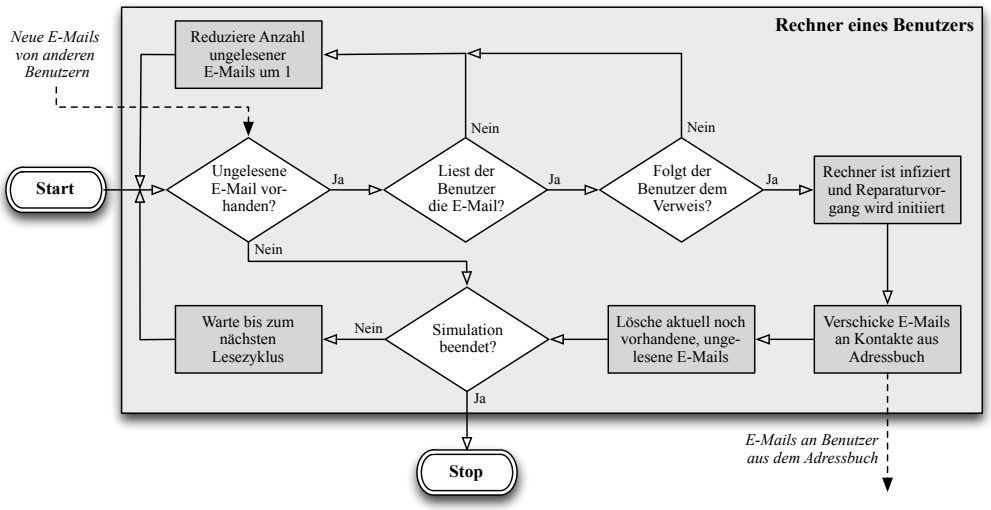


Abbildung 5.8.: Simuliertes (lokales) Verhalten eines einzelnen Benutzers
Ein simulierter Benutzer überprüft in bestimmten Intervallen den Posteingang seines Rechners. Sind neue, ungelesene E-Mails vorhanden, kann er diese lesen. Folgt er darüber hinaus dem darin enthaltenen Verweis, resultieren daraus eine Infektion des Rechners und damit auch der Versand neuer E-Mails.

Die Semantik der neun Elemente des Programmablaufplans aus Abbildung 5.8 wird im Folgenden erläutert. In die Semantik fließen unmittelbar der Ergebnisse des Kapitels 4 ein – vor allem die beobachteten Verteilungen und deren Parameter.

1. ***Ungelesene E-Mails vorhanden?***
   Der Benutzer öffnet seinen E-Mail-Client und überprüft, ob sich im Posteingang neue, ungelesene E-Mails befinden.
2. ***Liest der Benutzer die E-Mail?***
   Falls eine ungelesene E-Mail im Posteingang des Benutzers vorhanden ist, liest er mit einer Wahrscheinlichkeit p_{read} diese E-Mail.
3. ***Folgt der Benutzer dem Verweis?***
   Falls der Benutzer eine E-Mail liest, folgt er mit einer Wahrscheinlichkeit p_{click} dem Verweis innerhalb der E-Mail.
4. ***Reduziere Anzahl ungelesener E-Mails um 1***
   Während der Simulation wird für jeden simulierten Rechner die Anzahl der ungelesenen E-Mails gespeichert. Wenn der Benutzer eine ungelesene E-Mail nicht liest oder nicht dem Verweis der E-Mail folgt, trägt diese E-Mail nicht zur Ver-

breitung des Wurms bei. Entsprechend wird die Anzahl der ungelesenen E-Mails um eine E-Mail verringert.

5. ***Rechner ist infiziert und Reparaturvorgang wird initiiert***
   Falls der Benutzer dem Verweis folgt, gilt der Rechner als infiziert. Bei der Simulation wird davon ausgegangen, dass die Benutzer Webbrowser benutzen, die anfällig für einen Drive-By-Download sind – beispielsweise weil noch kein Patch für die entsprechende Sicherheitslücke existiert. Schadcode, der noch unbekannte Sicherheitslücken oder Sicherheitslücken ausnutzt, für die noch kein Patch existiert, wird als *Zero-Day-Exploit* bezeichnet. Da von solchen Exploits eine wachsende Gefahr ausgeht, ist dies ein sehr realistisches Szenario [Poh09, Fox09]. Infolge der Infektion wird ebenfalls ein Reparaturvorgang des Rechners initiiert, der mit einem gewissen zeitlichen Abstand nach der Infektion durchgeführt wird. Die Anzahl der Stunden zwischen der Infektion und dem Beginn der Säuberung ist mit einer Rate von $\lambda_{cleaning}$ exponentialverteilt.
6. ***Verschicke E-Mails an Kontakte aus Adressbuch***
   Nach der Infektion eines Rechners, durchsucht der E-Mail-Wurm das Adressbuch und den Posteingang des Rechners und verschickt E-Mails an alle gefunden E-Mail-Adressen. Die E-Mails enthalten wiederum den Verweis auf die bösartig präparierte Internetseite.
7. ***Lösche aktuell noch vorhandene, ungelesene E-Mails***
   Die Simulation ist so implementiert, dass davon ausgegangen wird, dass ein Benutzer, der sich mit dem E-Mail-Wurm infiziert hat, die Infektion irgendwie bemerkt und alle weiteren E-Mails mit dem Verweis auf die präparierte Internetseite ignoriert. Aus diesem Grund wird nach einer Infektion des Rechners die Anzahl der ungelesenen E-Mails auf den Wert 0 zurückgesetzt.
8. ***Simulation beendet?***
   Hier werden die auf Seite 138 beschriebenen Abbruchbedingungen überprüft. Sollte eine dieser globalen Bedingungen erfüllt sein, führt der Benutzer keine Aktionen mehr durch und wartet auf die Beendigung der Simulation. Sobald alle Benutzer diesen Wartezustand erreicht haben, ist die komplette Simulation beendet.
9. ***Warte bis zum nächsten Lesezyklus***
   Sollte die Simulation nicht beendet sein, wird in diesem Element auf den nächsten Lesezyklus gewartet. Die Verteilung der Stunden zwischen den einzelnen Lesezyklen eines Benutzers folgt einem Potenzgesetz mit den Parametern $\alpha_{reading}$ und $C_{reading}$.

Die Ereignisse der beiden Wahrscheinlichkeiten p_{read} und p_{click} könnte man zu einem einzelnen Ereignis „*der Benutzer liest die E-Mail und folgt dem darin enthaltenen Verweis*“ zusammenfassen, da sie in dem Programmablaufplan unmittelbar aufeinander folgen. Die Gesamtwahrscheinlichkeit dieses Ereignisses würde dem Produkt der beiden Einzelwahrscheinlichkeiten entsprechen, da die Entscheidung, ob ein Benutzer beim Lesen der E-Mail dem Verweis folgt, nicht davon beeinflusst wird, mit welcher Wahrscheinlichkeit er überhaupt die E-Mail liest. Da diese beiden Ereignisse aber zwei eigenständige Benutzerinteraktionen darstellen, haben wir uns dazu entschlossen, diese nicht zu einem Gesamtereignis zusammenzufassen.

5.2.4. Belegung der Parameter

In den beiden vorhergehenden Abschnitten wurden einige Parameter vorgestellt, über die sich die Simulation steuern lässt. Neben Parametern, die den Aufbau und die Struktur des simulierten E-Mail-Netzes betreffen, waren dies vor allem Parameter bezüglich des Benutzerverhaltens. In diesem Abschnitt werden vier weitere Parameter eingeführt, die ebenfalls die Verbreitung des E-Mail-Wurms beeinflussen können.

Neben der Netztopologie und dem Verzweigungsgrad ist für die Verbreitung des Wurms auch entscheidend, wie viele Benutzer zu Beginn der Simulation eine Spam-Mail in ihrem Posteingang haben. Erhält zum Beispiel kein Benutzer eine initiale Spam-Mail, so kann sich der Wurm selbstverständlich auch nicht verbreiten. Die Anzahl der Benutzer, die zu Beginn der Simulation eine Spam-Mail erhalten, wird durch den Parameter N_{init} gesteuert. Bezogen auf die Netztopologie nennen wir diese Knoten im weiteren Verlauf der Arbeit Startknoten.

Durch die Simulation soll auch der Einfluss einer Verzögerung in der Zustellgeschwindigkeit der E-Mails untersucht werden. Diese Verzögerung wird im Folgenden durch den Parameter ch_{delay} (für *channel delay*) konfiguriert und gibt die Anzahl der Sekunden an, die zwischen dem Versand einer E-Mail und dem Eintreffen im Posteingang des Empfängers vergehen.

Die letzten beiden noch nicht beschriebenen Parameter untersuchen den Einfluss einer möglichen Immunität der Rechner gegen den simulierten E-Mail-Wurm. Der erste Parameter p_{immune} gibt die Wahrscheinlichkeit an, dass ein Rechner nach einer Infektion mit dem E-Mail-Wurm immun gegen diesen wird. Der zweite Parameter d_{immune} (für *degree immune*) bestimmt anhand der Adressbuchgröße eines Rechners, ob dieser generell immun gegen den Wurm ist – also auch schon zu Beginn der Simulation. Hat ein Benutzer mehr als d_{immune} Kontakte in dem Adressbuch seines E-Mail-Clients, so ist der Rechner dieses Benutzers immun gegen den E-Mail-Wurm. Somit kann man den Effekt einer gezielten Immunisierung von zentralen Knoten des Netzes untersuchen.

Bevor im nächsten Abschnitt der Einfluss der einzelnen Parameter auf die Verbreitung des Wurms untersucht wird, zeigt Tabelle 5.1 einen zusammenfassenden Überblick über alle Simulationsparameter. Neben einer kurzen Beschreibung der Parameter wird in der Tabelle auch jeweils deren Standardbelegung festgelegt. Jedes Mal, wenn in dem folgenden Abschnitt nicht explizit eine Belegung eines Parameters angegeben ist, wird der entsprechende Standardwert der Tabelle benutzt.

Tabelle 5.1.: Überblick über die Simulationsparameter
Neben den Namen und einer kurzen Beschreibung der Parameter ist in der letzten Spalte der Tabelle deren Standardbelegung angegeben.

Parameter	Kurzbeschreibung	Standardwert
p_{read}	Die Wahrscheinlichkeit, dass ein Benutzer eine neue, noch ungelesene E-Mail liest	0,608

Fortsetzung auf nächster Seite →

→ *Fortsetzung der vorherigen Seite*

Parameter	Kurzbeschreibung	Standardwert
p_{click}	Die Wahrscheinlichkeit, dass ein Benutzer dem Verweis einer E-Mail folgt	0,25
$\lambda_{cleaning}$	Die Rate, welche die exponentialverteilte Zeit (in Stunden) zwischen der Infektion und der Säuberung eines Rechners beschreibt	0,01486432
$\alpha_{reading}$	Der Exponent der Verteilung, die einem Potenzgesetz unterliegt und die Zeit (in Stunden) zwischen dem Eintreffen und dem Lesen einer E-Mail beschreibt	1,2821
$C_{reading}$	Der Vorfaktor der Verteilung, die einem Potenzgesetz unterliegt und die Zeit (in Stunden) zwischen dem Eintreffen und dem Lesen einer E-Mail beschreibt	0,312
N_{init}	Die Anzahl der Rechner, die zu Beginn der Simulation eine E-Mail in ihrem Posteingang haben	20
ch_{delay}	Die Verzögerung in Sekunden, die eine E-Mail bis zu ihrer Zustellung benötigt	0
p_{immune}	Die Wahrscheinlichkeit, dass ein Rechner nach einer Infektion immun gegen den E-Mail-Wurm werden kann	0
d_{immune}	Minimaler Verzweigungsgrad eines Knotens / minimale Adressbuchgröße eines Rechners, damit der Knoten/Rechner immun gegen den E-Mail-Wurm ist	∞

Die Standardwerte der Tabelle 5.1 entsprechen den Ergebnissen der Experimente, die im vorherigen Kapitel beschrieben sind. Die beiden Standardwerte der Wahrscheinlichkeiten p_{read} und p_{click} wurden aus Tabelle 4.1 hergeleitet: Die Spam-Experimente haben ergeben, dass bei $15{,}2\,\%$ der E-Mails dem Verweis gefolgt wurde. Da es sich bei den beiden Wahrscheinlichkeiten p_{read} und p_{click}, wie auf Seite 140 beschrieben, um unabhängige Ereignisse handelt, sollte das Produkt dieser Einzelwahrscheinlichkeiten also $0{,}152$ sein. Somit gibt es unendlich viele Möglichkeiten, diese Bedingung mit zwei reellen Zahlen aus dem Wertebereich 0 bis 1 zu erfüllen.

Wir haben uns letztendlich auf $p_{read} = 0{,}608$ und $p_{click} = 0{,}25$ festgelegt, da wir die Wahrscheinlichkeit, dass ein Benutzer eine Spam-Mail liest, höher einschätzen als die Wahrscheinlichkeit, dass der Benutzer auch noch einem eventuell darin vorhandenen Verweis folgt. Unserer Meinung nach liegt dies auf der einen Seite an der menschlichen Neugier, den Inhalt einer Nachricht zu kennen – vor allem, wenn der Betreff der E-Mail sehr interessant oder verlockend formuliert ist. Zudem wird

auf der anderen Seite mittlerweile in etlichen Aufklärungskampagnen darauf hingewiesen, dass es gefährliche Folgen haben kann, wenn man blindlings jedem Verweis folgt, den man unaufgefordert zugeschickt bekommt. Somit sollte p_{read} also größer als p_{click} sein. Bei den von uns gewählten Werten ist dies der Fall und das Produkt beider Wahrscheinlichkeiten entspricht ebenfalls der gemessenen Rate aus den Spam-Experimenten.

5.3. Ergebnisse

Um die Auswirkungen eines bestimmten Parameters auf die Verbreitung des E-Mail-Wurmes zu untersuchen, haben wir diesem Parameter sukzessive unterschiedliche Werte zugewiesen. Für jeden einzelnen Wert des zu untersuchenden Parameters haben wir mehrere Simulationsläufe mit unterschiedlichen Anfangswerten für den Zufallsgenerator durchgeführt. Aus den daraus resultierenden Verbreitungsverläufen haben wir abschließend den durchschnittlichen Verbreitungsverlauf berechnet. Dieser wiederum wird der entsprechenden Parameterbelegung als daraus resultierender Verbreitungsverlauf zugewiesen.

Möchte man beispielsweise einen Parameter untersuchen, der die ganzzahligen Werte 0 bis 10 annehmen kann, und führt für jeden dieser Werte 20 Simulationsläufe durch (um daraus den Durchschnitt zu berechnen), muss man die Simulation insgesamt also 220 Mal ausführen. Bei einer Simulationsgröße von 50 000 Rechnern, einer feingranularen Einteilung der Parameter mit bis zu mehr als 100 Werten und mit 40 Wiederholungen pro einzelnem Parameterwert, ist dies ein sehr rechenintensiver Vorgang. Dabei haben wir die Simulation dahingehend optimiert, dass diese auf dem Hochleistungscluster *bwGRID* ausgeführt werden konnte [bwG10].

Auf den folgenden Seiten präsentieren wir mehrere Verbreitungsverläufe, in denen jeweils der Einfluss eines bestimmten Parameters untersucht wurde. Die einzelnen Verbreitungsverläufe werden auf zweierlei Arten grafisch dargestellt: Zum einen als 3-dimensionale Darstellung und zum anderen als sogenannte Heatmap. Erstere trägt in einem 3-dimensionalen Koordinatensystem die Anzahl der Infektionen (y-Achse) gegen die Anzahl der simulierten Stunden und gegen den zu untersuchenden Parameter auf (z- und x-Achse). Diese Darstellung ermöglicht eine schnelle visuelle Erfassung der Verbreitungsverläufe. Da in einer 3-dimensionalen Darstellung jedoch Teile der Verbreitungsverläufe verdeckt sein können, sind in den entsprechenden Heatmaps die gleichen Werte, jedoch aus einer senkrechten Sicht von oben, dargestellt. Auf diese Weise wird der Verbreitungsverlauf 2-dimensional und keine Informationen sind mehr verdeckt. Die beiden Achsen repräsentieren zum einen die zeitliche Dimension und zum anderen die Belegung des zu untersuchenden Parameters. Die Anzahl der infizierten Rechner ist in den Heatmaps farblich kodiert und die dazugehörigen Farbskalen sind jeweils neben den Heatmaps abgebildet.

5.3.1. Anzahl der Startknoten

Um den Einfluss der Anzahl der Startknoten zu untersuchen, haben wir dem Parameter N_{init} jeweils 101 verschiedene Werte zugewiesen. Die ersten Simulationsdurchläu-

fe haben wir mit $N_{init} = 0$ Startknoten ausgeführt. Anschließend wurde der Parameter sukzessive um den Wert 5 bis zu einem Wert von $N_{init} = 500$ (Startknoten) erhöht. Für jede Belegung des Parameters haben wir 40 unterschiedliche Simulationen durchgeführt und daraus einen Durchschnittsverlauf berechnet. Das Ergebnis dieser Analyse ist in Abbildung 5.9 zu sehen.

Die verwendeten Startknoten wurden zufällig aus der Gesamtzahl der Knoten ausgesucht. Um Nebeneffekte zu vermeiden, die auf dem Verzweigungsgrad der verwendeten Startknoten basieren, wurden bei jeder Belegung des Parameters N_{init} die Knoten der vorherigen Belegung übernommen und fünf weitere, zufällig ausgewählte Knoten der Menge der bereits bestehenden Startknoten hinzugefügt. Auf diese Weise ist es nicht möglich, dass Startknoten mit einem hohen Verzweigungsgrad, die in der Menge der Startknoten bei einem niedrigen N_{init} enthalten sind, bei einer höheren Parameterbelegung hingegen fehlen, die Analyse des Parameters N_{init} beeinflussen.

In der 3-dimensionalen Darstellung des Verbreitungsverlaufs erkennt man gut den unruhigen Verlauf entlang der Achse, die die Anzahl der Startknoten repräsentiert. Diese Sprünge kommen dadurch zustande, dass dem Parameter N_{init} keine kontinuierlichen Werte zugewiesen wurden, sondern der Wert sukzessive um 5 (Startknoten) erhöht wurde. Dennoch lässt sich der leichte, lineare Anstieg entlang der Achse erkennen. Besser ist dieser Anstieg noch in Abbildung 5.10 zu sehen, die den 3-dimensionalen Verbreitungsverlauf aus einem seitlichen Blickwinkel zeigt. In dieser Abbildung ist zusätzlich eine rote Linie eingezeichnet, die dem Anstieg der infizierten Rechner in Abhängigkeit von der Anzahl der Startknoten entspricht.

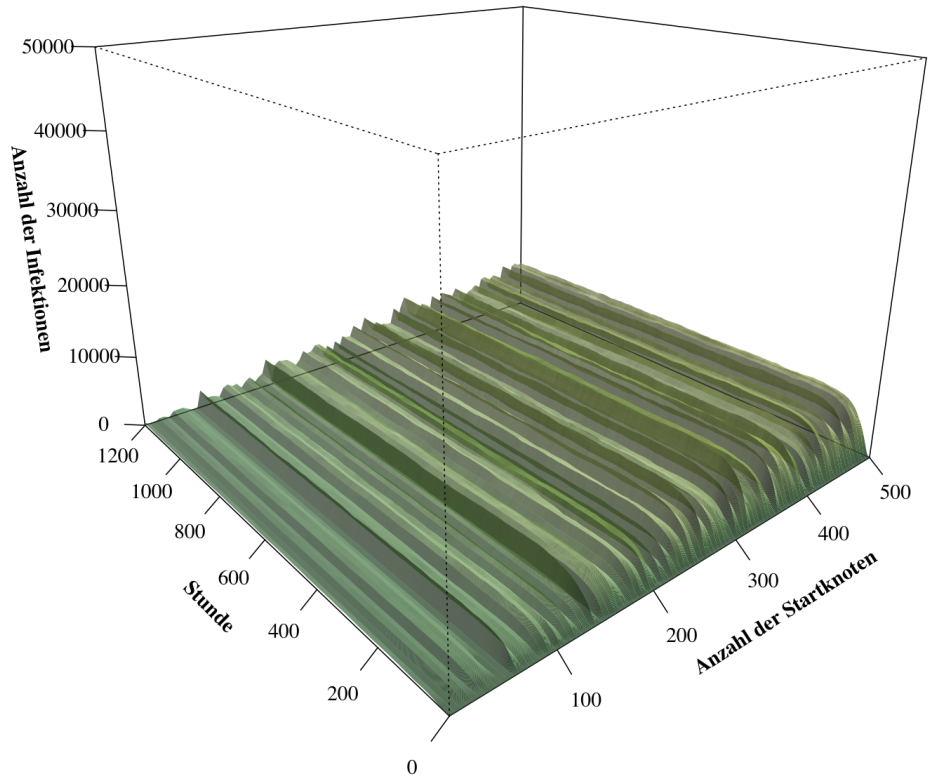
Aus diesem Anstieg folgt, dass die Anzahl der infizierten Rechner proportional von der Anzahl der Rechner abhängt, die initial eine Spam-Mail im Posteingang ihres E-Mail-Clients haben. Dass dieser Anstieg nicht monoton steigend ist, sondern auch ab und zu wieder leicht abfällt, ist wahrscheinlich der statistischen Streuung der Zufallswerte der Simulationen geschuldet. Zwar wird versucht, dem durch die wiederholten Durchführungen der Simulation für eine Parameterbelegung entgegenzuwirken, jedoch reichen 40 Wiederholungen dafür nicht vollständig aus. Da der Anstieg jedoch bereits mit dieser Anzahl der Wiederholungen gut zu erkennen ist und die einzelnen Simulationen trotz Rechencluster sehr zeitaufwendig sind, haben wir keine weiteren Simulationsläufe für diesen Parameter durchgeführt.

Die Verbreitungsverläufe der Belegungen

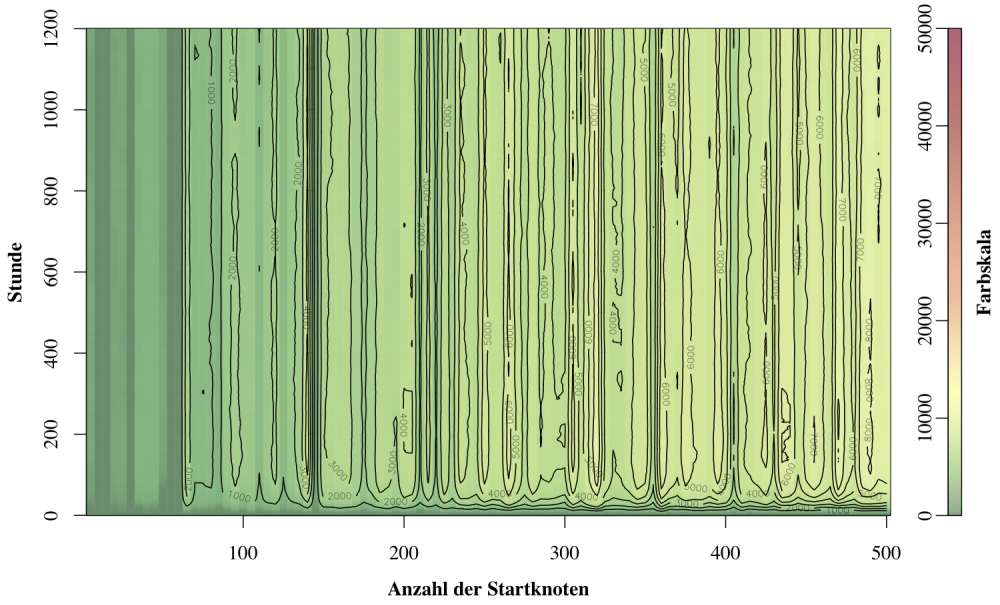
$$N_{init} \in \{5, 40, 100, 200, 300, 400, 500\}$$

sind in Abbildung 5.11 dargestellt. Auch hier erkennt man, dass der Anstieg nicht linear ist: Der Abstand zwischen den Verbreitungsverläufen mit 300 und 400 Startknoten ist geringer ist als der Abstand zwischen den Verbreitungsverläufen mit 200 und 300 Startknoten. Der Abstand zwischen den oberen beiden Verläufen, deren Differenz ebenfalls nur 100 Startknoten beträgt, ist hingegen wieder wesentlich größer.

Allen sieben Verbreitungsverläufen der Abbildung 5.11 hingegen gemein ist die gleiche charakteristische Form: Nach einem raschen Anstieg der Infektionsrate erreicht sie ihr Maximum nach ca. 150 bis 300 Stunden – also ungefähr 6 bis 12 Tage nach *Ausbruch* des E-Mail-Wurms. In der Abbildung ist dieser Bereich durch die beiden vertikalen Linien gekennzeichnet. Anschließend fällt die Anzahl der infizierten



(a) 3-dimensionale Darstellung des Verbreitungsverlaufs



(b) Verbreitungsverlauf als Heatmap

Abbildung 5.9.: Einfluss der Anzahl der Startknoten
Diese Abbildung zeigt den Einfluss der Anzahl der Rechner, die initial eine Spam-Mail in ihrem Posteingang haben, auf den Verbreitungsverlauf des E-Mail-Wurms. Die Anzahl der Startknoten entspricht dem Parameter N_{init}.

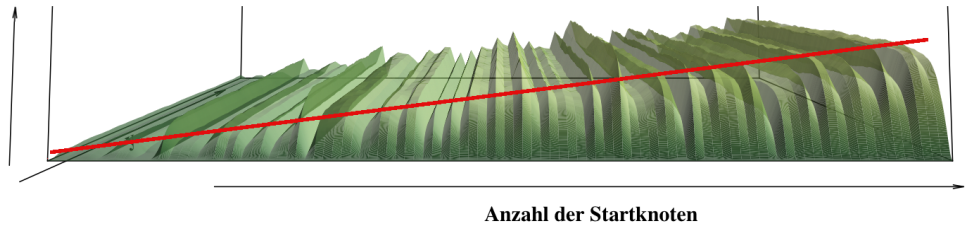


Abbildung 5.10.: Seitliche Ansicht des 3-dimensionalen Verbreitungsverlaufs
Dies ist eine seitliche Ansicht des 3-dimensionalen Verbreitungsverlaufs aus Abbildung 5.9. Die eingezeichnete, rote Linie verdeutlicht den proportionalen Anstieg der infizierten Rechner in Abhängigkeit von der Anzahl der Startknoten.

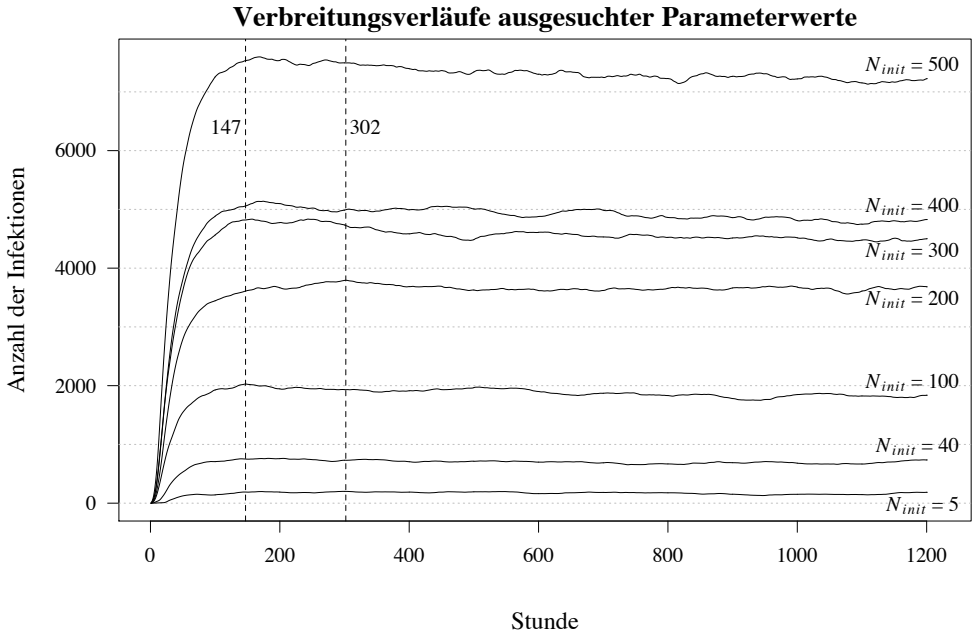


Abbildung 5.11.: Verbreitungsverläufe bei unterschiedlich vielen Startknoten
Dies sind die Verbreitungsverläufe, die sich bei 5, 40, 100, 200, 300, 400 und 500 Startknoten ergeben. Die eingezeichneten vertikalen Linien befinden sich an den Stellen des kleinsten und des größten Maximums dieser sieben Verbreitungsverläufe.

Rechner bei jeder Belegung des Parameters N_{init} leicht ab und pendelt sich dann recht stabil um einen festen Wert ein. Auch fällt auf, dass die Verbreitungsverläufe einer hohen Belegung, also mit eher vielen Startknoten, den Zeitpunkt des raschen Anstiegs schneller erreichen. Dies ist vor allem an dem untersten Verlauf der Abbildung zu sehen, der im Vergleich zu den anderen Verläufen zu Beginn wesentlich langsamer ansteigt.

Bei den Simulationen der nächsten Abschnitte verwenden wir den Standardwert $N_{init} = 20$ (Startknoten), d. h. dass bei allen noch folgenden Simulationen $0{,}04\,\%$ aller Rechner zu Beginn eine initiale Spam-Mail in ihrem Posteingang haben. Tabelle 5.2 zeigt eine Übersicht über die Startknoten und enthält ebenfalls pro Startknoten die Anzahl der Knoten, die direkt von den einzelnen Startknoten aus erreichbar sind. Im Schnitt hat jeder der zufällig gewählten Startknoten $4{,}25$ Nachbarknoten. Die in der Tabelle angegebene Knotennummer ist eine interne, fortlaufende Nummerierung der Knoten, die bei der Generierung des E-Mail-Netzes verwendet wurde, um die Knoten eindeutig zu referenzieren.

Tabelle 5.2.: Bei den Simulationen verwendete Startknoten
Die in dieser Tabelle angegebenen Knoten haben bei den noch folgenden Simulationen eine initiale Spam-Mail in ihrem Posteingang. Neben der intern verwendeten Knotennummer ist für jeden Knoten jeweils auch die Anzahl der direkten Nachbarn angegeben.

Knotennr.	Anzahl der Nachbarn	Knotennr.	Anzahl der Nachbarn
6843	13	18511	4
9579	7	35054	1
12539	1	41187	2
22725	5	43170	1
46746	1	43701	1
918	10	2437	20
14055	2	27664	7
20228	1	30130	2
22212	2	31435	1
41749	2	41582	2

Die Knoten der Tabelle entsprechen den ersten 20 Knoten der zufällig ausgewählten Knoten zur Generierung des präsentierten Verbreitungsverlaufs. Die oberen fünf Startknoten der linken Teiltabelle bilden die erste Belegung des Parameters N_{init}, dann kommen bei der zweiten Belegung die unteren fünf Knoten der linken Teiltabelle hinzu, bei der dritten Belegung die ersten fünf Knoten der rechten Teiltabelle und die vierte Belegung besteht aus alles 20 Knoten.

5.3.2. Wahrscheinlichkeit, dass eine E-Mail gelesen wird

Der nächste Parameter, dessen Einfluss auf die Verbreitung des E-Mail-Wurms untersucht werden soll, ist der Parameter p_{read}. Dieser steuert die Wahrscheinlichkeit dafür, dass ein Benutzer eine neu eingetroffene E-Mail liest. Der Verbreitungsverlauf des

E-Mail-Wurms, in Abhängigkeit von diesem Parameter, ist in Abbildung 5.12 dargestellt.

In der Abbildung ist sowohl wieder eine 3-dimensionale Darstellung als auch eine entsprechende Heatmap zu sehen. Für die Analyse des Parameters p_{read} haben wir diesen sukzessive mit Werten aus dem Bereich $\{0,\ 1 \times 10^{-2},\ 2 \times 10^{-2},\ \ldots,\ 1\}$ belegt und die entsprechenden Verbreitungsverläufe berechnet.

Wenig überraschend verbreitet sich der E-Mail-Wurm schneller, je höher die Wahrscheinlichkeit ist, mit der die Benutzer neu eintreffende E-Mails lesen. Interessant ist, dass diese Wahrscheinlichkeit erst ab einem Wert von ca. $0{,}2$ eine Auswirkung auf die Wurmverbreitung zeigt. Bei niedrigeren Werten reicht diese Wahrscheinlichkeit in Kombination mit den anderen Parameterbelegungen nicht dafür aus, dass sich der Wurm verbreiten kann.

Bei höheren Werten von p_{read} kann sich der Wurm jedoch besser verbreiten und erreicht eine Lebenszeit bis zum Ende der Simulationsdauer von 50 Tagen. In dem Wertebereich $p_{read} \in [0{,}15,\ 0{,}25]$ kommt es teilweise noch zu einem abrupten Abfall der Infektionsrate, weil der E-Mail-Wurm bei allen 40 Simulationswiederholungen einer Belegung *ausstirbt*. In der Heatmap der Abbildung 5.12b ist exemplarisch solch eine Situation durch einen roten Kreis hervorgehoben. Auch in der 3-dimensionalen Ansicht sind diese Situationen teilweise zu erkennen.

Die bereits im vorherigen Abschnitt angesprochene, charakteristische Form der Verbreitungsverläufe ist auch bei der Untersuchung des Parameters p_{read} zu beobachten: Nach einem raschen Anstieg der Infektionsrate, erreicht diese ihr Maximum und pendelt sich anschließend auf ein konstantes Niveau ein. Abbildung 5.13 zeigt sechs Verbreitungsverläufe, die sich bei einer Belegungen von

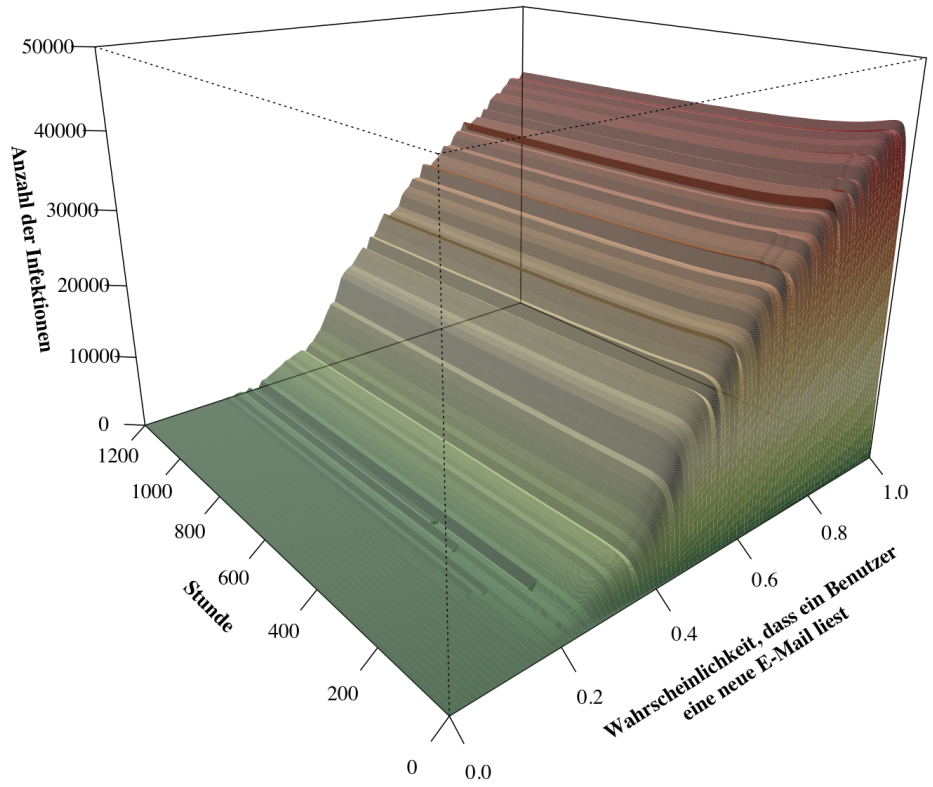
$$p_{read} \in \{0{,}25,\ 0{,}35,\ 0{,}5,\ 0{,}7,\ 0{,}9,\ 1\}$$

ergeben. An dem Verbreitungsverlauf der Belegung $p_{read} = 0{,}25$ kann man ebenfalls die bereits oben angesprochene Situation gut erkennen, bei der die Infektionsrate abrupt einbricht. Diese Situation verursacht den plötzlichen Abfall des Verbreitungsverlaufs nach ca. 140 Stunden. Die in der Abbildung 5.12b durch den roten Kreis gekennzeichnete Stelle entspricht dem plötzlichen Abfall genau dieses Verbreitungsverlaufs.

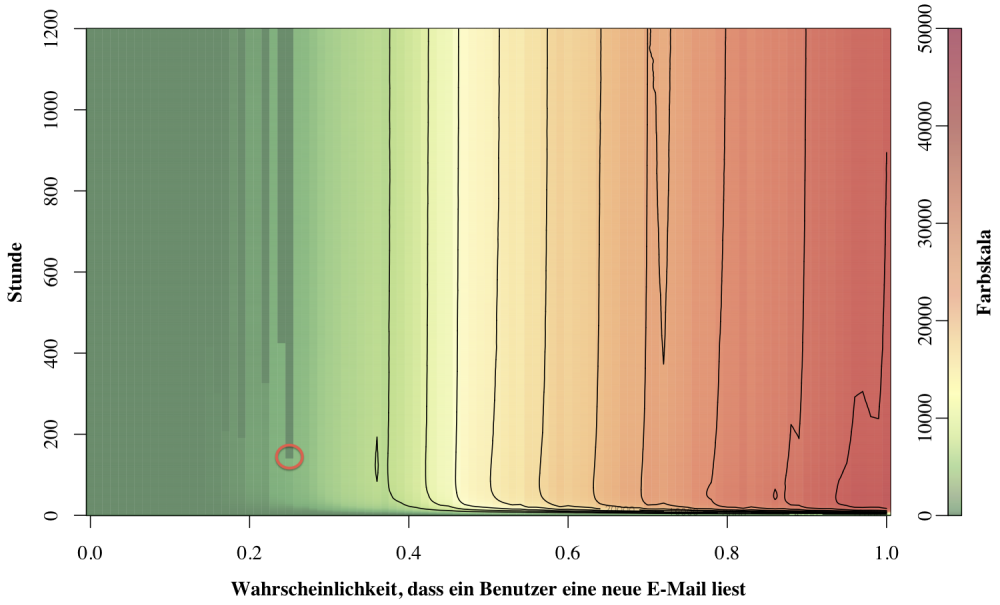
5.3.3. Wahrscheinlichkeit, dass einem Verweis gefolgt wird

Der Parameter p_{click}, der die Wahrscheinlichkeit angibt, mit der ein Benutzer dem Verweis einer E-Mail folgt, hat einen ähnlichen Einfluss auf die Verbreitung des Wurms wie der im vorherigen Abschnitt untersuchte Parameter p_{read}. Dies liegt daran, dass man, wie bereits beschrieben, die beiden Einzelereignisse *Lesen der E-Mail* und *Folgen des Verweises* auch zu einem Ereignis zusammenfassen könnte. Bei der Analyse des Parameters p_{click} haben wir diesem Werte aus dem Bereich 0 bis 1 zugewiesen. Der Verbreitungsverlauf ist in beiden bereits bekannten Darstellungsformen in Abbildung 5.14 dargestellt.

Auch hier reichen niedrige Werte des Parameters p_{click} nicht aus, damit sich der E-Mail-Wurm überhaupt verbreitet. Jedoch ist bei diesem Parameter der Schwellwert, ab dem der Wurm die vollen 50 Simulationstage *überleben* kann, geringer als



(a) 3-dimensionale Darstellung des Verbreitungsverlaufs



(b) Verbreitungsverlauf als Heatmap

Abbildung 5.12.: Einfluss der Wahrscheinlichkeit, eine E-Mail zu lesen
Diese Abbildung zeigt den Einfluss der Wahrscheinlichkeit, mit der ein Benutzer eine neue E-Mail liest, auf den Verbreitungsverlauf des E-Mail-Wurms. Diese Wahrscheinlichkeit wird in der Simulation durch den Parameter p_{read} gesteuert.

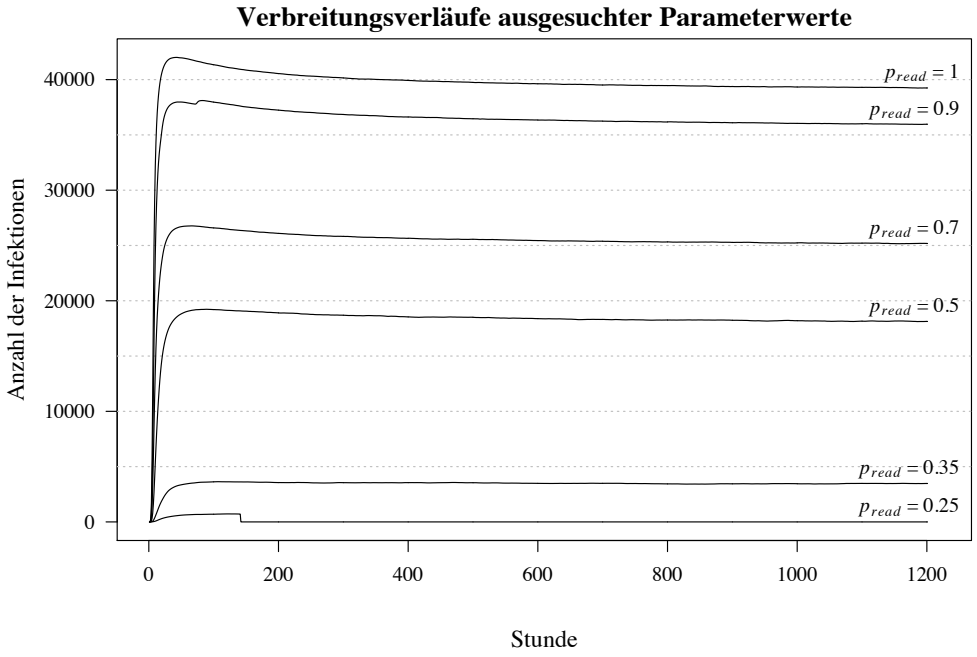
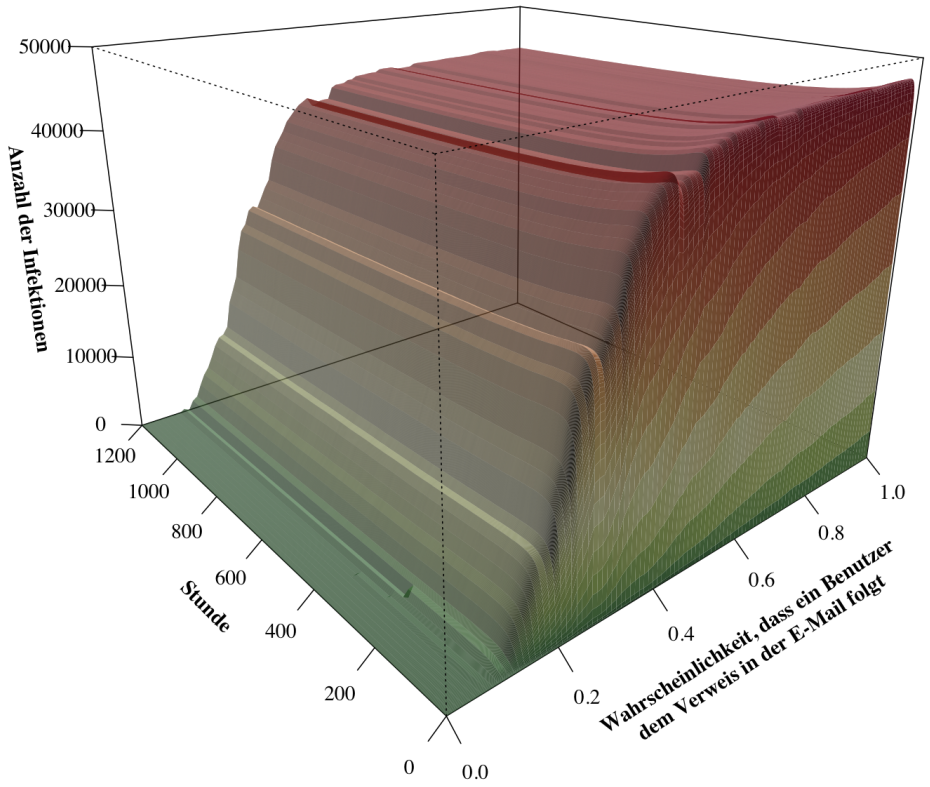


Abbildung 5.13.: Verbreitungsverläufe bei unterschiedlichen Lese-Wahrscheinlichkeiten Dies sind sechs Verbreitungsverläufe, die sich bei unterschiedlichen Belegungen von p_{read} ergeben. Dabei handelt es sich um die Wahrscheinlichkeit dafür, dass eine neue E-Mail vom Empfänger gelesen wird.

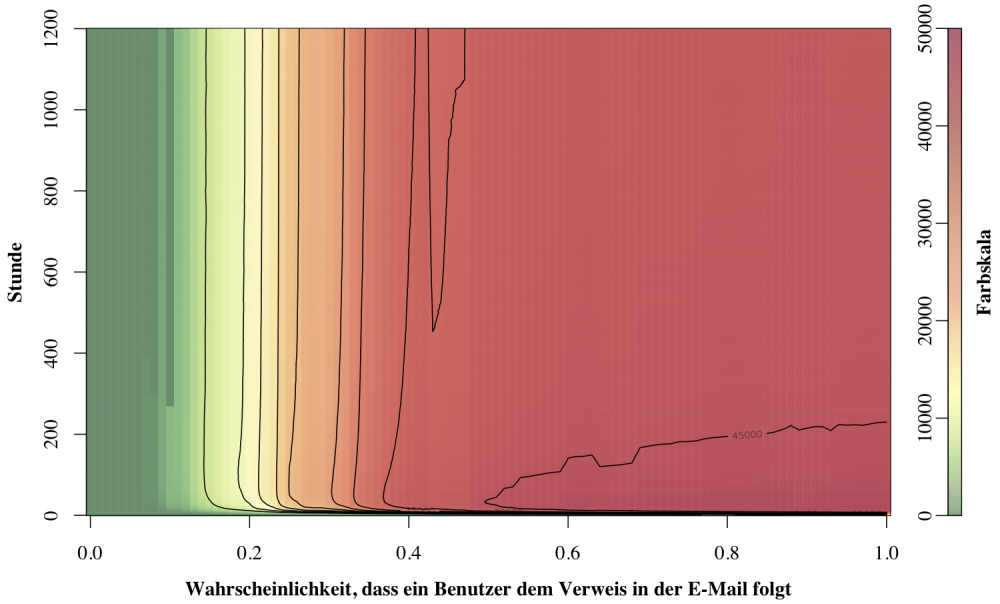
bei p_{read}: Dieser liegt ungefähr bei einem Wert von $p_{click} = 0{,}09$. Des Weiteren ist der Anstieg der Verbreitungsverläufe, der nach diesem Schwellwert einsetzt, weitaus stärker als der entsprechende Anstieg bei den Verbreitungsverläufen der Analyse des Parameters p_{read}.

Die Grafik der Abbildung 5.15 macht diesen Unterschied zwischen den beiden Verbreitungsverläufen der beiden Parameter p_{click} und p_{read} noch deutlicher. In der Grafik sind zwei Querschnitte durch die 3-dimensionalen Verbreitungsverläufe dieser beiden Parameter abgebildet – jeweils 600 Stunden nach dem Start der Simulation. Man erkennt den deutlich schnelleren Anstieg der Infektionsrate bei dem Parameter p_{click}. Ebenfalls erkennt man in diesem Vergleich der beiden Parameter, dass bei dem Parameter p_{click} bereits kleinere Wahrscheinlichkeiten (im Vergleich zu p_{read}) ausreichen, damit der Wurm nicht bereits zu Beginn der Simulation *ausstirbt*.

Der Grund für diesen Unterschied liegt an den unterschiedlichen Standardwerten der beiden Parameter p_{read} und p_{click}. Bei der Analyse des Parameters p_{click} ist der Parameter p_{read} mit dessen Standardwert von $0{,}608$ belegt. Während der Analyse des Parameters p_{read} ist dem Parameter p_{click} hingegen der niedrigere Standardwert von $0{,}25$ zugewiesen. Somit ist die Gesamtwahrscheinlichkeit, dass eine E-Mail gelesen und dem Verweis gefolgt wird, tendenziell während der Analyse des Parameters p_{click} leicht höher.



(a) 3-dimensionale Darstellung des Verbreitungsverlaufs



(b) Verbreitungsverlauf als Heatmap

Abbildung 5.14.: Einfluss der Wahrscheinlichkeit, dem Verweis zu folgen
Diese Abbildung zeigt den Einfluss der Wahrscheinlichkeit, mit der ein Benutzer dem Verweis einer E-Mail folgt, auf den Verbreitungsverlauf des E-Mail-Wurms. Diese Wahrscheinlichkeit wird in der Simulation durch den Parameter p_{click} gesteuert.

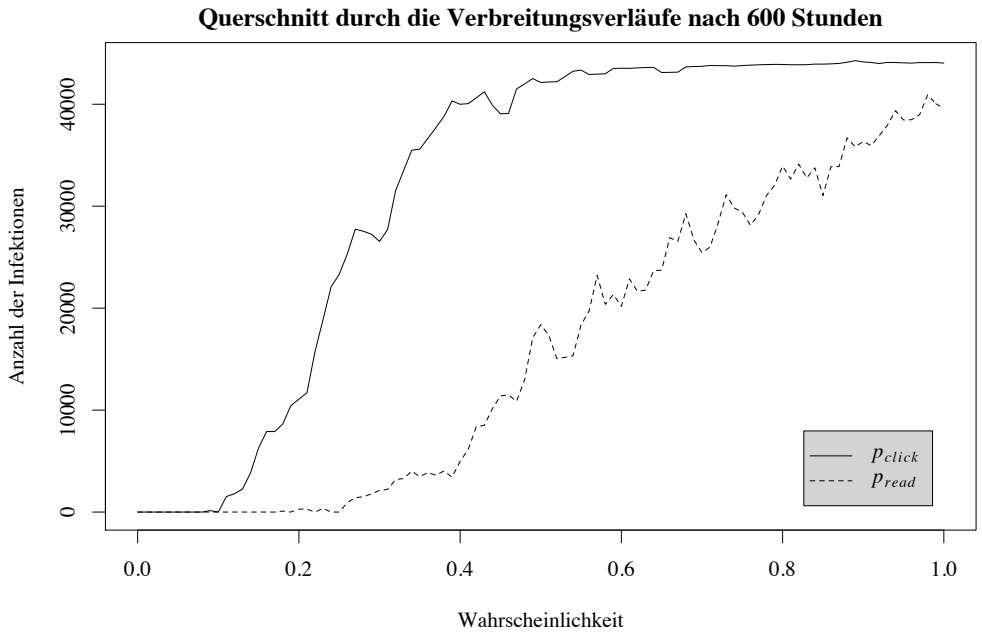


Abbildung 5.15.: Vergleich von p_{read} und p_{click}
Anhand der Querschnitte durch die Verbreitungsverläufe sieht man, dass mit hohen Wahrscheinlichkeiten der beiden untersuchten Parameter, die Anzahl der Infektionen bei dem Parameter p_{click} wesentlich schneller steigt als bei dem Parameter p_{read}.

5.3.4. Zeitspanne zwischen dem Eintreffen und Lesen einer E-Mail

Das Mailserver-Experiment, das in Abschnitt 4.3 beschrieben ist, hat gezeigt, dass die Zeitspannen, die zwischen dem Eintreffen und dem Lesen einer E-Mail vergehen, einer Verteilung folgen, die einem Potenzgesetz

$$p(x) = C \cdot x^{-\alpha},\ \alpha \in \mathbb{R}_{>0}$$

genügt. Da der Vorfaktor C lediglich die Funktion skaliert, haben wir diesen nicht separat untersucht. Bei der Analyse des Einflusses der Zeitspannen auf die Verbreitung des E-Mail-Wurms haben wir uns dementsprechend auf den Parameter $\alpha_{reading}$ beschränkt. Dieser entspricht dem Exponenten α obiger Verteilung.

Abbildung 5.16 zeigt vier verschiedene Wahrscheinlichkeitsdichten solch einer Verteilung. Der Vorfaktor ist mit $C = 0{,}312$ bei allen vier Dichtefunktionen konstant. Der einzige Unterschied besteht in den verschiedenen Exponenten der Dichtefunktionen. In der Legende der Abbildung sind die konkreten Werte der Exponenten aller vier Verteilungen angegeben. Wie man sieht, werden kurze Zeitspannen immer wahrscheinlicher, je größer der Exponent gewählt wird.

Um den Einfluss dieser Zeitspannen auf die Verbreitung des E-Mail-Wurms zu untersuchen, haben wir dem Parameter $\alpha_{reading}$ sukzessive Werte aus dem Bereich $[0,\ 0{,}02,\ 0{,}04,\ \ldots,\ 2]$ zugewiesen. Das Ergebnis dieser Analyse ist in den beiden bekannten Darstellungsformen in Abbildung 5.17 visualisiert.

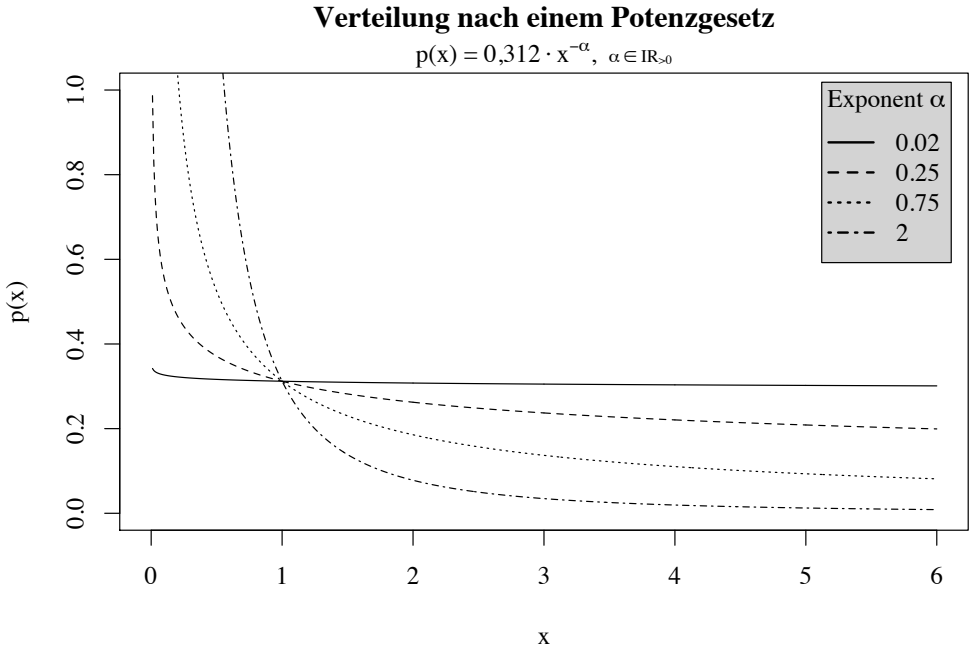
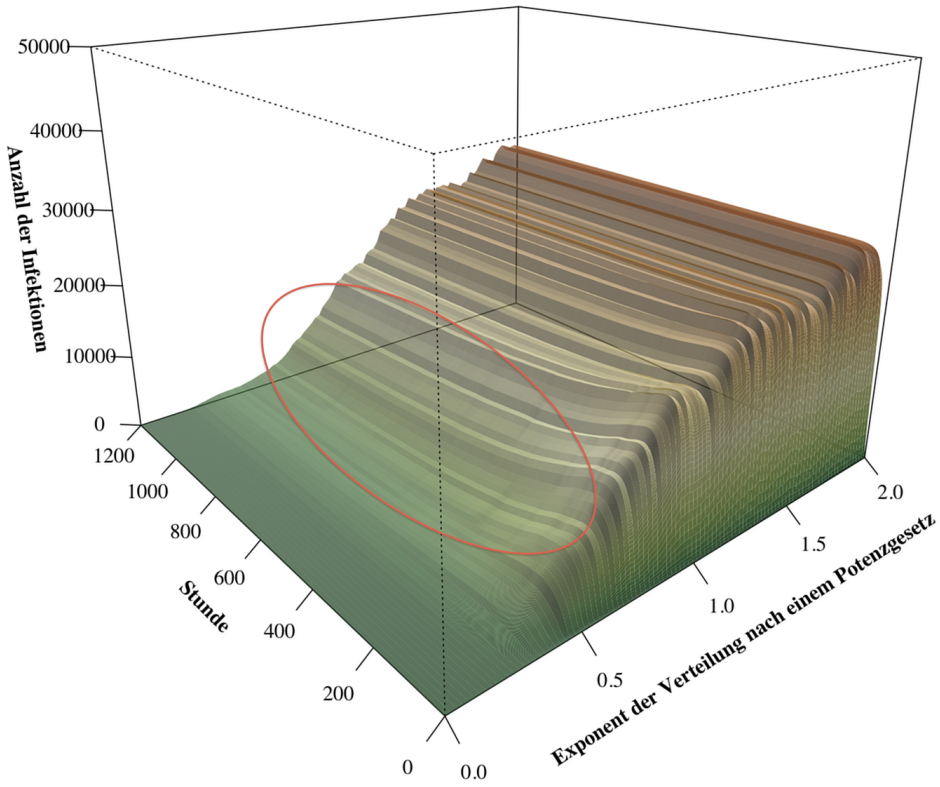


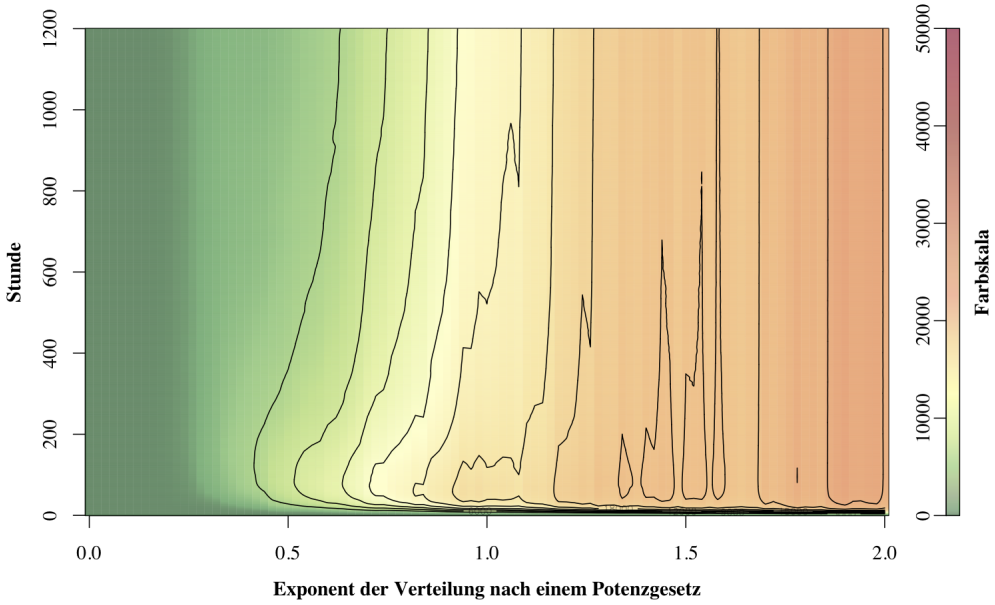
Abbildung 5.16.: Verteilungen, die einem Potenzgesetz genügen
Dies sind vier unterschiedliche Wahrscheinlichkeitsdichten, die alle einem Potenzgesetz folgen. Der Vorfaktor ist mit $C = 0{,}312$ bei allen Dichtefunktionen konstant, die Exponenten α der einzelnen Verteilungen sind jedoch unterschiedlich und jeweils in der Legende angegeben.

Die kürzeren Zeitspannen zwischen dem Eintreffen und dem Lesen der E-Mails, die sich bei einem wachsenden Exponenten $\alpha_{reading}$ ergeben, spiegeln sich in der Simulation durch eine schnellere Verbreitung des E-Mail-Wurms wider. Entsprechend steigt die Anzahl der infizierten Rechner mit dem Parameter $\alpha_{reading}$ proportional an. Auffällig ist, dass sich in dem Wertebereich zwischen $0{,}35$ und 1 eine Art Senke ergibt. Diese ist in der 3-dimensionalen Darstellung der Abbildung 5.17 durch ein rotes Oval hervorgehoben.

Auch in Abbildung 5.18, die sieben einzelne Verbreitungsverläufe zeigt, ist dieser Effekt zu erkennen: Während bei den drei Verbreitungsverläufen mit $\alpha > 1$ die Anzahl der infizierten Rechner nach einer gewissen Zeit relativ konstant bleibt, ist bei den Verläufen mit kleineren Exponenten ein deutlich länger anhaltender und stärkerer Abfall der Rechnerinfektionen zu erkennen. Dies liegt wahrscheinlich daran, dass in diesem Wertebereich von $\alpha_{reading}$ die Zeitspannen zwischen dem Eintreffen und dem Lesen der E-Mails noch so groß sind, dass, nachdem das Maximum an Rechnerinfektionen erreicht ist, Rechner schneller gesäubert werden als Neuinfektionen (bedingt durch die hohen Zeitspannen) auftreten.



(a) 3-dimensionale Darstellung des Verbreitungsverlaufs



(b) Verbreitungsverlauf als Heatmap

Abbildung 5.17.: Einfluss der Zeitdauer zwischen Eintreffen und Lesen einer E-Mail
Die Abbildung zeigt den Einfluss der Zeitspannen zwischen dem Eintreffen und dem Lesen einer E-Mail auf den Verbreitungsverlauf des E-Mail-Wurms. Die Zeitspannen genügen einer Verteilung, die einem Potenzgesetz folgt. Der untersuchte Parameter dieser Analyse ist der Exponent des Potenzgesetzes.

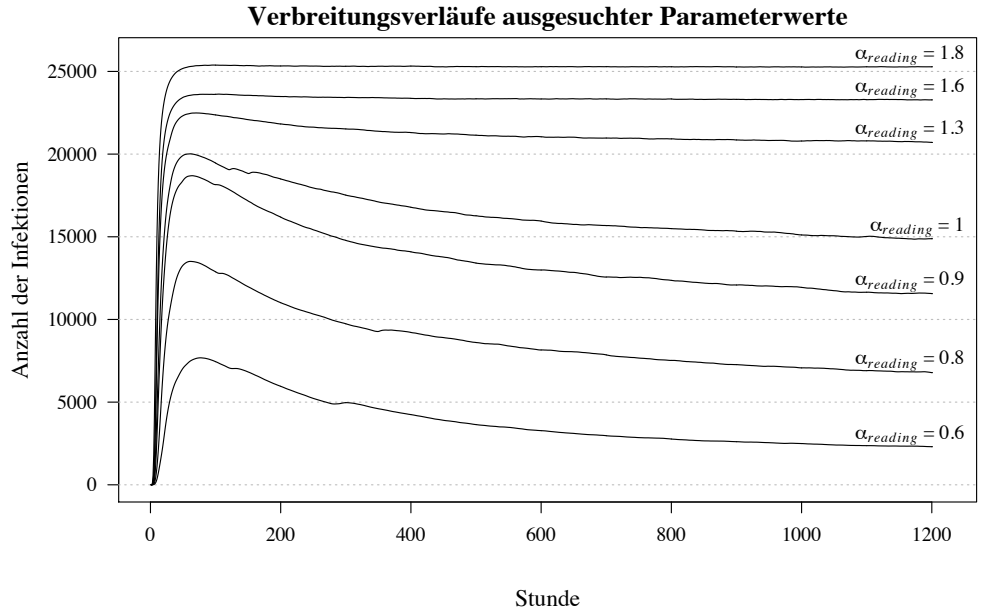


Abbildung 5.18.: Verbreitungsverläufe bei unterschiedlichen Exponenten $\alpha_{reading}$
Dies sind sieben Verbreitungsverläufe, die sich bei unterschiedlichen Belegungen von $\alpha_{reading}$ ergeben. Dieser Parameter entspricht dem Exponenten des Potenzgesetzes, das die Verteilung der Zeitspannen zwischen dem Eintreffen und dem Lesen einer E-Mail beschreibt.

5.3.5. Künstliche Verzögerung in der Zustellgeschwindigkeit einer E-Mail

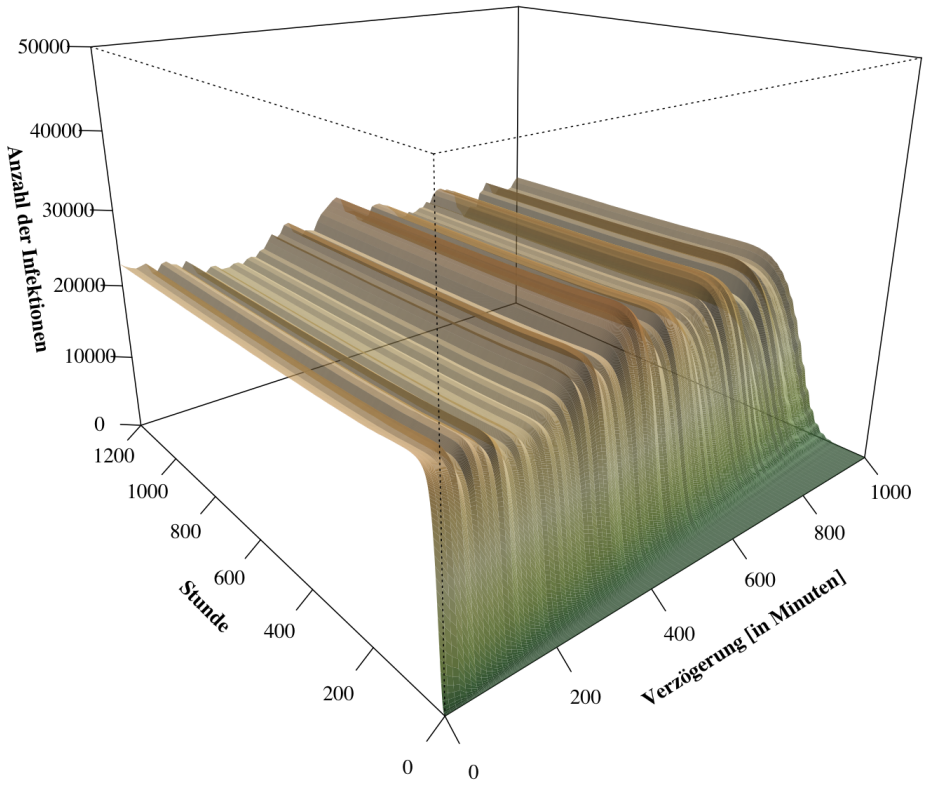
Die Zeitspannen, die zwischen dem Eintreffen und dem Lesen einer E-Mail vergehen, sind rein durch die menschlichen Benutzer bedingt. In der Regel gibt es keine technische Möglichkeit diese zu vergrößern, um die Verbreitung des E-Mail-Wurms zu entschleunigen oder gar zu verhindern. Jedoch kann man aus technischer Sicht diese Zeit indirekt beeinflussen, indem man bei der Zustellung der E-Mails eine künstliche Verzögerung einbaut. Den Effekt einer solch künstlichen Verzögerung bezüglich der Zustellgeschwindigkeit von E-Mails haben wir ebenfalls untersucht.

Bei den von uns durchgeführten Simulationen enthält der Simulationsparameter ch_{delay} die Anzahl der Sekunden, die eine E-Mail vor ihrer Zustellung zurückgehalten wird. Erst nach Ablauf der durch den Parameter spezifizierten Zeitspanne, wird eine E-Mail an den Posteingang des Empfängers ausgeliefert. Für die Analyse des Effektes dieses Parameters haben wir diesem sukzessive Werte aus folgendem Wertebereich zugewiesen:

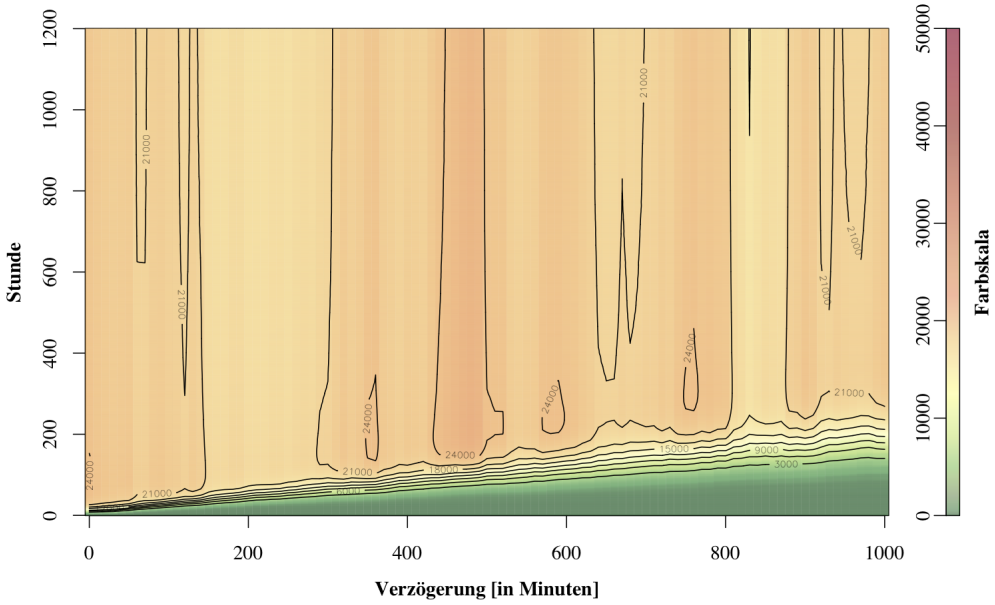
$$ch_{delay} \in \{x \cdot 600 \,|\, x \in \{0, \ldots, 100\}\}$$

Das Ergebnis dieser Analyse ist in Form der bekannten Verbreitungsverläufe in Abbildung 5.19 dargestellt.

In den Verbreitungsverläufen erkennt man, dass sich eine künstliche Verzögerung bezüglich der Zustellgeschwindigkeit der E-Mails unmittelbar auf den Zeitpunkt auswirkt, ab dem die Anzahl der Infektionen rasant ansteigt. Vor allem in der Heatmap der



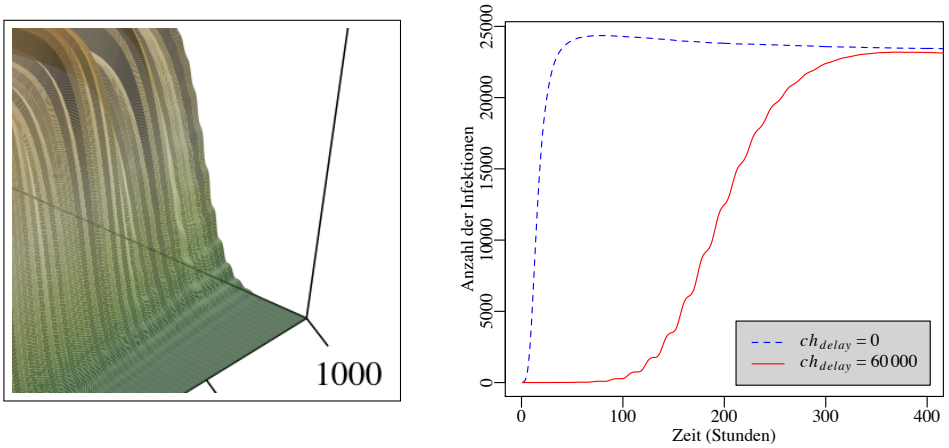
(a) 3-dimensionale Darstellung des Verbreitungsverlaufs



(b) Verbreitungsverlauf als Heatmap

Abbildung 5.19.: Einfluss einer Verzögerung in der Zustellgeschwindigkeit einer E-Mail Die Abbildung zeigt den Einfluss einer künstlichen Verzögerung in der Zustellgeschwindigkeit einer E-Mail auf den Verbreitungsverlauf des Wurms. Anstatt die E-Mail sofort zuzustellen, trifft die E-Mail erst nach einer mehrminütigen Verzögerung im Postfach des Empfängers ein.

Abbildung 5.19 erkennt man den linearen Zusammenhang zwischen der Verzögerung und der Anzahl der Rechnerinfektionen. Dieser Zusammenhang spiegelt sich in der geraden Kante zwischen dem grünlichen Dreieck (am unteren Rand der Abbildung) und dem Rest der Abbildung wider. Abbildung 5.20b verdeutlicht diesen Effekt nochmals. Hier erkennt man klar den langsameren Anstieg in der Verbreitung des Wurms, der durch die künstliche Verzögerung in der Zustellgeschwindigkeit der E-Mails erzeugt wird.



(a) Ausschnitt aus Abbildung 5.19a

(b) Verbreitungsverläufe ohne und mit 60 000 Sekunden künstlicher Verzögerung

Abbildung 5.20.: Flatternde Verbreitungsverläufe durch verzögerte E-Mail-Zustellung
Teil (a) zeigt einen vergrößerten Ausschnitt aus Abbildung 5.19a. Teil (b) verdeutlicht nochmals den Effekt der künstlichen Verzögerung. In beiden Darstellungen ist klar ein Flattern der Verläufe bei einer hohen Parameterbelegung zu erkennen.

In der Abbildung ist auch ein Nebeneffekt der langsameren Zustellung zu erkennen: Der Verbreitungsverlauf des E-Mail-Wurms fängt immer mehr an zu flattern, je länger die Zustellung der E-Mails künstlich hinausgezögert wird. Dieser Effekt ist ebenfalls bereits in der 3-dimensionalen Darstellung des Verbreitungsverlaufs zu sehen. Deshalb enthält Abbildung 5.20a einen vergrößerten Ausschnitt des 3-dimensionalen Verbreitungsverlaufs. Das Flattern kommt dadurch zu Stande, dass durch die Verzögerung Phasen entstehen, in denen immer mehr E-Mails fast zeitgleich verschickt werden. Diese Phasen verursachen natürlich auch ruckartige Anstiege im Verbreitungsverlauf des E-Mail-Wurms, die sich grafisch als eine Art Flattern bemerkbar machen.

In der Realität, vor allem bei der geschäftlichen Nutzung des Mediums E-Mail, ist solch eine Verzögerung in der Zustellgeschwindigkeit von E-Mails nur schwer vorstellbar. Gerade die verzögerungsfreie Zustellung von E-Mails ist wohl einer der wesentlichen Gründe für deren Beliebtheit. Jedoch zeigen unsere Analysen, dass bereits durch eine Verzögerung von nur 200 Minuten (also weniger als 3,5 Stunden) der kritische Zeitpunkt, ab dem die Anzahl der infizierten Systeme rasant ansteigt, um mehr als 2 Tage nach hinten verschoben werden kann. Diese Zeitspanne könnte von Sicherheitsorganisationen oder vom Hersteller der verwundbaren Anwendung (im Falle unserer Simulationen also der Hersteller des verwundbaren Browsers) dazu genutzt werden, um die Benutzer vor dem E-Mail-Wurm zu warnen oder im besten Falle neue

Sicherheitspatches zu veröffentlichen. Dadurch ließe sich die Verbreitung des E-Mail-Wurms frühzeitig eindämmen.

Damit nicht zu große Einschränkungen bei der Nutzung des Mediums E-Mail auftreten, wäre auch eine teilweise Umsetzung solch einer künstlichen Verzögerung denkbar. Beispielsweise könnte man eine Verzögerung der E-Mail mit bestimmten Bedingungen verknüpfen – etwa folgenden:

1. Bei einer ausschließlich privaten E-Mail-Nutzung
2. Lediglich bei E-Mails, in denen ein Verweis oder ein Anhang enthalten
3. Lediglich bei Zielsystemen, die keinen ausreichenden Patch-Level aufweisen

Gerade die zweite Bedingung ist einfach zu realisieren und würde die Benutzung von E-Mails ohne Verweis oder Dateianhang in keinerlei Art und Weise einschränken. Ebenfalls lassen sich die Bedingungen beliebig kombinieren oder weiter ausarbeiten. So ist beispielsweise auch eine Art Zertifizierung denkbar, die es geprüften Instanzen erlaubt, weiterhin E-Mails verzögerungsfrei zu verschicken, auch wenn diese einen Verweis oder einen Anhang enthalten. Derartige Überlegungen gehen aber über das Thema der vorliegenden Dissertation hinaus und werden hier nicht weiter vertieft.

5.3.6. Zeitspanne zwischen der Infektion und der Säuberung eines Rechners

Auch die Zeit, die zwischen der Infektion und der Reinigung eines Rechners vergeht, nimmt Einfluss auf die Verbreitung des E-Mail-Wurms. Die Auswertung der Keylogger-Daten (siehe Abschnitt 4.4) hat gezeigt, dass diese exponentialverteilt sind. Exponentialverteilungen werden durch einen Parameter $\lambda \in \mathbb{R}_{>0}$ beschrieben. Wie sich dieser Parameter auf die Verteilung der beschriebenen Daten auswirkt, zeigt Abbildung 5.21: Je größer die Rate der Exponentialverteilung gewählt wird, desto wahrscheinlicher werden kleine Werte.

Bei den von uns durchgeführten Simulationen wird ebenfalls die Rate der Exponentialverteilung dazu benutzt, die Zeiten zwischen einer Rechnerinfektion und -reinigung zu steuern. Diese wird durch den Simulationsparameter $\lambda_{cleaning}$ beschrieben. Zur Analyse des Einflusses der Zeitspannen zwischen Infektion und Reinigung eines Rechners, haben wir dem Parameter $\lambda_{cleaning}$ sukzessive Werte aus dem Bereich $[0,\ 0{,}03,\ 0{,}06,\ \ldots,\ 3]$ zugewiesen. Die aus dieser Analyse resultierenden Verbreitungsverläufe sind in Abbildung 5.22 dargestellt. Bei der 3-dimensionalen Darstellung sind aus Darstellungsgründen die Achsen im Vergleich zu den vorherigen 3-dimensionalen Verbreitungsverläufen anders ausgerichtet.

Da mit größeren Raten die Zeitspannen zwischen Rechnerinfektion und -reinigung immer geringer werden, nimmt die Anzahl der Infektionen wenig überraschend mit ansteigendem $\lambda_{cleaning}$ durchschnittlich ab. Dies liegt daran, dass durch die schnelleren Reparaturzeiten im Schnitt weniger Rechner infiziert sind. Bei Raten, die noch größer sind als die hier dargestellten Werte, würde dies soweit führen, dass sich der E-Mail-Wurm irgendwann langsamer verbreiten würde als die Rechner von diesem bereinigt werden. Dementsprechend würde der Wurm bei entsprechend hohen Werten von $\lambda_{cleaning}$ aussterben.

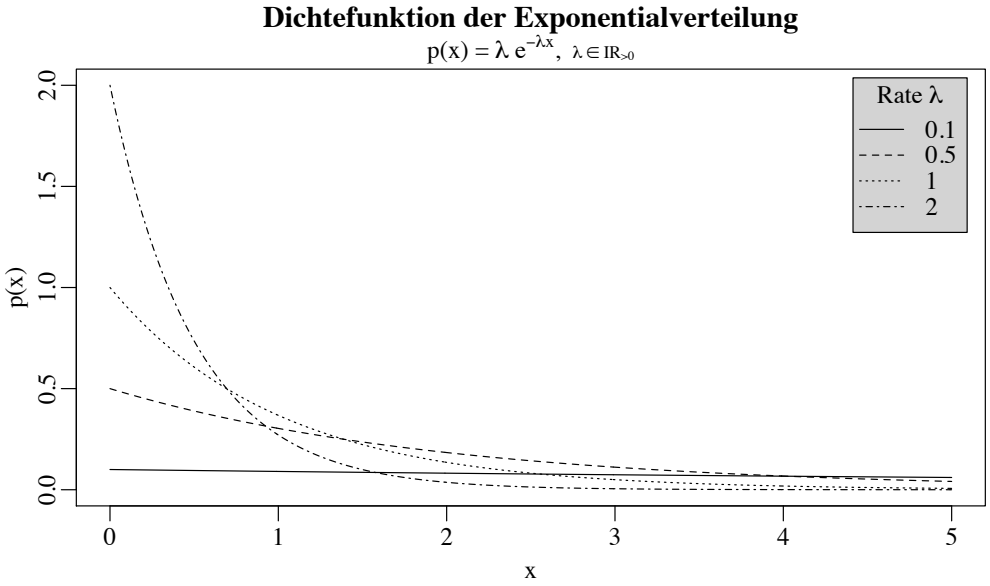


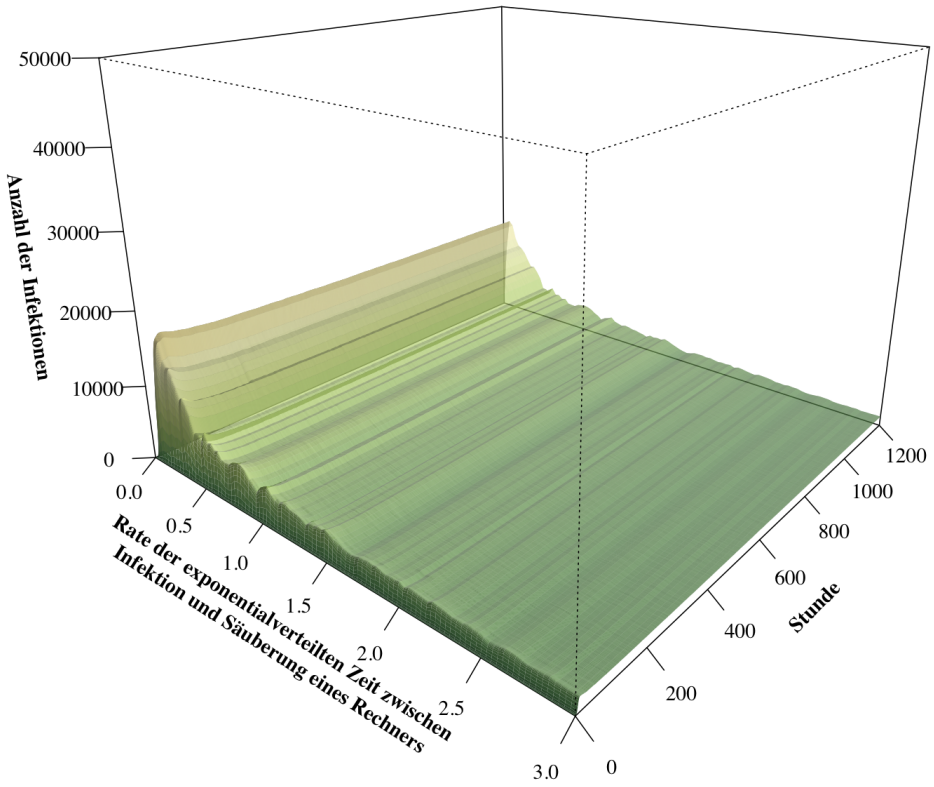
Abbildung 5.21.: Exponentialverteilung
Dies sind vier unterschiedliche Wahrscheinlichkeitsdichten, die alle einer Exponentialverteilung genügen. Die Raten λ der einzelnen Dichtefunktionen sind unterschiedlich und jeweils in der Legende angegeben.

Der Abfall der Rechnerinfektionen bei steigender Rate ist nicht linear, sondern ähnlich einem exponentiellen Abfall. In dem 3-dimensionalen Verbreitungsverlauf der Abbildung 5.22a ist dies gut zu erkennen: Bei kleineren Raten $\lambda_{cleaning} < 0{,}5$ ist dieser Abfall recht stark. Bei größeren Raten fällt die Anzahl der Rechnerinfektionen nicht mehr ganz so stark. Noch deutlicher wird dieses Verhalten in Abbildung 5.23. Diese zeigt einen Querschnitt durch den Verbreitungsverlauf nach genau 24 Stunden.

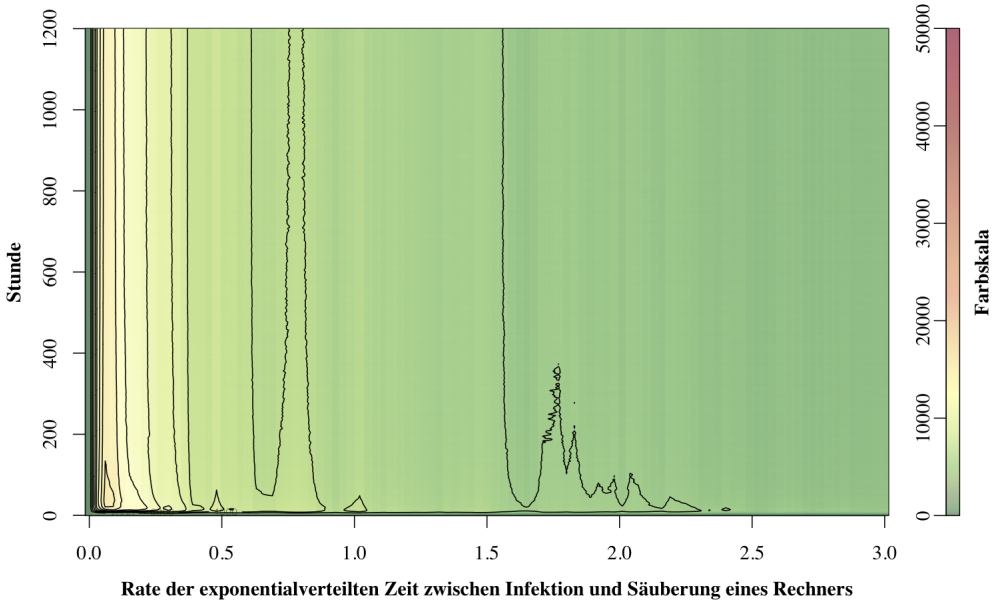
5.3.7. Wahrscheinlichkeit, dass ein Rechner immun werden kann

Im vorherigen Abschnitt haben wir den Einfluss einer Rechnerreinigung auf den Verbreitungsverlauf des E-Mail-Wurms untersucht. Dabei kann es vorkommen, dass ein Rechner während der vollen 50 Tage Simulationsdauer mehrmals infiziert und wieder bereinigt wird. In der Realität ist es nicht unwahrscheinlich, dass ein Benutzer, wenn sein Rechner immer wieder von dem gleichen Schadprogramm infiziert wird, aus diesen Vorfällen lernt. Nach einer gewissen Zeit ist er dann nicht mehr anfällig für die Social Engineering Techniken der E-Mail oder installiert sogar einen Patch, der die ausnutzbare Sicherheitslücke in seinem Browser schließt. Dies bedeutet, dass nach jeder Infektion eines Rechners mit einer gewissen Wahrscheinlichkeit eine Chance darauf besteht, dass dieser immun gegen den E-Mail-Wurm wird – sich also nicht mehr mit diesem infizieren kann. Dieses Wahrscheinlichkeit wird in der Simulation durch den Simulationsparameter p_{immune} gesteuert.

Zur Analyse des Effektes dieser Wahrscheinlichkeit auf den Verbreitungsverlauf des E-Mail-Wurms, haben wir dem Parameter p_{immune} zu Beginn die Wahrscheinlich-



(a) 3-dimensionale Darstellung des Verbreitungsverlaufs



(b) Verbreitungsverlauf als Heatmap

Abbildung 5.22.: Einfluss der Zeitspanne zwischen Rechnerinfektion und -reinigung
Die Abbildung zeigt den Einfluss der Zeitspannen, die zwischen der Infektion und der Reinigung eines Systems vergehen, auf den Verbreitungsverlauf des Wurms. Diese Zeitspannen sind exponentialverteilt und lassen sich durch die Rate der Exponentialverteilung steuern.

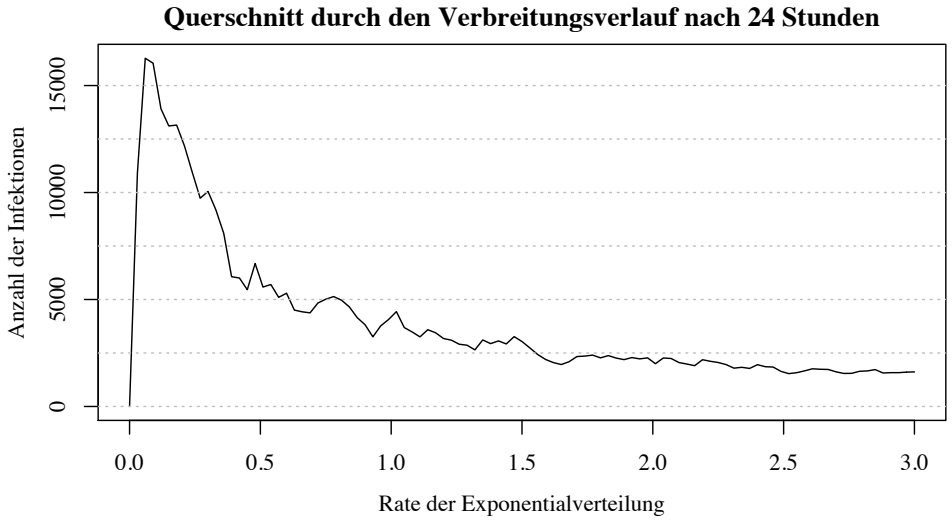


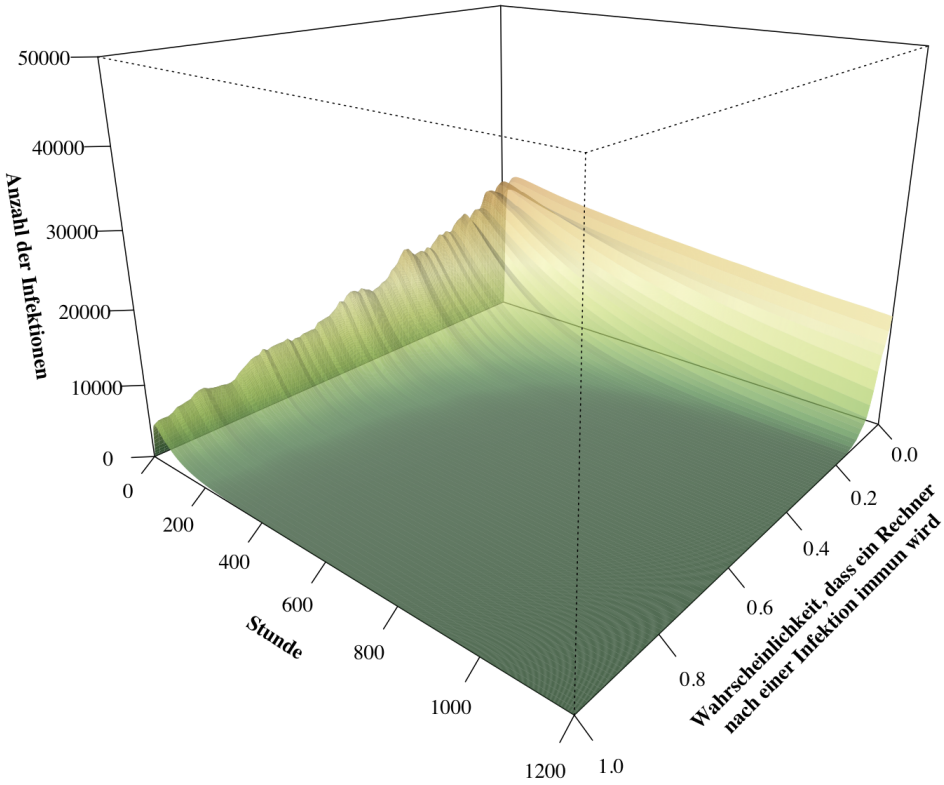
Abbildung 5.23.: Querschnitt durch den Verbreitungsverlauf nach genau einem Tag
Durch die höheren Raten der exponentialverteilten Zeitspannen zwischen einer Rechnerinfektion und -reinigung fallen diese Zeitspannen durchschnittlich geringer aus. Dies spiegelt sich in den Verbreitungsverläufen in einer geringeren Anzahl der Rechnerinfektionen wider.

keit 0 zugewiesen. Anschließend haben wir die Wahrscheinlichkeit sukzessive um den Wert $0{,}01$ erhöht, bis der Parameter den Wert 1 enthielt. Abbildung 5.24 zeigt das Ergebnis dieser Analyse.

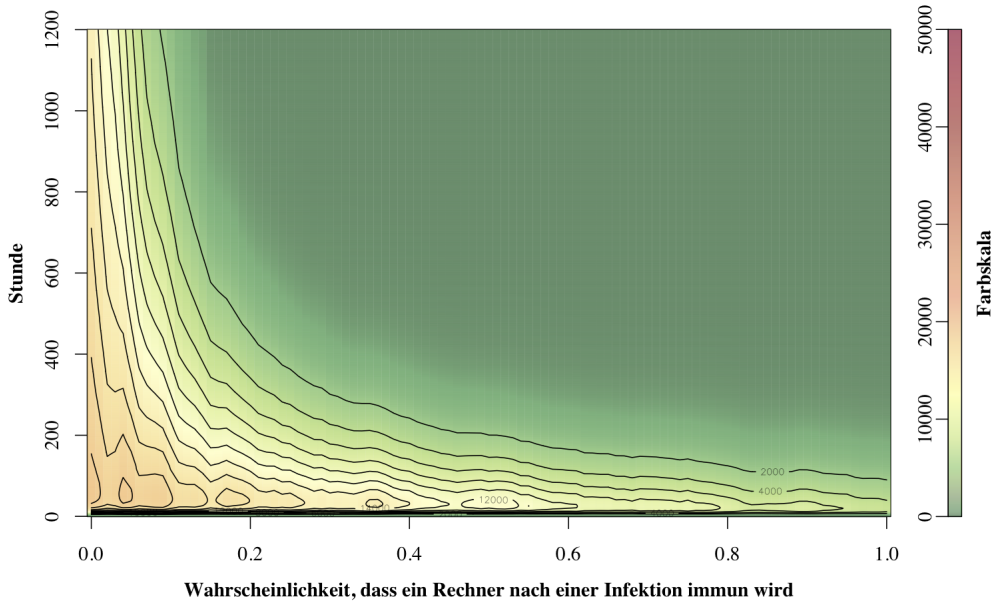
Auch hier sind im Vergleich zu den ersten 3-dimensionalen Verbreitungsverläufen aus Darstellungsgründen die Achsen wieder anders angeordnet. Charakteristisch für den Effekt dieses Parameters ist die halbkreisförmige Senke. Diese wird mit höheren Parameterwerten und mit fortlaufender Simulationsdauer immer tiefer. Das Zentrum der Senke befindet sich in der Abbildung 5.24b somit in der oberen, rechten Ecke der Heatmap. Der Ursache, weshalb die Senke mit diesen beiden Dimensionen immer ausgeprägter wird, ist folgender:

1. **Simulationsdauer**: Je länger die Simulation andauert, desto häufiger infiziert sich ein Rechner mit dem E-Mail-Wurm. Da sich nach jeder Infektion entscheidet, ob ein Rechner ab nun immun gegen den Wurm ist, steigt bei einer mehrmaligen Infektion auch die Gesamtwahrscheinlichkeit einer möglichen Immunität eines Rechners über die gesamte Simulationsdauer.
2. **Wert des Parameters**: Je größer der Wert des Parameters p_{immune} ist, desto wahrscheinlicher ist es, dass ein Rechner nach einer Infektion immun gegen den E-Mail-Wurm wird. Ist dies der Fall, kann er sich nicht mehr weiter mit dem E-Mail-Wurm infizieren und trägt somit auch nicht mehr zur dessen Verbreitung bei.

Punkt 2 dieser Begründung nimmt gleichermaßen Einfluss auf die Form des Verlaufs der Maxima der Verbreitungsverläufe – also auf die Linie, die sich bei einem zu der zeitlichen Dimension relativ frühen, orthogonalen Querschnitt durch den Verbreitungsverlauf ergibt. Man erkennt in der 3-dimensionalen Darstellung, dass die Maxima der



(a) 3-dimensionale Darstellung des Verbreitungsverlaufs



(b) Verbreitungsverlauf als Heatmap

Abbildung 5.24.: Einfluss der Wahrscheinlichkeit, dass ein Rechner immun wird.
Die Abbildung zeigt den Einfluss der Wahrscheinlichkeit, dass ein Rechner nach einer Infektion immun gegen den E-Mail-Wurm wird, auf dessen Verbreitungsverlauf.

Verbreitungsverläufe mit steigender Wahrscheinlichkeit fast linear abnehmen. Noch deutlicher wird dies in Abbildung 5.25. In dieser ist eine seitliche Ansicht des Verbreitungsverlaufs dargestellt, in der zusätzlich der Verlauf der Maxima durch eine rote Linie hervorgehoben ist. Die Wahrscheinlichkeit einer möglichen Immunität nimmt also auch direkten Einfluss auf die maximale Anzahl der infizierten Maschinen.

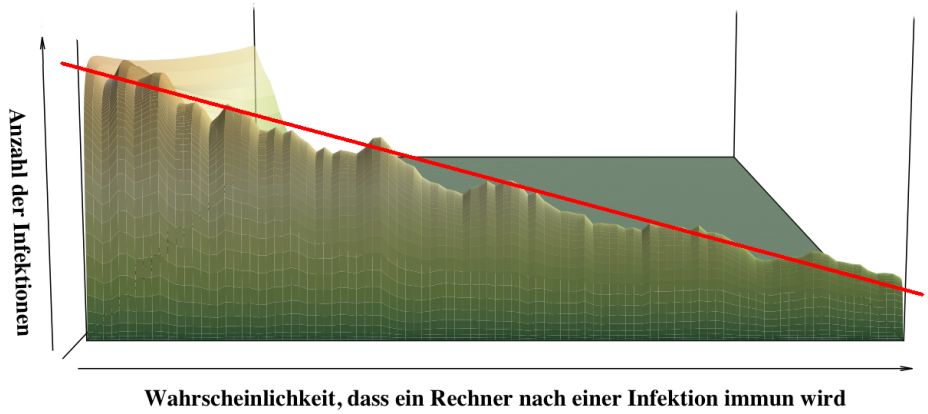


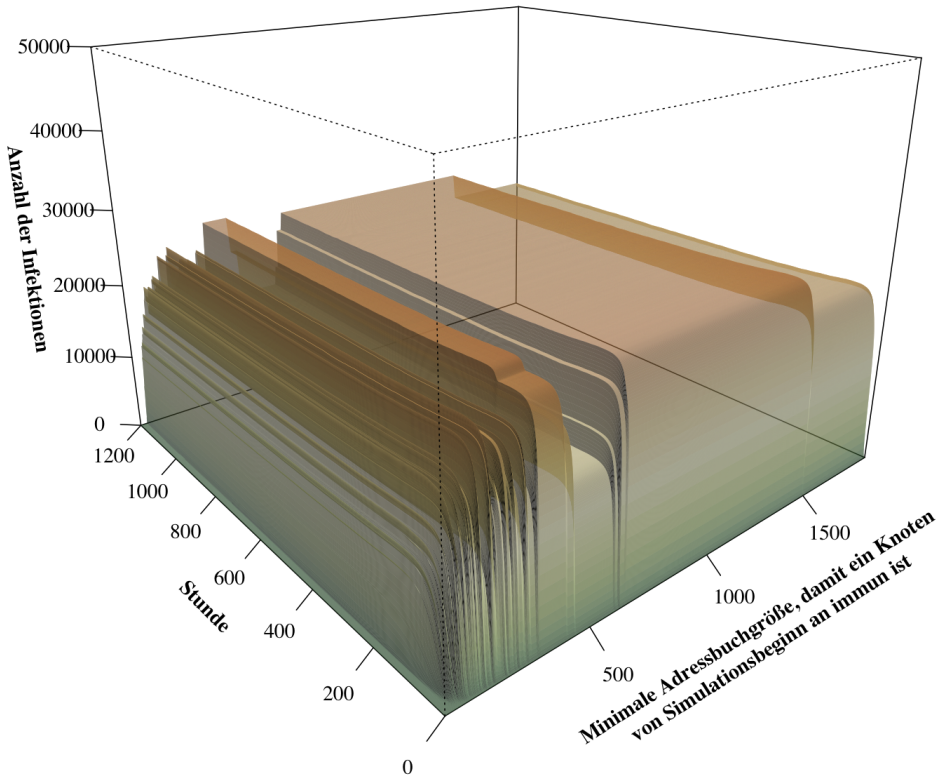
Abbildung 5.25.: Seitliche Ansicht des 3-dimensionalen Verbreitungsverlaufs
Dies ist eine seitliche Ansicht der Abbildung 5.24a. Die eingezeichnete, rote Linie verdeutlicht den fast linearen Abfall der infizierten Rechner mit ansteigender Wahrscheinlichkeit, dass ein Rechner nach einer Infektion immun wird.

5.3.8. Gezielte Immunisierung zentraler Knoten

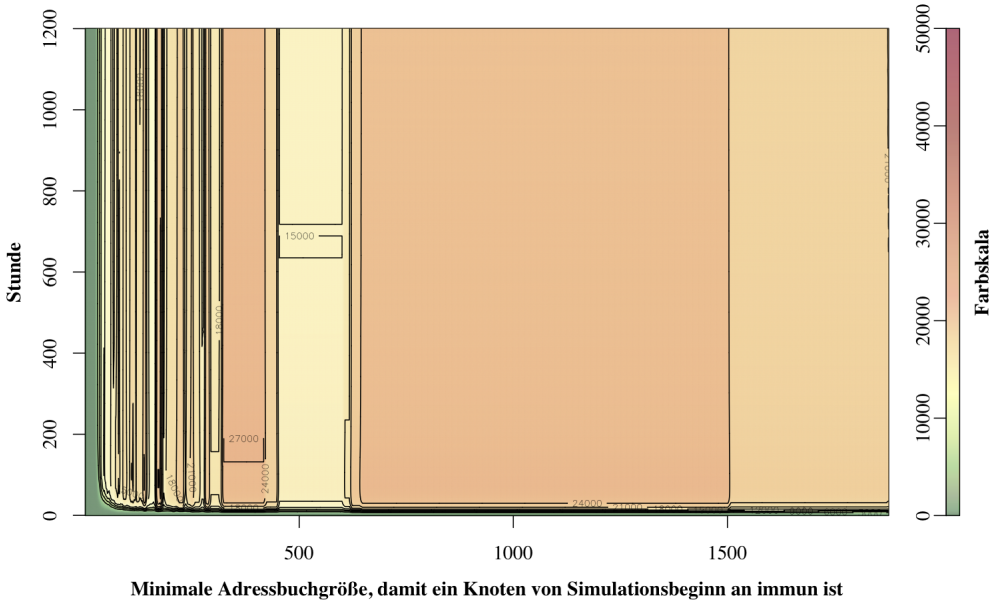
Der letzte Parameter, dessen Einfluss auf den Verbreitungsverlauf untersucht wird, ist der Simulationsparameter d_{immune}. Dieser beinhaltet eine minimale Adressbuchgröße. Alle Rechner, die mindestens genau so viele oder mehr Kontakte in ihrem Adressbuch haben als dieser Schwellwert beinhaltet, gelten bereits beim Start der Simulation als immun. Durch diesen Simulationsparameter kann also der Effekt einer gezielten Immunisierung bestimmter, zentraler Knoten untersucht werden. Das Ergebnis der Analyse befindet sich in Abbildung 5.26.

Bei der Analyse haben wir dem Parameter nacheinander folgende Werte zugewiesen: Der erste Wert ist der maximale Verzweigungsgrad des E-Mail-Netzes – also $d_{immune} = 1\,876$. Der zweite Wert entspricht dem Verzweigungsgrad des Knotens, der am zweitmeisten Nachbarknoten besitzt, usw. Der letzte Wert ist somit der kleinste Verzweigungsgrad des E-Mail-Netzes – also $d_{immune} = 1$. Dadurch ergibt sich natürlich, dass die Werte nicht kontinuierlich sind, sondern, vor allem in hohen Wertebereichen, Lücken aufweisen. Simulationen mit Parameterwerten aus diesen Lücken ergeben jedoch keine neuen Informationen, da bei diesen Belegungen die Simulationsergebnisse den Ergebnissen des nächst höheren Wertes entsprechen, der auch tatsächlich in dem E-Mail-Netz als Verzweigungsgrad vorhanden ist.

Folgendes Beispiel macht dies deutlich: Angenommen, dass in der sortierten Reihenfolge der Verzweigungsgrade die Werte 10 und 12 direkt nebeneinander stehen. Daraus folgt, dass kein Knoten mit genau 11 Nachbarknoten existiert. Die Menge



(a) 3-dimensionale Darstellung des Verbreitungsverlaufs



(b) Verbreitungsverlauf als Heatmap

Abbildung 5.26.: Einfluss einer gezielten Immunisierung
Die Abbildung zeigt den Einfluss einer gezielten Immunisierung, abhängig von der Adressbuchgröße eines Benutzers, auf den Verbreitungsverlauf des E-Mail-Wurms. Jeder Rechner, dessen Adressbuchgröße mit dem Parameterwert identisch ist oder diesen übersteigt, ist von Simulationsbeginn an immun gegen den Wurm.

der Knoten mit mindestens 11 Nachbarknoten entspricht somit genau der Menge der Knoten, die mindestens 12 Nachbarknoten haben. Aus diesem Grund haben wir die Verbreitungsverläufe, die sich bei Parameterwerten aus den beschriebenen Lücken ergeben, dadurch interpoliert, dass wir diese mit den Verbreitungsverläufen der nächst höheren, real existierenden Belegung gleichgesetzt haben. Dementsprechend weist der Verbreitungsverlauf in Abbildung 5.26 scheinbar mehrere ebene Flächen auf. Ganz eben sind diese Flächen nicht, da es leichte Schwankungen der Infektionsrate im Verlauf der Zeit gibt. Jedoch sind diese so gering, dass sie in beiden Darstellungsformen kaum bis gar nicht auffallen.

Das Diagramm der Abbildung 5.27 enthält einen Querschnitt durch den 3-dimensionalen Verbreitungsverlauf orthogonal zur zeitlichen Dimension. Dieser wird dargestellt durch die durchgezogene Linie und entspricht der Anzahl der infizierten Rechner nach genau 600 Stunden. Da bereits in dem Verbreitungsverlauf zu erkennen ist, dass eine gezielte Immunisierung in den kleineren Wertebereichen von d_{immune} einen größeren Effekt erzielt als in den großen Wertebereichen, ist die x-Achse der Abbildung logarithmisch skaliert. Die gestrichelte Linie gibt für jede Belegung des Parameters d_{immune} die Anzahl der Rechner an, die bei den entsprechenden Simulationen durch die gezielte Immunisierung nicht aktiv an der Verbreitung des E-Mail-Wurms teilnehmen.

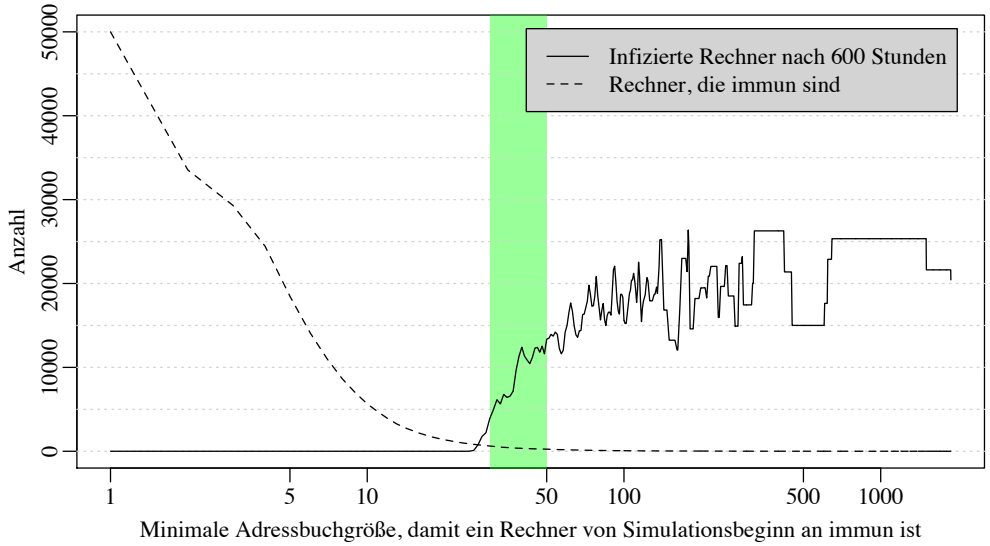


Abbildung 5.27.: Querschnitt durch den Verbreitungsverlaufs
Die durchgezogene Linie ist ein Querschnitt durch den Verbreitungsverlauf nach genau 600 Stunden. Die gestrichelte Linie gibt die kumulierte Anzahl der Rechner an, die bei entsprechender Parameterbelegung der x-Achse, nicht immun sind.

Anhand des 3-dimensionalen Verbreitungsverlaufs, aber vor allem auch anhand des Querschnitts aus Abbildung 5.27, erkennt man relativ gut, dass bei dem gegebenen E-Mail-Netz eine gezielte Immunisierung zentraler Knoten nicht bei allen Parameterwerten einen Effekt auf die Verbreitung des E-Mail-Wurms hat: Betrachtet man den Verbreitungsverlauf entlang der Parameterdimension von rechts nach links (also bei

den hohen Werten beginnend), sieht man, dass die Anzahl der infizierten Rechner nicht erkennbar mit dem Parameterwert abnimmt. Zwar gibt es immer wieder Schwankungen bei den resultierenden Verbreitungsverläufen, etwa die auffällig große Lücke um den Wert 500 herum, jedoch sind diese wohl eher auf eine statistische Streuung der Zufallswerte zurückzuführen als auf einen Effekt des untersuchten Parameters. Zumal die erwähnte Lücke lediglich aus einem einzelnen Parameterwert resultiert und nur aufgrund der bereits erklärten Interpolation künstlich auf ca. 150 Parameterwerte verbreitert wird.

Ein deutlich wahrnehmbarer Effekt setzt jedoch, die Verbreitungsverläufe immer noch von rechts nach links betrachtend, ungefähr ab einer Parameterbelegung von $d_{immune} = 50$ ein. In diesem Bereich gibt es zwar nach wie vor Schwankungen, aber die Anzahl der infizierten Rechner verringert sich wahrnehmbar mit dem Wert des Parameters. Im Wertebereich $d_{immune} \leq 25$ kann sich der E-Mail-Wurm letztendlich gar nicht mehr verbreiten, d. h. hier erzielt eine gezielte Immunisierung ihren größten Effekt.

Allerdings müssen dafür immer mehr Rechner gehärtet werden, was an der gestrichelten Linie der Abbildung 5.27 nachvollzogen werden kann. Bei zu kleinen Werten von d_{immune} kann also nicht mehr unbedingt von einer *gezielten* Immunisierung gesprochen werden. Aus diesem Grund ist in Abbildung 5.27 ein grün hinterlegter Wertebereich gekennzeichnet. In diesem Bereich ist eine gezielte Immunisierung am sinnvollsten, da der Anteil der zu härtenden Rechner recht gering ist und die Immunisierung bereits einen deutlichen Effekt aufweist. Den Wertebereich haben wir nicht mathematisch bestimmt, sondern nur grob anhand der Abbildung eingegrenzt. Ob und in welchem Ausmaß eine gezielte Immunisierung in einem realen Szenario sinnvoll bzw. rentabel ist, muss sicherlich ganz individuell entschieden werden und ist nicht Ziel dieser Arbeit.

5.4. Zusammenfassung

Im vorangegangenen Kapitel wurden verschiedene Tests und Experimente durchgeführt, die das Ziel hatten, das Verhalten von Computerbenutzern in Situationen zu erfassen und/oder zu messen, die zu einer Infektion durch Social Malcode führen können. In diesem Kapitel werden die erzielten Ergebnisse benutzt, um die Verbreitung von Social Malcode anhand einer Simulation eingehender zu untersuchen.

Für die Implementierung der Simulationen haben wir das Simulationsframework *OMNeT++* verwendet. Dieses ermöglicht die Erstellung von Simulationen in verschiedensten Anwendungsbereichen. Durch eine eigene Beschreibungssprache wird die Struktur beziehungsweise die Topologie der zu erstellenden Simulation festgelegt. Das Verhalten einer Simulation wird in C++-Code spezifiziert.

Mit Hilfe dieses Frameworks haben wir eine Simulation der Verbreitung von Social Malcode erstellt. Damit die Ergebnisse der Simulationen besser zu interpretieren sind, haben wir vor deren Präsentation zuerst die Rahmenbedingungen der Simulation abgesteckt:

- **E-Mail-Wurm**: Zur Analyse der Verbreitung von Social Malcode haben wir ein Schadprogramm simuliert, das sich über das Medium E-Mail verbreitet. Dabei

wird nach einer Infektion der befallene Rechner nach E-Mail-Adressen durchsucht. Anschließend verschickt das Schadprogramm an alle gefundenen E-Mail-Adressen eine neue E-Mail, in der sich ein Verweis auf eine Internetseite befindet. Besucht ein Benutzer diese Seite mit einem verwundbaren Browser, wird durch einen Drive-By-Download eine neue Instanz des Wurms auf den Rechner geladen und die Verbreitungsroutine beginnt auf dem neu infizierten Rechner wieder von vorn.

- **Modell-Struktur**: Neben dem simulierten Wurm an sich, nimmt auch die Struktur des simulierten Netzes Einfluss auf dessen Verbreitung. Für die vorliegende Arbeit haben wir ein E-Mail-Netz simuliert, in dem sich $50\,000$ Rechner befinden. Zur Generierung des Netzes haben wir einen Algorithmus von Holme und Kim verwendet.Dadurch entstanden in dem generierten Netz Cluster, also stark miteinander vernetzte Teilbereiche, und die Verteilung aller Verzweigungsgrade genügt einem Potenzgesetz.

- **Modell-Verhalten**: Das Modell-Verhalten haben wir anhand zweier Programmablaufpläne beschrieben. Der erste Plan ist eine Beschreibung des globalen Modell-Verhaltens, der das Zusammenspiel aller Rechner veranschaulicht. Das interne Verhalten der Knoten wurde durch den zweiten Programmablaufplan visualisiert und orientiert sich stark an der Funktionsweise des zuvor beschriebenen E-Mail-Wurms.

Bei der Beschreibung der Rahmenbedingungen wurden bereits einige Simulationsparameter der Simulation ersichtlich. Mit Hilfe dieser Parameter kann der Verlauf der Simulation gesteuert und kontrolliert werden. Jeder einzelne Parameter beschreibt einen wichtigen Faktor bei der Verbreitung von Social Malcode – etwa die Wahrscheinlichkeit, mit der Benutzer dem Verweis der E-Mail folgen, oder die Rate der Exponentialverteilung, die die Zeitspannen zwischen Infektion und Reinigung eines Rechners beschreibt. In Abschnitt 5.2.4 wurden nochmals alle Simulationsparameter übersichtlich zusammengefasst und kurz beschrieben.

Anschließend wurden die Simulationen gestartet und untersucht, welchen Einfluss die einzelnen Parameter auf die Verbreitung des E-Mail-Wurms nehmen. Dafür haben wir jeweils einen Parameter herausgenommen und diesen mit verschiedenen Werten belegt, die anderen Parameter enthielten alle einen festen Standardwert. Für jeden Wert haben wir insgesamt 40 Simulationsläufe durchgeführt und aus diesen Ergebnissen einen durchschnittlichen Verbreitungsverlauf berechnet. Anschließend haben wir die so erhaltenen Verbreitungsverläufe zusammengesetzt und dadurch eine 3-dimensionale Darstellungsform erhalten. Diese erlaubt eine schnelle, visuelle Erfassung des Einflusses der einzelnen Parameter auf die Verbreitung des Wurmes. Neben den beiden, im vorherigen Absatz als Beispiel angesprochenen Simulationsparametern, haben wir auf diese Weise ebenfalls den Effekt folgender Parameter analysiert:

- **Anzahl der Rechner, die zu Beginn der Simulation eine initiale Spam-Mail in ihrem Posteingang haben**: Je mehr Rechner zu Beginn der Wurmverbreitung eine E-Mail in ihrem Posteingang haben, desto mehr Rechner infiziert der Wurm – diese beiden Größen verhalten sich also proportional zueinander.

- **Wahrscheinlichkeit, mit der Benutzer eine neu eingetroffene E-Mail lesen**: Durch eine Erhöhrung der Wahrscheinlichkeit, mit der die Benutzer der Simulation neue E-Mails lesen, steigt die Anzahl der Infektionen rasant an.
- **Zeitspannen zwischen dem Eintreffen einer E-Mail im Posteingang und dem Lesen durch den Benutzer**: Dieser Parameter bewirkt mit steigenden Werten auch einen rasanten Anstieg der Infektionen – jedoch nicht so stark wie bei dem zuvor untersuchten Parameter. Zudem ergibt sich in einem bestimmten Wertebereich eine Senke im 3-dimensionalen Verbreitungsverlauf, da in dieser Konfiguration Rechner teilweise schneller bereinigt als infiziert werden.
- **Künstliche Verzögerung in der Zustellgeschwindigkeit einer E-Mail**: Durch eine langsamere Zustellung der E-Mails ergibt sich ebenfalls eine langsamere Verbreitung des Wurms. Die Zeit zwischen dem Ausbruch des Wurms und dem Zeitpunkt, ab dem er sich rasant verbreitet, hängt proportional mit der Verzögerung der E-Mails zusammen. Zudem ist ein Flattern der Verläufe bei langen Verzögerungen zu beobachten.
- **Wahrscheinlichkeit, mit der ein Rechner nach einer Wurminfektion immun gegen diesen werden kann**: Der Effekt dieses Parameters macht sich durch eine Senke im 3-dimensionalen Verbreitungsverlauf bemerkbar. Mit einer höheren Wahrscheinlichkeit sinkt natürlich die Anzahl der Rechner, die aktiv an der Wurmverbreitung teilnehmen. Bei einem festen Wert des Parameters, sinkt diese Anzahl ebenfalls mit dem Fortlauf der Simulation, da bei mehrmaliger Infektion eines Rechners immer wieder neu entschieden wird, ob dieser nun immun gegen den Wurm ist.
- **Gezielte Härtung oder Immunisierung einzelner, zentraler Rechner des Netzes**: Das Ergebnis der Untersuchung ist, dass eine gezielte Immunisierung zentraler Knoten des E-Mail-Netzes nur in einem bestimmten Umfang Sinn macht. Härtet man zu wenige zentrale Knoten, verbreitet sich der Wurm dennoch relativ gut. Härtet man viele Knoten, dämmt dies zwar sehr gut die Verbreitung des Wurms ein, jedoch ist der entsprechende Aufwand, den man durch die vielen Härtungen betreiben muss, extrem hoch. Davon abgesehen kann bei zu vielen gehärteten Rechnern nicht mehr von einer *gezielten* Immunisierung gesprochen werden.

KAPITEL VI

Diskussion

In diesem Kapitel der Arbeit werden die erzielten Ergebnisse kritisch diskutiert. Dazu stellen wir in Abschnitt 6.1 Arbeiten und Veröffentlichungen vor, die einen inhaltlichen Zusammenhang zu der hier vorgestellten Arbeit oder Teilen dieser Arbeit besitzen. Neben einer kurzen Zusammenfassung dieser Arbeiten wird ebenfalls herausgestellt, inwieweit sich diese von der vorliegenden Arbeit unterscheiden.

Ein spezieller Vergleich der in dieser Arbeit generierten Verbreitungsverläufe mit Verbreitungsmodellen und -metriken anderer Arbeiten wird in Abschnitt 6.2 gezogen. Hier werden Ähnlichkeiten zu den Modellen anderer Forschungsarbeiten herausgestellt. Aber auch die Unterschiede zwischen den Ergebnissen werden beleuchtet und erklärt.

Einige der Simulationsparameter bieten die Möglichkeit, Maßnahmen gegen die Verbreitung des E-Mail-Wurms umzusetzen. In Kapitel 5 wurden jeweils die Effekte einzelner Simulationsparameter auf die Verbreitung des Wurms untersucht. Abschnitt 6.3 hingegen zeigt den Effekt einer Kombination verschiedener Gegenmaßnahmen.

In Abschnitt 6.4 werden Limitierungen der vorliegenden Arbeit diskutiert. Dabei handelt es sich um Einschränkungen, die im Rahmen der Arbeit nicht zu lösen oder zu umgehen waren. Teilweise entstanden diese Einschränkungen durch einen begrenzten zeitlichen Rahmen, hatten jedoch auch rechtliche Aspekte als Ursache. Beispielsweise konnten wir mittels einem von uns geschriebenen Skript nur einen Teil der Postfächer der Universität Mannheim analysieren. Dieses durfte nur auf die Postfächer derjenigen Benutzer angewandt werden, die uns dafür ihr Einverständnis gegeben hatten. Auch die Spam-Experimente durften nur nach Beauftragung durch einen Kunden durchgeführt werden. Diese Einschränkungen wirken sich auf die Datenbasis aus, die wir zur Analyse des menschlichen Verhaltens benutzt haben. Jedoch bieten Einschränkungen auch die Grundlage für Verbesserungsansätze und Erweiterungen der Arbeit. Neben einer Beschreibung der Einschränkungen werden in dem Abschnitt ebenfalls Ideen und Anregungen für weiterführende Arbeiten vorgestellt.

Abgeschlossen wird das Kapitel durch eine Zusammenfassung, die die Kernaussagen des Kapitels in kurzer und kompakter Form wiedergibt.

6.1. Verwandte Arbeiten

In diesem Abschnitt präsentieren wir andere, wissenschaftliche Arbeiten, die einen inhaltlichen Zusammenhang zu der vorliegenden Arbeit aufweisen. Er ist in mehrere andere Abschnitte unterteilt, die jeweils einen inhaltlich abgeschlossenen Teil der vorliegenden Arbeit umfassen. In Abschnitt 6.1.1 werden beispielsweise Arbeiten präsentiert, die sich inhaltlich ebenfalls damit beschäftigen, das Verhalten von Benutzern zu erfassen und zu analysieren. Des Weiteren werden in Abschnitt 6.1.2 Arbeiten vorgestellt, die sich mit der Verbreitung von Schadprogrammen allgemein beschäftigen. Arbeiten, in deren Fokus speziell benutzerabhängige Schadprogramme und deren Analyse stehen, werden in Abschnitt 6.1.3 erörtert.

6.1.1. Benutzerverhalten

In ihrer Arbeit *Spamalytics: An Empirical Analysis of Spam Marketing Conversion* [KKL+08] haben Kanich et al. die Reaktion von Empfängern auf den Erhalt von Spam-Mails untersucht. Im Fokus ihrer Arbeit standen die Rücklaufraten eines Botnetzes, das unter anderem Spam-Mails mit Verweisen auf Internetseiten verschickt. Auf diesen Seiten wurden verschiedene, teils dubiose Artikel beworben und zum Kauf angeboten. Zu diesem Zweck haben die Autoren ein existierendes Botnetz, eine Instanz des Sturm-Wurms, infiltriert und so modifiziert, dass ein Teil der Spam-Mails nicht mehr auf die ursprünglichen Internetseiten verweist, sondern auf eigens von den Autoren eingerichtete Internetseiten. Auf diesen betrieben sie eine optische Kopie der ursprünglichen Internetseiten – jedoch fehlte die Funktionalität zum Infizieren neuer Opfer und es wurden auch keine sensiblen Daten der Seitenbesucher abgespeichert.

Zwar konnten die Autoren durch die Modifikation eines großen Botnetzes über 17 000 Besuche ihrer nachgebauten Seiten registrieren, jedoch wurden dazu über 469 Millionen Spam-Mails von den modifizierten Spam-Bots versendet. Wie sie auch selbst in ihrer Arbeit anmerken, bewegen sie sich durch dieses Vorgehen in einer rechtlichen und ethischen Grauzone. Die von uns in Abschnitt 4.2 beschriebenen Spam-Experimente sind hingegen alle rechtlich abgesichert.

Ziel der Arbeit *Social Phishing* [JJJM07] von Jagatic et al. war es, die Erfolgsraten eines Phishing-Angriffs zu messen. In einem ersten Schritt haben die Autoren soziale Netzwerke geparst und dadurch eine Datenbank erstellt, die die Beziehungen zwischen den Profilen widerspiegelt. Da lediglich Personen, die in einem Zusammenhang mit der Universität Indiana stehen, das Ziel des Phishing-Angriffs sein sollten, glichen sie die gesammelten E-Mail-Adressen mit einem Adressbuch der Universität ab und schränkten so den Personenkreis ein. Anschließend verschickten sie E-Mails, in denen die Empfänger aufgefordert wurden, ihre universitätsinternen Zugangsdaten zu ändern. Ebenso wie bei den von uns durchgeführten Phishing-Experimenten (siehe Abschnitt 4.2.4), erstellten sie eine Phishing-Seite, die zur angeblichen Passwortänderung verwendet werden sollte. Die Autoren führten ihr Experiment in zwei Varianten durch: Zum einen griffen sie auf die vorher erstellte Beziehungsdatenbank zurück und verschickten ein Teil der E-Mails im Namen eines Freundes (Variante 1) und zum anderen verschickten sie einen Teil der E-Mails mit der Identität eines komplett unbekannten Absenders (Variante 2).

Die durch ihr Experiment gesammelten Daten werteten die Autoren auf verschiedene Arten aus. Die von ihnen registrierte Erfolgsrate der ersten Variante ist mit 72 % wesentlich höher als die von uns gemessene Rate von ca. 20 %. Jedoch ist laut eigenen Angaben auch die von den Autoren gemessene Erfolgsrate der zweiten Variante um einen ähnlichen Faktor höher als die Beobachtungen, die das Marktforschungsunternehmen *Gartner, Inc.*[1] gemacht hat. Die zeitliche Verteilung der Reaktionszeiten ist mit unseren Ergebnissen vergleichbar. Ein weiteres Ergebnis ihrer Arbeit war unter anderem, dass die Phishing-Angriffe bei Studenten, die noch nicht lange studieren, erfolgreicher waren als bei Studenten eines höheren Semesters. Zudem beobachteten sie, dass sich die Erfolgswahrscheinlichkeit des Angriffs erhöht, wenn die Phishing-Mail von einem Absender stammt, der jeweils dem anderen Geschlecht angehört.

6.1.2. Verbreitung von Schadprogrammen

In dem 2002 veröffentlichten Paper *Code-Red: a case study on the spread and victims of an Intemet worm* [MSC02] beschreiben Moore, Shannon und Brown die Computerwürmer Code Red und Code Red 2. Nach einer ausführlichen Beschreibung der Verbreitungsroutinen beider Würmer erläutern die Autoren im methodischen Teil, wie sie die globalen Infektionsraten bestimmen konnten: Sie betrachteten einen Rechner als infiziert, falls dieser mindestens zwei TCP-SYN Pakete zu nicht existierenden Rechnern ihrer Netze schickte. Auf diese Weise konnten sie innerhalb der ersten 24 Stunden nach dem Start der Wurmepidemie bereits über 359 000 unterschiedliche, befallene IP-Adressen registrieren.

Die Arbeit konzentriert sich ausschließlich auf die beiden erwähnten Computerwürmer, die sich vollständig autonom verbreiten. Der Hauptteil der Arbeit besteht aus einer Präsentation der Analyseergebnisse der gesammelten Daten. Diese ist recht ausführlich und durch viele Diagramme sehr anschaulich aufbereitet.

Eine weitere Arbeit aus dem Jahr 2002, die speziell den Ausbruch von Code Red thematisiert, ist das Paper *Code Red Worm Propagation Modeling and Analysis* [ZGT02] von Zou, Gong und Towsley. Im Gegensatz zur Arbeit von Moore et al., die überwiegend aufgezeichnete Daten präsentiert und auswertet, liegt der Fokus dieser Arbeit eher auf einer Modellierung der Ausbreitung. Die Autoren führen ein neues Modell ein, das sie *Two-Factor Worm Model* nennen. Dieses erweitert klassische, epidemiologische Modelle um zwei neue Faktoren: Das Modell berücksichtigt zum einen mögliche Gegenmaßnahmen von Computerbenutzern und/oder der ISP, zum anderen eine Verringerung der Infektionsrate im Verlauf der Epidemie, die durch die rasante Verbreitung des Wurms und den daraus entstehenden Router-Problemen bedingt ist.

Die mit Hilfe ihres Modells erstellten Verbreitungsverläufe erreichen eine recht genaue Annäherung an tatsächlich gemessene Infektionszahlen von Code Red. Allerdings wird die Genauigkeit laut den Autoren dadurch erreicht, dass sie die verwendeten Modellparameter im Nachhinein passend gewählt haben. Methoden oder Techniken, um diese Parameter bereits in einem frühen Anfangsstadium der Epidemie passend zu bestimmen, liefern die Autoren nicht. Im Gegensatz zu den von uns durchgeführten Simulationen, bei denen ein kompromittierter Rechner nur Kontakte aus dem Adress-

[1] Internet: *http://www.gartner.com/*

buch oder Posteingang infizieren kann, verwenden Zou et al. ein homogenes Netz. Bei diesem kann, ausgehend von einem Knoten, jeder andere, beliebige Knoten infiziert werden.

Ebenfalls mit dem Thema Verbreitung von Computerwürmer beschäftigen sich Staniford, Paxson und Weaver in ihrem Paper *How to 0wn the Internet in Your Spare Time* [SPW02]. Neben den Computerwürmern *Code Red* und *Code Red 2* gehen die Autoren zusätzlich noch auf den Computerwurm *Nimda* ein. Das Hauptaugenmerk des Paper liegt aber auf der Entwicklung und Präsentation neuer Verbreitungsstrategien, die einem Computerwurm eine noch schnellere Verbreitung erlauben. In diesem Zusammenhang diskutieren und evaluieren die Autoren unter anderem folgende intelligente Techniken zur Generierung von IP-Adressen möglicher, neuer Opfer:

- Das *hit-list scanning*, bei dem der Autor des Schadprogramms diesem eine sogenannte Hit-List mitgibt. Nach einer Infektion versucht der Wurm zuerst die IP-Adressen dieser Liste zu infizieren. Ist er bei einer der Adressen erfolgreich, halbiert er die Liste, übergibt der neuen Wurminstanz eine Hälfte und prüft selbst nur noch die IP-Adressen der anderen Hälfte. Die Hit-List sollte demnach aus einer Reihe von IP-Adressen bestehen, die für eine Infektion sehr anfällig sind – etwa Rechner mit einem bestimmten Betriebssystem. Ist die Liste abgearbeitet, so werden IP-Adressen anderweitig generiert, beispielsweise zufällig. Auf diese Weise ist die anfängliche Infektionsrate sehr hoch.

- Das *permutation scanning*, bei dem nur eine IP-Adresse als zufälliger Startpunkt gewählt wird. Es werden dann sequentiell alle nachfolgenden IP-Adressen geprüft, bis ein Rechner gefunden wird, der bereits mit dem Wurm infiziert ist. Anschließend wird ein neue, zufällige IP-Adresse als Startpunkt generiert. Auf diese Weise werden weniger Adressen mehrmals geprüft als bei einer rein zufälligen Generierung der IP-Adressen.

- Die Verwendung von *Internet-sized hit-lists*, bei der der Wurm initial ebenfalls eine Hit-List mitbekommt. In diesem Fall besteht diese aber aus allen Rechnern des Internets, die den verwundbaren Dienst anbieten. Die erste Instanz des Wurms teilt diese Liste in n Teile und übergibt diese den n neuen Wurminstanzen. Diese verfahren auf die gleiche Weise, so dass innerhalb von nur wenigen Sekunden fast alle Rechner mit dem verwundbaren Dienst infiziert sind. Nachteil dieser Methode ist jedoch, dass die Hit-List zu Beginn der Epidemie sehr groß ist und somit das Risiko der Erkennung ebenfalls.

Des Weiteren wird in dem Paper eine Stealth-Technik vorgestellt, die einen Computerwurm wesentlich transparenter macht und dessen Auffindbarkeit erschwert. Dabei werden nur diejenigen Rechner infiziert, zu denen eine normalerweise legitime Verbindung von einer bereits infizierten Anwendung aufgebaut wird. Dadurch verbreitet sich ein Wurm zwar langsamer, aber es entstehen auch keine speziellen Muster im Netzverkehr, über die der Wurm identifiziert werden könnte.

Gerade aufgrund der neu erarbeiteten Verbreitungsmethoden und der vorgestellten Stealth-Technik erhält das Paper einen sehr offensiven Charakter. Dadurch wird die Arbeit mit der Kritik konfrontiert, dass diese von einigen Lesern als Bastelanleitung eines *Superwurms* angesehen werden könnte. Dem wirken die Autoren entgegen, in-

dem sie in dem letzten Kapitel die Einführung eines *Cyber-Center for Disease Control* diskutieren – einer Behörde, speziell zur Bekämpfung von Computerwürmern.

Ein weiteres Modell zur Modellierung der Verbreitung eines Computerwurms wird in der Arbeit *Modeling the Spread of Active Worms* [CGK03] von Chen, Gao und Kwiat vorgestellt. Mit Hilfe des *Analytical Active Worm Propagation* (kurz AAWP) Modells lässt sich die Verbreitung eines Computerwurms untersuchen, der IP-Adressen neuer Angriffsziele zufällig generiert. Im Gegensatz zu klassischen, epidemiologischen Modellen arbeitet dieses Modell mit einer diskreten Zeiteinteilung. Dadurch versprechen sich die Autoren eine genauere Approximation der Verbreitung, da ein Rechner erst dann neue Rechner infizieren kann, nachdem seine eigene Kompromittierung vollständig abgeschlossen ist. Weitere Unterschiede zwischen epidemiologischen Modellen und dem AAWP-Modell sind, dass bei dem AAWP-Modell eine Immunisierung der Rechner betrachtet wird und dass ein Rechner von mehreren Rechnern gleichzeitig angegriffen werden kann.

Des Weiteren nutzen die Autoren ihr Modell, um zu überprüfen, wie viele Sensoren benötigt werden, um eine Wurmepidemie zu bemerken, und mit wie vielen Sensoren eine Wurmepidemie annähernd genau modelliert werden kann. Abschließend stellen die Autoren das LAAWP-Modell vor. Dieses ist eine Erweiterung des AAWP-Modells, so dass auch Würmer modelliert werden können, die bevorzugt Rechner des lokalen Subnet befallen.

6.1.3. Benutzerabhängige Schadprogrammen

Eine der ersten Arbeiten, in der die Verbreitung von Viren durch epidemiologische Modelle beschrieben wird, ist das 1991 veröffentlichte Paper *Directed-Graph Epidemiological Models of Computer Viruses* [KW91] von Kephart und White. Aufgrund der Ähnlichkeit zwischen biologischen Viren und ihren digitalen Gegenstücken, haben sie die mathematischen Modelle der Epidemiologie adaptiert, um die Verbreitung von Computerviren zu untersuchen. Als mathematisches Modell benutzen sie eine Differentialgleichung, die die zeitliche Entwicklung einer Infektionsrate beschreibt.

Die Autoren untersuchen ausschließlich die Verbreitung von Computerviren. Da sich Viren definitionsgemäß nicht autonom verbreiten, handelt es sich bei der Arbeit von Kephart und White also auch um eine Analyse von Social Malcode. Die untersuchten Dateiviren können nur dann ein neues System befallen. wenn sie händisch von einem Computerbenutzer zum nächsten weitergegeben werden. Im Gegensatz zur vorliegenden Arbeit benutzen die Autoren deshalb bei ihrer Modellierung einen gerichteten Graphen. Angesichts des Alters der Arbeit ist auch die Anzahl der Gesamtpopulation bei ihren Analysen wesentlich kleiner: Kephart und White betrachten 100 Systeme bei der Erstellung der Verbreitungsverläufe. Weitere Unterschiede bestehen darin, dass sie ein homogenes Netz zugrunde legen, in dem jeder Knoten jeden anderen, beliebigen Knoten infizieren kann, und dass sie keine mögliche Immunisierung der Knoten in ihre Arbeit einbeziehen.

Die relevante Forschung auf diesem Themengebiet ist in dem Buch *Epidemic Modelling: An Introduction* [DGG01] von Daley zusammengefasst. In dem Paper *Propagation of active worms: A survey* [XFZ09] von Xiang, Fan und Zhu befindet sich zudem eine Zusammenfassung und Gegenüberstellung der bisher vorgestellten Mo-

delle. Die Modelle, die zur Analyse von autonomen Computerwürmern und deren Verbreitung verwendet werden können, sind ebenfalls in der Gegenüberstellung von Xiang et al. enthalten.

Ein immer häufiger benutzter Angriffsvektor zur Verbreitung von Schadprogrammen sind Drive-By-Downloads. Dabei reicht es zur Infektion eines Computers aus, wenn ein Benutzer lediglich eine präparierte Internetseite in einem dafür anfälligen Browser aufruft. Provos et al. haben genau diesen Angriffsvektor in ihrem Paper *The Ghost In The Browser: Analysis of Web-based Malware* [PMM+07] genauer untersucht. Sie identifizieren genau vier Ursachen für eine mögliche Infektion: Sicherheitslücken eines Webservers (der nach einer Kompromittierung beliebig von einem Angreifer manipuliert werden kann), ungenügende Filterung von Benutzer-erstellten Inhalten (die z. B. einen persistenten Cross-Site-Scripting-Angriff [DLFMT04] ermöglichen), externe Werbebanner (auf deren Auswahl ein Seitenbetreiber nur geringen Einfluss hat) und Widgets von Drittanbietern (z. B. ein Besucherzähler). Anschließend wird anhand eines Beispiels erklärt, wie Schadprogramme Schwachstellen eines Browsers ausnutzen können. Abgeschlossen wird die Arbeit durch verschiedene Trends und Statistiken, etwa der Verbreitung von Schadprogrammen über verschiedene Internetseiten. Ein Schadprogramm war beispielsweise über 412 verschiedene Top-Level-Domains zu erreichen.

Obwohl der Fokus der Arbeit auf Schadprogrammen liegt, deren Verbreitung von einer menschlichen Interaktion abhängt, findet diese Tatsache kaum Beachtung. Lediglich in einem kurzen Abschnitt wird erwähnt, dass Angreifer auf Social Engineering zurückgreifen müssen, falls kein Drive-By-Download möglich sein sollte. Dass aber selbst bei einem möglichen Drive-By-Download der Benutzer erst dazu gebracht werden muss, die entsprechende Internetseite zu besuchen, wird ebenfalls nicht thematisiert.

Dagegen findet das Verhalten der Benutzer in dem Paper *Email Worm Modeling and Defense* [ZTG04] von Zou, Towsley und Gong einen wesentlichen höheren Stellenwert. Die Autoren haben die Verbreitung eines E-Mail-Wurms simuliert. Dafür haben sie ein mathematisches Modell aufgestellt, das sich über Parameter steuern lässt. Die Simulation haben sie für drei verschiedene Netztopologien durchgeführt und die so berechneten Verbreitungsverläufe gegenübergestellt. Als Netztopologien verwendeten sie eine *Power-Law-Topologie*, eine *Small-World-Topologie* und einen Zufallsgraphen. Abschließend untersuchten sie den Effekt einer Immunisierung bestimmter Knoten auf die daraus resultierenden Verbreitungsverläufe. Als Ergebnis stellten die Autoren fest, dass sich ein E-Mail-Wurm in einer Power-Law-Topologie wesentlich schneller verbreitet als in den beiden anderen Topologien. Genauso hat eine selektive Immunisierung in dieser Topologie die größten Auswirkungen (im Sinne einer langsameren Verbreitung) auf die Ausbreitung des Wurms.

Im Gegensatz zu der vorliegenden Arbeit konzentrieren sich die Autoren des Paper stärker auf die Netztopologien und deren Auswirkungen auf die Verbreitung des simulierten E-Mail-Wurms. Der grobe Aufbau des Paper ist dem Aufbau der vorliegenden Arbeit recht ähnlich. Der größte Unterschied ist jedoch, dass Zou, Towsley und Gong den Parametern ihrer Simulation feste Werte zuweisen, ohne diese Wertewahl zu begründen.

Wie bereits in Abschnitt 2.2.4 erklärt, benötigen auch die meisten Schadprogramme, die sich per Instant Messenger verbreiten, eine Benutzerinteraktion. Mit diesem Thema beschäftigt sich das Paper *On Instant Messaging Worms, Analysis and Countermeasures* [MvO05] von Mannan und van Oorschot. Nach einer Vorstellung ausgewählter Würmer beschreiben die Autoren verschiedene Wege der Verbreitung dieser Würmer. Ebenfalls fassen sie bereits existierende Methoden und Techniken zusammen, die die Verbreitung von IM-Würmern verhindern bzw. einschränken sollen: temporäres Abschalten der IM-Server, temporäres Trennen der Benutzer mit der größten Freundesliste vom IM-Netz und eine von Williamson et al. vorgeschlagene Methode [WPB04], die die Nachrichtenrate pro Benutzer einschränkt. Selbst stellen die Autoren zwei, ihrer Meinung nach, bessere Methoden vor: Zum einen eine Weiterentwicklung der Methode von Williamson et al., so dass nur noch der Versand von Nachrichten mit einem Verweis oder einem Dateitransfer limitiert ist, und zum anderen das Lösen eines *CAPTCHA* (Completely Automated Public Turing test to tell Computers and Humans Apart) beim Versand von Nachrichten, die einen Verweis enthalten oder einen Dateitransfer initiieren. Abschließend werden in dem Paper einige Kennzahlen aufgelistet, die die Autoren aus der Beobachtung eines Chat-Systems innerhalb eines Zeitraumes von 3,5 Jahren gesammelt haben.

Das Paper stellt einige sehr spezielle Instant Messenger Würmer und deren Verbreitung vor. Auch Methoden zur Verlangsamung und Einschränkung der Verbreitung werden vorgestellt. Jedoch wird weder die Verbreitung der Würmer quantifiziert, noch werden die Methoden der Eindämmung evaluiert, so dass dieses Paper lediglich eine Vorstellung der Schadprogramme und theoretischer Gegenmaßnahmen ist. Auch auf den Aspekt des Social Engineering wird nicht eingegangen.

Dieser Aspekt stellt jedoch in dem Artikel *An overview of social engineering malware: Trends, tactics, and implications* [ACS10] von Abraham und Chengalur-Smith ein zentrales Thema dar. Die Autoren möchten in diesem Artikel Trends und Methoden von „Social Engineering Malware“ aufzeigen. Nach einer kurzen, motivierenden Einführung in das Thema stellen die Autoren ein definitionsähnliches Modell auf, das aus vier Schritten besteht, die normalerweise von den von ihnen betrachteten Schadprogrammen durchgeführt werden: (1) Einen Benutzer zur Ausführung des Schadprogramms verleiten, (2) bestehende Antiviren-Lösungen umgehen/deaktivieren, (3) das Ausführen der Schadfunktion und (4) sich selbst weiterverbreiten. Anschließend präsentieren die Autoren die gängigsten Angriffsvektoren, psychologische Aspekte für den Erfolg der Schadprogramme und stellen deren Schadroutinen vor. Abgeschlossen wird der Artikel durch ein Kapitel, in dem die Autoren die an einem fiktiven Befall beteiligten Parteien/Personen benennen und besonders deren soziale Verpflichtung des gemeinsamen Handelns hervorheben.

Vor allem Kapitel 2 der vorliegenden Dissertation weist inhaltliche Überschneidungen mit dem gerade beschriebenen Artikel auf. Auch bieten die Autoren des Artikels eine definitionsähnliche Beschreibung der Schadprogramme, die sie selbst als *Social Engineering Malware* bezeichnen. Dieser Definitionsansatz ähnelt einem Teil des Kapitels 3 der vorliegenden Dissertation, ist jedoch wesentlich kürzer und weniger präzise ausgearbeitet. Den in dem Artikel vorgestellten Schadroutinen und Ausführungsschritten nach einer Infektion kommen in der vorliegenden Dissertation kaum, bis fast gar keine, Bedeutung zu. In dem Artikel sind diese zwar anschaulich beschrieben,

jedoch nicht charakteristisch für diese Klasse von Schadprogrammen. Vielmehr sind diese allgemeingültig für Schadprogramme. Das gleiche gilt für die Beschreibung der beteiligten Personen und Parteien: Auch hier ist das beschriebene Szenario nicht ausschließlich charakteristisch für Schadprogramme, in deren Verbreitungsprozess eine Benutzerinteraktion erforderlich ist, sondern allgemeingültig für viele verschiedene Arten von Schadprogrammen.

6.2. Vergleich mit bestehender Forschung

Nachdem wir einen Überblick über die verwandten Arbeiten gegeben haben, vergleichen wir deren Ergebnisse mit den Ergebnissen unserer Arbeit. Wir stellen Parallelen zwischen den fremden und den eigenen Ergebnissen vor und zeigen, inwieweit sich die Ergebnisse unterscheiden. Zudem stellen wir die Verbreitung eines autonomen Computerwurms der Verbreitung des von uns simulierten E-Mail-Wurms gegenüber. Anhand der Unterschiede zwischen den Verbreitungsverläufen, beleuchten wir dabei die speziellen Charakteristika von Social Malcode.

6.2.1. Phasen der Verbreitung

Die Verbreitungsverläufe des Abschnitts 5.3 weisen durchgängig eine recht ähnliche Grundform auf. Aufgrund der unterschiedlichen Parameterbelegungen entstehen zwar leichte Schwankungen bezüglich der Ausprägung dieser Grundform, jedoch lassen sich stets drei Phasen erkennen. Zur Verdeutlichung der Grundform, ist Abbildung 6.1 ein Verbreitungsverlauf des E-Wurms dargestellt. Bei der entsprechenden Simulation wurden die Simulationsparameter mit ihren Standardwerten belegt, wie sie in Tabelle 5.1 angegeben sind.

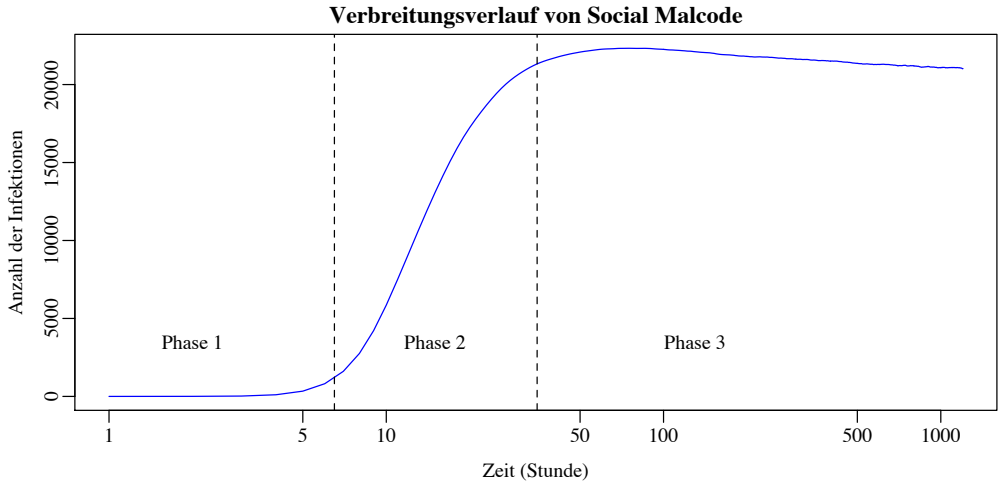


Abbildung 6.1.: Phasen der Verbreitung von Social Malcode
Der dargestellte Verbreitungsverlauf resultiert aus einer Simulation, bei der alle Parameter mit ihren Standardwerten belegt sind. Der komplette Verlauf lässt sich grob in drei Phasen einteilen: Startphase, Wachstumsphase und Kontinuitätsphase.

Die drei Phasen sind in der Abbildung durch die beiden vertikalen, gestrichelten Linien voneinander abgegrenzt. Die x-Achse der Grafik ist logarithmisch skaliert, damit man die ersten beiden Phasen, die im Vergleich zur dritten Phase recht kurz sind, besser erkennen kann. Die Phasen entsprechen drei „Lebensabschnitten“ des Wurms und lassen sich folgendermaßen interpretieren:

1. **Startphase**: Zu Beginn der Wurmepidemie ist lediglich ein kleiner Anteil der Rechner durch den E-Mail-Wurm befallen. Folglich werden in dieser Phase dementsprechend wenig E-Mails verschickt, so dass sich der Wurm nur langsam verbreiten kann. Daraus resultierend zeichnet sich die Startphase durch einen sehr flachen und nur langsam ansteigenden Verbreitungsverlauf aus.
2. **Wachstumsphase**: Mit der Zeit werden immer mehr Rechner von dem E-Mail-Wurm befallen und entsprechend steigt auch die Anzahl der E-Mails, die in den Posteingängen der Benutzer eintreffen. Da die Simulationsparameter während der Simulation konstant sind, ergibt sich somit auch eine steigende Anzahl an Benutzern, die eine E-Mail lesen und dem Verweise folgen. Dies wiederum verursacht, dass noch mehr E-Mails verschickt werden. Es entsteht also eine Art Spirale, in der sich Ursache und Wirkung abwechselnd umkehren: Mehr Infektionen bedingen einen vermehrten Versand von E-Mails und ein höheres Aufkommen an E-Mails zieht eine höhere Anzahl an Infektionen nach sich. Auf den Verbreitungsverlauf wird sich dieser Umstand so aus, dass der Wurm unmittelbar nach dem Eintritt in die Wachstumsphase eine so starke Verbreitung erreicht, dass die Verbreitung aufgrund der beschriebenen Spirale im weiteren Verlauf der Phase exponentiell ansteigt.
3. **Kontinuitätsphase**: Da bei den Simulationen die Anzahl der simulierten Rechner begrenzt ist, erreicht die Wurmverbreitung ab einem bestimmten Zeitpunkt eine Art Sättigung und stabilisiert sich. Die Phase der Wurmverbreitung nach diesem Zeitpunkt haben wir Kontinuitätsphase genannt. In dieser werden zwar weiterhin neue Rechner befallen, jedoch ungefähr auch in gleichem Maße bereinigt. Dadurch ergibt sich ein recht waagerechter und kontinuierlicher Verbreitungsverlauf. Das Maximum des Verbreitungsverlaufs liegt ebenfalls in dieser Phase. Typischerweise befindet sich das Maximum am Phasenanfang.

Alle von uns erstellten Verbreitungsverläufe steigen nach ihrem Eintritt in die Kontinuitätsphase noch leicht an, erreichen dann ihr Maximum und fallen wieder leicht ab. Kein Verlauf ist über die gesamte Simulationsdauer monoton steigend. Der Grund dafür ist, dass bei den von uns durchgeführten Simulationen befallene Rechner nach einer gewissen Zeit wieder bereinigt werden. Wäre dies nicht der Fall, könnte die Anzahl der Infektionen im Verlauf der Simulation niemals sinken.

Warum das Maximum der Verbreitungsverläufe stets am Anfang der Kontinuitätsphase zu finden ist, lässt sich folgendermaßen erklären: Nachdem sich in der Wachstumsphase immer mehr Rechner mit dem E-Mail-Wurm infizieren, werden in dieser Phase natürlich auch immer mehr E-Mails verschickt. Da der Erwartungswert der Zeitspannen zwischen dem Eintreffen und dem Lesen einer E-Mail ungefähr $1{,}5$ Stunden beträgt, liegen die E-Mails auch durchschnittlich so lange ungelesen im Posteingang der Empfänger. Anschließend werden diese mit einer bestimmten Wahrscheinlichkeit gelesen und es wird, falls ein Empfänger zusätzlich dem Verweis folgt, aufgrund

der daraus resultierenden Infektion ein Reparaturvorgang eingeleitet (siehe Abbildung 5.8). Der Reparaturvorgang wird durchschnittlich ca. 67 Stunden später durchgeführt. Das Maximum des Verbreitungsverlaufs aus Abbildung 6.1 liegt bei ca. 75 Stunden. Subtrahiert man davon die genannten *Verzögerungszeiten* von 67 Stunden und 1,5 Stunden erhält man genau den Zeitpunkt, zu dem die Wurmverbreitung in ihre Wachstumsphase eintritt – nämlich nach ca. 6,5 Stunden. Der Zeitpunkt, an dem maximal viele Rechner infiziert sind, ist also unter der Bedingung, dass man das Benutzerverhalten kennt, berechenbar.

In dem Paper *Propagation of active worms: A survey* [XFZ09] von Xiang, Fan und Zhu findet sich ebenfalls eine Einteilung der Wurmverbreitung in Phasen. Die Autoren betrachten jedoch ausschließlich autonome Computerwürmer und identifizieren folgende vier Phasen:

1. Die **Latency Phase** entspricht der Zeit zwischen der Veröffentlichung einer neuen Sicherheitslücke und der ersten Beobachtung eines Wurms, der diese Lücke ausnutzt.
2. In der **Growth Phase** versucht ein Computerwurm, neue Rechnersysteme zu infizieren.
3. In der **Decay Phase** werden infizierte Systeme gepatcht oder komplett vom Netz genommen.
4. In der **Persistence Phase** sind nur noch wenige Rechner infiziert; der Großteil der Rechner ist bereinigt.

Die Autoren haben diese Phasen nicht an einem Beispiel verdeutlicht, so dass ein visueller Vergleich mit den drei von uns identifizierten Phasen nicht möglich ist. Da sich Social Malcode jedoch komplett ohne das Ausnutzen von Sicherheitslücken verbreiten kann[2], ist die *Latency Phase* bei der Verbreitung von Social Malcode im Allgemeinen nicht relevant.

Des Weiteren entspricht die *Growth Phase* von Xiang et al. einer Kombination aus der von uns identifizierten *Start-* und *Wachstumsphase*, da in diesen beiden Phasen ein Anstieg an infizierten Systemen festzustellen ist. Der von uns als *Kontinuitätsphase* bezeichnete Teil eines Verbreitungsverlaufs entspricht wiederum, aufgrund der immer stärker einsetzenden Gegenmaßnahmen, am ehesten einer Kombination der *Decay Phase* und *Persistence Phase* von Xiang et al. Zusammenfassend zeigt Tabelle 6.1 eine Gegenüberstellung der Phasen, die Xiang et al. bei der Verbreitung eines autonomen Computerwurms beobachtet haben, mit den von uns bei der Verbreitung des E-Mail-Wurms identifizierten Phasen.

6.2.2. Verbreitungsverlauf von Code Red 2

Es existieren jedoch viele Veröffentlichungen, in denen ebenfalls Verbreitungsverläufe von autonomen Computerwürmern visualisiert werden. Chen, Gao und Kwiat haben in ihrer Arbeit *Modeling the Spread of Active Worms* [CGK03] mit Hilfe des AAWP-Modells einen Wurm simuliert, der sich ähnlich wie der Computerwurm Code Red 2

[2] Der Zustandsübergang u_{i_3} der Definition 3.8 beruht nicht zwangsläufig auf dem Ausnutzen einer Sicherheitslücke. Beispielsweise kann ein Angreifer Social Malcode auch als regulären Dateianhang versenden und das Opfer „bitten“, diesen Anhang auszuführen.

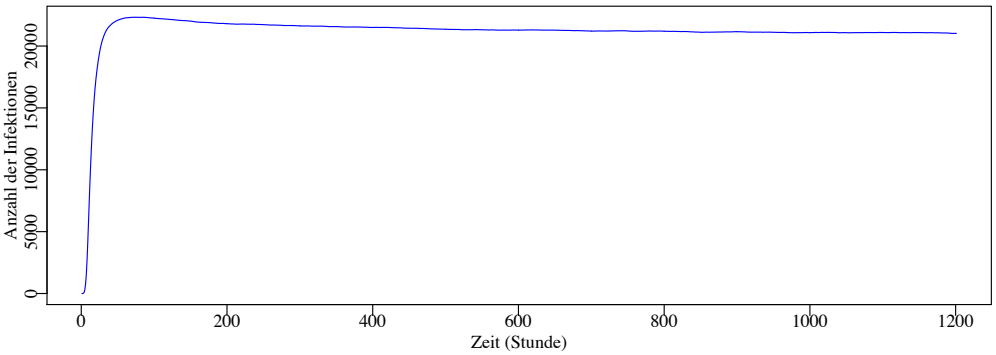
Tabelle 6.1.: Gegenüberstellung der Phasen einer Wurmverbreitung
In der linken Spalte befinden sich die drei von uns identifizierten Phasen der Verbreitung von Social Malcode. Diesen gegenübergestellt sind die von Xiang et al. identifizierten Phasen der Verbreitung eines autonomen Computerwurms.

Phasen der vorliegenden Arbeit	Phasen von Xiang et al.
—	Latency Phase
Startphase Wachstumsphase	Growth Phase
Kontinuitätsphase	Decay Phase Persistence Phase

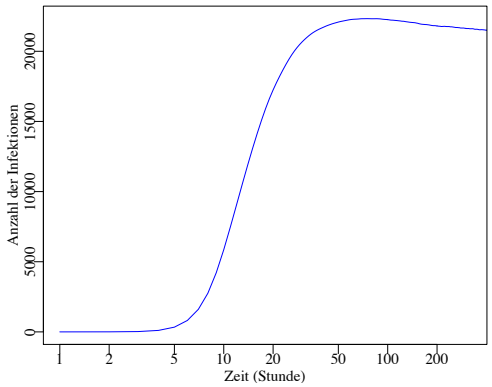
verbreitet. Abbildung 6.2c enthält eine Kopie der Abbildung 3(a) aus der Arbeit von Chen et al. Deren Simulation beruht auf einem Netz mit $500\,000$ Rechnern. Zu Beginn der Simulation war lediglich ein einziger Rechner mit dem simulierten Wurm infiziert.

Vergleicht man den vom Chen et al. erzeugten Verbreitungsverlauf mit dem Verbreitungsverlauf des von uns simulierten E-Mail-Wurms aus Abbildungsteil (a), erkennt man auf den ersten Blick keine große Ähnlichkeit. Dies liegt vor allem an der Simulationszeit, die bei unseren Simulationen 50 Tage betrug. Die Simulation von Chen et al. ist wesentlich kürzer, da sich Code Red 2 nur wenige Stunden aktiv verbreitete. In Abbildungsteil (b) haben wir nur den Anfang des Verbreitungsverlaufs aus Abbildungsteil (a) dargestellt. Im Gegensatz zum kompletten Verbreitungsverlauf ist die x-Achse dieses Diagramms zusätzlich logarithmisch skaliert. In dieser Darstellung ist nun die Ähnlichkeit zwischen den Grundformen der Verbreitungsverläufe von Code Red 2 und des von uns simulierten E-Mail-Wurms ersichtlich. Auch bei dem Verbreitungsverlauf von Code Red 2 lassen sich die drei von uns identifizierten Phasen nachvollziehen: Nach einem längeren, flachen Anstieg der Infektionszahlen (Startphase), steigen diese in einer zweiten Phase exponentiell an (Wachstumsphase). Im weiteren Verlauf der Verbreitung stabilisiert sich die Anzahl der infizierten Rechner und bleibt relativ kontinuierlich auf einem leicht abfallenden Niveau (Kontinuitätsphase).

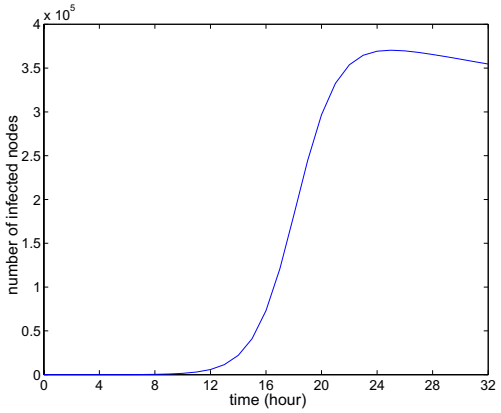
Ein Unterschied besteht jedoch in der Ursache des leichten Abfalls: Bei unseren Simulationen beruht das leicht sinkende Niveau auf Rechnerreinigungen. Bei der Arbeit von Chen et al. ist der Grund, dass Rechner mit der Zeit gepatcht oder aus dem Netz entfernt werden. Somit tragen diese Rechner nicht mehr zur Verbreitung von Code Red 2 bei. Die infizierten Rechner unserer Simulation tragen jedoch nur kurzfristig nicht zur Verbreitung des E-Mail-Wurms bei, da sie nach ihrer Reinigung wieder infiziert werden können und somit auch wieder E-Mails verschicken, über die sich der Wurm verbreitet. Deshalb ist die Neigung bei dem Verbreitungsverlauf von Chen et al. auch stärker abfallend als bei unseren Verbreitungsverläufen. Durch die logarithmische Skalierung der x-Achse wird dieser Effekt in den von uns produzierten Verbreitungsverläufen zudem noch weiter verstärkt. Bei einer normal skalierten x-Achse, wäre die Neigung also geringer.



(a) Kompletter Verbreitungsverlauf des simulierten E-Mail-Wurms mit einer Standardbelegung der Simulationsparameter



(b) Anfang der Verbreitungsverlaufs aus Teil (a)



(c) AAWP-Simulation von Code Red 2 (Abbildung 3(a) aus Chen et al. [CGK03])

Abbildung 6.2.: Verbreitungsverläufe des simulierten E-Mail-Wurms und Code Red 2
Teil (a) zeigt den Verbreitungsverlauf des simulierten E-Mail-Wurms mit einer Standardbelegung der Simulationsparameter. Der Anfang dieses Verlaufs ist mit einer logarithmisch skalierten x-Achse in Abbildungsteil (b) dargestellt. Teil (c) der Abbildung enthält eine AAWP-Simulation von Code Red 2, die von Chen et al. erstellt wurde.

Der größte Unterschied zwischen den beiden Verbreitungsverläufen der Abbildungsteile (b) und (c) wird aber durch die logarithmische Skalierung ein wenig verschleiert: Die Ausbreitung des E-Mail-Wurms benötigt wesentlich mehr Zeit als die Ausbreitung von Code Red 2. In dem Verbreitungsverlauf des von uns simulierten E-Mail-Wurms ist im Vergleich zum Verbreitungsverlauf des Abbildungsteils (c) die 10-fache Zeitspanne dargestellt. Zudem waren bei uns zu Beginn der Simulation bereits 20 Rechner infiziert, bei Chen et al. war es hingegen nur ein einzelner Rechner. Die Gründe für diese viel langsamere Verbreitung sind mannigfaltig:

- **Verzögerungszeiten**: Der von uns simulierte Wurm verbreitet sich durch E-Mails. Diese liegen nach ihrer Zustellung für eine bestimmte Zeit ungelesen im Posteingang des Empfängers. Bis zum Lesen der E-Mails kann der Wurm den entsprechenden Rechner nicht infizieren. Dadurch entsteht eine Art menschlich

verursachte Verzögerung in der Verbreitung des E-Mail-Wurms. Code Red 2 hingegen infiziert einen verwundbaren Rechner sofort.

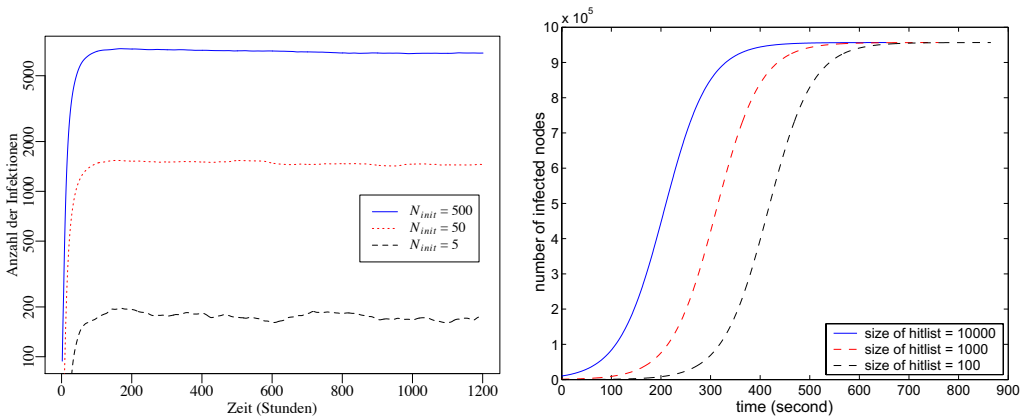
- **Wahrscheinlichkeit einer Infektion**: In den von uns durchgeführten Simulationen müssen einige Bedingungen erfüllt sein, damit ein Rechner durch den E-Mail-Wurm infiziert werden kann. Erst wenn ein Empfänger eine E-Mail liest und auch dem darin enthaltenen Verweis folgt, gilt ein Rechner als infiziert. Dagegen ist es bei der Simulation von Chen et al. ausreichend, dass eine Wurminstanz auf einem Rechner A bei der Zielgenerierung die IP-Adresse eines Rechners B generiert. Sobald die Instanz des Wurms auf Rechner A mit dem Angriff auf die generierten Ziele beginnt, gilt Rechner B automatisch als infiziert.
- **Netztopologie**: Ein weiterer Grund für die langsamere Verbreitung des E-Mail-Wurms liegt in dessen lokaler Ausbreitung. Von einem Knoten des E-Mail-Netzes können nur unmittelbar angrenzende Knoten befallen werden. Diese Einschränkung beruht auf der ausschließlichen Verwendung der Adressbücher und des Posteingangs bei der Generierung neuer E-Mail-Adressen (siehe Abschnitt 5.2.1). Der von uns simulierte E-Mail-Wurm muss zur Infektion eines beliebigen Rechners also zuerst die eventuell dazwischen liegenden Rechner infizieren. Code Red 2 hingegen kann jeden Rechner des Simulationsnetzes innerhalb eines einzigen Infektionszyklus befallen.

In diesem Zusammenhang fällt jedoch auch auf, dass, entgegen obiger Feststellung, der Verbreitungsverlauf des simulierten E-Mail-Wurms früher von der Startphase in die Wachstumsphase wechselt. Bei dem von uns simulierten E-Mail-Wurm geschieht dies nach ca. $6{,}5$ Stunden und bei der Simulation von Code Red 2 nach ca. 14 Stunden. Dies kann zum einen durch die höhere Anzahl initial infizierter Knoten bei den von uns durchgeführten Simulationen begründet werden und zum anderen durch die viel höhere Gesamtpopulation bei der Simulation von Chen et al. Der Übergang zwischen der Start- und der Wachstumsphase findet immer dann statt, wenn der relative Anteil von infizierten Knoten einen bestimmten Wert übersteigt. Im Vergleich zu einem kleinen Netz, findet bei einem größeren Netz der Wechsel zwischen den beiden angesprochenen Phasen somit später statt.

6.2.3. Größe der Hit-List eines autonomen Wurms

In der gleichen Arbeit [CGK03] untersuchen Chen et al. ebenfalls den Effekt einer unterschiedlich großen Hit-List auf den Verbreitungsverlauf eines autonomen Computerwurms. In Abbildung 6.3b befindet sich eine Kopie der Abbildung 1a aus ihrem Paper. Der von ihnen simulierte Wurm generiert zufällig neue Ziel-IP-Adressen, überprüft 100 Rechner pro Sekunde auf das Vorhandensein einer ausnutzbaren Sicherheitslücke und eine Infektion benötigt genau eine Sekunde. Bei ihrer Simulation benutzen sie eine Säuberungsrate (in der Arbeit als *death rate* bezeichnet) von $0{,}001$/Sekunde[3] und verwenden ein Netz bestehend aus $1\,000\,000$ Rechnern.

[3] Dies bedeutet, dass ein Tausendstel aller aktuell infizierten Rechner in der nächsten Sekunde wieder von dem Wurm bereinigt ist.



(a) Effekt verschieden vieler Startknoten bei unseren Simulationen

(b) Effekt der Größe einer Hit-List (Abbildung 1a aus Chen et al. [CGK03])

Abbildung 6.3.: Vergleich zwischen Anzahl von Startknoten und Größe von Hit-List
Teil (a) zeigt den Effekt unterschiedlich vieler Startknoten bei den von uns durchgeführten Simulationen. Teil (b) ist die Kopie einer Abbildung aus der Arbeit von Chen et al., die den Einfluss unterschiedlich großer Hit-Listen zeigt. In ihrer Arbeit ist dies Abbildung 1a.

Da bei den Simulationen von Chen et al. die IP-Adressen der Hit-List zu Beginn der Simulation als infiziert gelten, sind sie das Gegenstück zu dem von uns verwendeten Simulationsparameter N_{init}. Aus diesem Grund haben wir der Abbildung von Chen et al. drei Verbreitungsverläufe unserer Simulation mit einer unterschiedlichen Belegung von N_{init} gegenübergestellt. Diese drei Verläufe befinden sich in Abbildung 6.3a. Beide Ergebnisse haben gemeinsam, dass sich die simulierten Würmer langsamer verbreiten, je weniger Rechner zu Beginn der Simulation befallen sind.

Ein wesentlicher Unterschied ist jedoch, dass bei Chen et al. alle drei Verbreitungsverläufe nach einer gewissen Zeit das gleiche Infektionsniveau erreichen. Bei den von uns durchgeführten Analysen ist dies nicht der Fall: Die Verbreitungsverläufe, bei denen initial mehr Rechner mit dem Wurm infiziert waren, erreichen auch über die komplette Simulationsdauer eine viel höhere Anzahl an Infektionen. Die Simulationen mit einem niedrigeren N_{init} erreichen nur geringere Infektionsraten.

Dieser Unterschied lässt sich folgendermaßen erklären: Bei Chen et al. wird zu jedem Zeitpunkt der Simulation, in Abhängigkeit von der Anzahl aktuell infizierter Systeme, ein Anteil neuer Rechner infiziert und ein Anteil bereits infizierter Rechner bereinigt. Beide Raten, also sowohl die Süberungs- als auch die Infektionsrate, bleiben während der gesamten Simulationsdauer konstant. Gleiches gilt für die Gesamtanzahl der Rechner. Daraus resultiert, dass die Anzahl der infizierten Systeme im Verlauf der Zeit unweigerlich gegen einen festen Wert konvergiert. Die Verbreitungsverläufe werden durch die unterschiedlichen Größen der Hit-Listen anfänglich nur unterschiedlich stark verzögert bzw. beschleunigt – entwickeln sich danach aber bei allen Hit-List-Größen gleich.

Bei unseren Simulationen benutzen wir keine Infektionsrate. Die Anzahl der Rechner, die von einem neu infizierten Rechner befallen werden können, hängt zum einen von der Größe des Adressbuches ab und zum anderen von dem Verhalten der Benut-

zer, denen die E-Mail-Adressen des Adressbuches zuzuordnen sind. Des Weiteren verwenden wir keine feste Säuberungsrate, sondern jedem neu infizierten Rechner wird sozusagen ein eigener Säuberungsprozess zugewiesen (siehe Abschnitt 5.2.3). Dieser tritt zeitverzögert nach der Infektion ein – jede Infektion zieht somit eine Säuberung nach sich. Es gilt also:

- Eine Infektion impliziert immer eine zeitverzögerte Säuberung.
- Die Anzahl der Neuinfektion ist abhängig von der Adressbuchgröße und dem Verhalten der Benutzer.

Bei diesem System ergibt sich im Laufe der Simulation irgendwann ein Gleichgewicht, wodurch genauso viele Rechner gesäubert wie neu infiziert werden. Dementsprechend bleibt der daraus resultierende Verbreitungsverlauf ab einem gewissen Zeitpunkt nahezu auf einem Niveau. Die Anzahl der ersten Infektionen hängt von der Anzahl der Rechner ab, in deren Posteingang initial eine E-Mail vorhanden ist. Sollte es durch diese zu mindestens einer Infektion kommen, ist das beschriebene System zu Beginn der Simulation also in einem Ungleichgewicht, da bereits Rechner infiziert sind, aber noch keine Säuberungen durchgeführt werden. Somit hängt der Zeitpunkt, ab dem sich ein Gleichgewicht in diesem System einstellt, von der Anzahl der Rechner ab, die initial eine E-Mail in ihrem Posteingang haben.

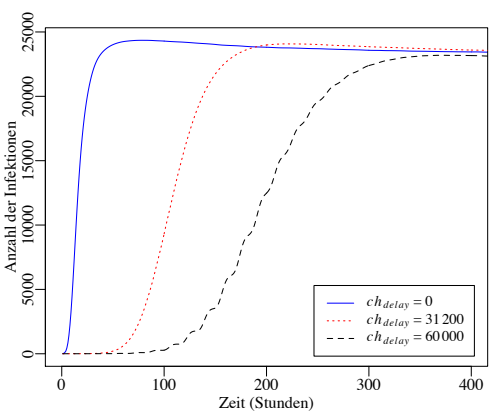
Die Größe des Parameters N_{init} bestimmt also den Zeitpunkt, zu dem sich ein Gleichgewicht zwischen Neuinfektionen und Säuberungen einstellt, was wiederum gleichbedeutend mit einem nahezu konstanten Infektionsniveau ist. Aus diesem Grund pendeln sich die Verbreitungsverläufe unser Simulation, abhängig von der Anzahl der Startknoten, auf ein unterschiedliches Niveau ein und nicht alle auf das gleiche Niveau, wie dies bei Chen et al. passiert.

6.2.4. Verzögerung durch die Dauer einer Infektion

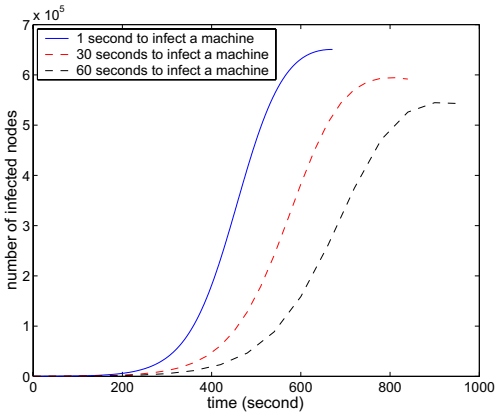
Ebenfalls in der Arbeit *Modeling the Spread of Active Worms* [CGK03] haben Chen et al. den Einfluss der Dauer analysiert, die von einem Computerwurm zur Infektion eines neuen Rechners benötigt wird. Ihre dafür durchgeführte Simulation lief mit fast den gleichen Einstellungen, wie die Simulation zur Analyse unterschiedlich großer Hit-Listen. Die einzigen beiden Unterschiede sind, dass diesmal eine feste Hit-List mit 100 IP-Adressen verwendet wurde und dass zusätzlich Rechner immun gegen den Wurm werden können – und zwar mit einer Rate von $0,0005$/Sekunde[4]. In Abbildung 6.4b sind die Ergebnisse dieser Untersuchung dargestellt. Die Abbildung ist eine Kopie der Abbildung 1c aus der Arbeit von Chen et al.

Da die benötigte Infektionsdauer eines autonomen Computerwurms dessen Verbreitung verlangsamt, ist die Infektionsdauer der von uns analysierten künstlichen Verzögerung in der Zustellgeschwindigkeit einer E-Mail ähnlich. In Abbildung 6.4a sind aus diesem Grund drei von uns generierte Verbreitungsverläufe zu sehen, die bei der Analyse des Simulationsparameters ch_{delay} erzeugt wurden. Der Effekt der beiden untersuchten Größen, die Infektionsdauer bei einem autonomen Wurm auf der einen Seite und die Zustellgeschwindigkeit einer E-Mail auf der anderen Seite, sind nahezu

[4]Dies bedeutet, dass pro Sekunde $\frac{1}{2000}$ aller Rechner, die noch nicht immun sind, immun gegen den Wurm werden.



(a) Effekt einer verzögerten E-Mail-Zustellung bei unseren Simulationen



(b) Effekt der Infektionsdauer (Abbildung 1c aus Chen et al. [CGK03])

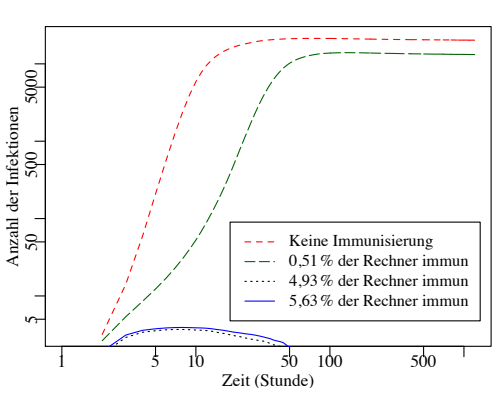
Abbildung 6.4.: Vergleich zwischen Infektionsdauer und verzögerter E-Mail-Zustellung Teil (a) zeigt den Effekt einer Verzögerung in der Zustellgeschwindigkeit bei den von uns durchgeführten Simulationen. In Teil (b) ist eine Abbildungskopie aus der Arbeit von Chen et al. dargestellt, die den Einfluss der Zeitspanne verdeutlicht, die eine einzelne Infektion eines autonomen Wurms in Anspruch nimmt. In ihrer Arbeit ist dies Abbildung 1c.

identisch: In beiden Fällen tritt eine Verzögerung auf, die die Verbreitung des Schadprogramms verlangsamt. Der größte Unterschied zwischen den Verbreitungsverläufen besteht darin, dass bei Chen et al. die drei Verläufe nicht auf ein ähnliches Niveau kommen, wie dies bei den von uns erstellten Verbreitungsverläufen der Fall ist. Dies hängt aber nicht mit der unterschiedlich langen Infektionsdauer zusammen, sondern ist der von Chen et al. verwendeten Immunisierung geschuldet. Durch die länger andauernden Infektionen, verzögert sich die Verbreitung des von ihnen simulierten Wurms. Dadurch dauert es wesentlich länger, bis die Verbreitung ihr Maximum erreicht und somit werden bis dahin wesentlich mehr Rechner immun gegen den Wurm. Dementsprechend sinkt auch die maximale Anzahl der Rechner, die infiziert werden können.

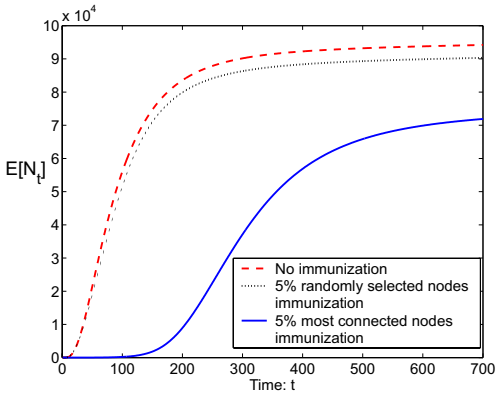
6.2.5. Gezielte Immunisierung zentraler Knoten

Wie bereits bei der Beschreibung verwandter Arbeiten erwähnt, haben Zou et al. ebenfalls die Auswirkungen einer Immunisierung auf die Verbreitung eines E-Mail-Wurms untersucht. In ihrem Paper *Email Worm Modeling and Defense* [ZTG04] stellen sie die Ergebnisse ihrer Arbeit vor. Da wir uns bei unseren Simulationen ausschließlich auf ein Simulationsnetz konzentriert haben, dessen Verzweigungsgrade einem Potenzgesetz genügen (die Gründe dafür haben wir in Abschnitt 5.2.2.1 erörtert), vergleichen wir hier die entsprechenden Ergebnisse miteinander. In Abbildung 6.5b sind die Ergebnisse von Zou et al. grafisch dargestellt. Dieser Abbildungsteil entspricht bis auf die fehlende Überschrift der Abbildung 6a der genannten Arbeit.

Zou et al. haben bei ihren Untersuchungen ein Simulationsnetz mit $100\,000$ Computern verwendet. Der durchschnittliche Verzweigungsgrad beträgt 8 (Kanten pro Knoten) und bei der Generierung des Netzes haben sie den Exponenten $\alpha = 1{,}7$ benutzt. In der Abbildung ist die Anzahl der Infektionen $E[N_t]$ gegen die Zeit aufgetragen. Die



(a) Effekt einer gezielten Immunisierung bei unseren Simulationen



(b) Effekt einer gezielten Immunisierung bei Zou et al. (Abbildung 6a aus Zou et al. [ZTG04])

Abbildung 6.5.: Vergleich des Effektes einer gezielten Immunisierung
Teil (a) zeigt den Effekt einer gezielten Immunisierung bei den von uns durchgeführten Simulationen. In Teil (b) ist eine Abbildungskopie aus der Arbeit von Zou et al. dargestellt, die diesen Effekt ebenfalls untersucht haben. In ihrer Arbeit ist dies die Abbildung 6a.

genaue Einheit der Zeitachse ist in der Arbeit nicht vermerkt, sondern allgemein mit *time tick* beschrieben. Die gestrichelte, rote Linie entspricht dem Verbreitungsverlauf, der sich ergibt, wenn keine Immunisierung verwendet wird. Die schwarze Linie, bestehend aus kleinen Querstrichen, repräsentiert den Verbreitungsverlauf, der entsteht, wenn $5\,000$ zufällig ausgewählte Rechner immun gegen den Wurm sind. Den Verbreitungsverlauf, falls die $5\,000$ Rechner mit dem höchsten Verzweigungsgrad immun gegen den Wurm sind, zeigt die durchgezogene, blaue Linie.

Die von uns durchgeführten Simulationen des Parameters d_{immune}, der die minimale Adressbuchgröße eines Rechners enthält, damit dieser immun gegen den simulierten E-Mail-Wurm ist, sind mit den Analysen Zou et al. vergleichbar. Abbildung 6.5a zeigt den von uns gemessenen Effekt in einer ähnlichen Darstellungsweise, wie sie Zou et al. verwendet haben. Hier sind jedoch beide Achsen logarithmisch skaliert, damit zum einen die Verbreitungsverläufe mit den niedrigen Infektionszahlen und zum anderen die Unterschiede in der Verbreitungsgeschwindigkeit besser erkennbar sind. Da wir dem Parameter d_{immune} bei dessen Analyse sukzessive alle in unserem Simulationsnetz vorkommenden Verzweigungsgrade zugewiesen haben, haben wir für einen Vergleich leider keinen Verbreitungsverlauf zur Verfügung, bei dem exakt $5\,\%$ der Rechner immun sind. Jedoch haben $2\,813$ Rechner (dies entspricht ca. $5{,}63\,\%$ aller Rechner) mindestens 14 Nachbarknoten und $2\,465$ Rechner (ca. $4{,}93\,\%$) mindestens 15 Nachbarknoten. Die beiden entsprechenden Verbreitungsverläufe der Abbildung 6.5a zeigen, dass eine gezielte Immunisierung bei den von uns durchgeführten Simulationen einen höheren Effekt hat als bei den von Zou et al. durchgeführten Simulationen. Während bei ihnen die gezielte Immunisierung lediglich zu einer Verlangsamung der Wurmausbreitung führt, ist bei uns der Effekt so stark, dass der Wurm nach kurzer Zeit komplett *ausstirbt*.

Bereits eine gezielte Immunisierung von nur $0{,}51\,\%$ der Rechner erzielt bei unseren Analysen einen nachweisbaren Effekt. Der entsprechende Verbreitungsverlauf ist ebenfalls in Abbildung 6.5a dargestellt. Bei dieser Simulation sind alle Rechner mit mindestens 50 Kontakten in ihrem Adressbuch immun gegen den Wurm. Neben einer langsameren Ausbreitung hat eine gezielte Immunisierung ebenfalls eine niedrigere Anzahl an infizierten Rechnern am Ende der Simulation zur Folge: Bei einem Wert von $d_{immune} = 50$ sind am Ende der Simulation über $7\,000$ Rechner weniger infiziert als bei einem kompletten Verzicht auf Immunisierung. Durch eine gezielte Immunisierung, tragen in erster Linie die Rechner nicht mehr aktiv zur Verbreitung des Wurms bei, die in einem Schritt E-Mails an sehr viele Empfänger verschicken können. Alle Benutzer, die nur mit Benutzern solcher Rechnern in Kontakt stehen, sind somit auch vor dem simulierten Wurm geschützt. Entsprechend fällt die Anzahl der Rechner, die überhaupt infiziert werden können, mit dem Parameter d_{immune}.

Die Ursache für den größeren Effekt einer gezielte Immunisierung bei den von uns durchgeführten Simulationen, liegt wohl an dem durchschnittlich geringeren Verzweigungsgrad unseres Netzes. Bei den Analysen von Zou et al. hat ein Rechner im Schnitt 8 E-Mail-Adressen in seinem Adressbuch. Bei unseren Analysen sind dies im Schnitt hingegen nur $5{,}194$ Kontakte. Somit infiziert der von Zou et al. simulierte Wurm auch bei Knoten mit einem geringeren Verzweigungsgrad durchschnittlich mehr Rechner als bei den von uns durchgeführten Simulationen.

Im Vergleich zur Arbeit von Zou et al. bewirkt also bei unseren Simulationen bereits die gezielte Immunisierung eines geringeren Prozentsatzes an Rechnern einen positiven Effekt auf die Verbreitung des Wurms. Doch wie bereits in Abschnitt 5.3.8 beschrieben, muss in einem realen Szenario ganz individuell entschieden werden, ob und in welchem Ausmaß eine gezielte Immunisierung sinnvoll bzw. rentabel ist. Legt man beispielsweise die Gesamtzahl von ungefähr $3{,}9$ Milliarden E-Mail-Konten im Jahr 2013 weltweit zugrunde [RL13], so entsprechen $0{,}51\,\%$ immerhin noch fast 20 Millionen E-Mail-Konten. Vorausgesetzt, jedes dieser Konten befindet sich auf einem eigenen Rechner, müssten genauso viele Rechner, bzw. deren Benutzer, gegen einen einzigen E-Mail-Wurm gehärtet werden. Technisch und organisatorisch wird dieser Aufwand wohl kaum realisierbar sein. Jedoch könnte man auf diese Weise sicherlich die Verbreitung des Wurms in bestimmten Teilnetzen einschränken, indem man gezielt zentrale Rechner einer bestimmten Region oder Organisation gegen den Wurm absichert.

6.3. Kombinierte Gegenmaßnahmen

In Abschnitt 5.3 haben wir die Effekte jedes einzelnen Simulationsparameters auf die Verbreitung des simulierten E-Mail-Wurms untersucht. Die Parameter, die sich extern manipulieren lassen, sind für Gegenmaßnahmen gegen die Verbreitung des Wurms besser geeignet als die restlichen Parameter. So hängt beispielsweise die Zeitspanne, die zwischen dem Eintreffen einer E-Mail und dem Lesen durch den Empfänger vergeht, nur vom Empfänger der E-Mail selbst ab. Um extern Einfluss auf diesen Wert zu nehmen, müsste man also das Verhalten des Empfängers manipulieren. Der Simulationsparameter, der der Zustellgeschwindigkeit der E-Mails entspricht, lässt sich

hingegen extern manipulieren. So kann man etwa die beteiligten Mailserver derart konfigurieren, dass sie entsprechende E-Mails vor ihrer Zustellung künstlich zurückhalten.

Tabelle 6.2 enthält eine Übersicht über die externen Manipulationsmöglichkeiten der einzelnen Simulationsparameter. Neben jedem Parameter ist angegeben, ob sich dieser extern manipulieren lässt. In der dritten Spalte befindet sich jeweils eine kurze Begründung für die getroffene Einstufung. Die Einstufung einiger Simulationsparameter wird im folgenden noch genauer erläutert.

Tabelle 6.2.: Externe Manipulationsmöglichkeit der Simulationsparameter
Neben jedem Simulationsparameter ist angegeben, ob und warum sich dieser, bzw. warum sich dieser nicht, extern manipulieren lässt.

Parameter	Extern manipulierbar?	Begründung
p_{read}	Bedingt	Abhängig vom Verhalten der Benutzer, könnte jedoch durch eine bessere Spam-Erkennung reduziert werden
p_{click}	Bedingt	Abhängig vom Verhalten der Benutzer, könnte jedoch durch eine Vorabüberprüfung der Verweise reduziert werden
$\lambda_{cleaning}$	Nein	Abhängig vom Verhalten der Benutzer
$\alpha_{reading}$ $C_{reading}$	Nein Nein	Beide Werte hängen vom Verhalten der Benutzer ab
N_{init}	Nein	Abhängig vom Verhalten des Angreifers
ch_{delay}	Ja	Entsprechende Konfiguration der Mailserver
p_{immune}	Bedingt	Abhängig vom Verhalten der Benutzer, könnte jedoch durch eine bessere Aufklärung reduziert werden
d_{immune}	Ja	Zentrale Rechner sind gezielt zu immunisieren

Einige Parameter sind nur bedingt extern manipulierbar. Beispielsweise hängt die Wahrscheinlichkeit, mit der Benutzer einem Verweis folgen, nur von den Benutzern selbst ab. Jedoch sind E-Mail-Clients denkbar, die die Verweise in einer E-Mail als verdächtig markieren, falls die Domain des Verweises auf einer internen Blackliste steht oder sich der HTML-Text des Verweises von dem eigentlichen Verweisziel unterscheidet. Solch eine Markierung würde die Wahrscheinlichkeit, dass ein Empfänger dem Verweis folgt, sicherlich reduzieren. Jedoch kann der Erfolg solch einer Maßnahme nicht mit absoluter Sicherheit vorhergesagt werden.

Da für eine gezielte Immunisierung nur wenige, zentrale Rechner immunisiert werden müssen, ist eine gezielte Immunisierung laut Tabelle stets durchführbar. Zu den zentralen Rechnern eines E-Mail-Netzes zählen zum Beispiel auch Server, die zum Versand von elektronischen Newslettern benutzt werden, da diese häufig sehr viele Empfänger adressieren. Diese sind, bezogen auf den E-Mail-Wurm, bereits von sich

aus immun gegen den Wurm, da die Newsletter normalerweise automatisiert verschickt werden und kein menschlicher Benutzer eingehende E-Mails liest und somit auch nicht dem Verweis innerhalb der E-Mail gefolgt wird. Folglich haben wir in der Tabelle den Parameter d_{immune} als extern manipulierbar eingestuft.

Alle Simulationsparameter, die extern manipulierbar sind, eignen sich für mögliche Maßnahmen gegen die Verbreitung des E-Mail-Wurms. Abbildung 6.6 zeigt zwei Verbreitungsverläufe: Die durchgehende, schwarze Linie repräsentiert den Verbreitungsverlauf, der aus einer Simulation mit der Standardbelegung der Simulationsparameter (siehe Tabelle 5.1) resultiert. Dies entspricht einer Verbreitung des Wurms komplett ohne Gegenmaßnahmen, da keine gezielte Immunisierung stattfindet, keine Rechner immun gegen den Wurm werden können und auch die E-Mails verzögerungsfrei zugestellt werden.

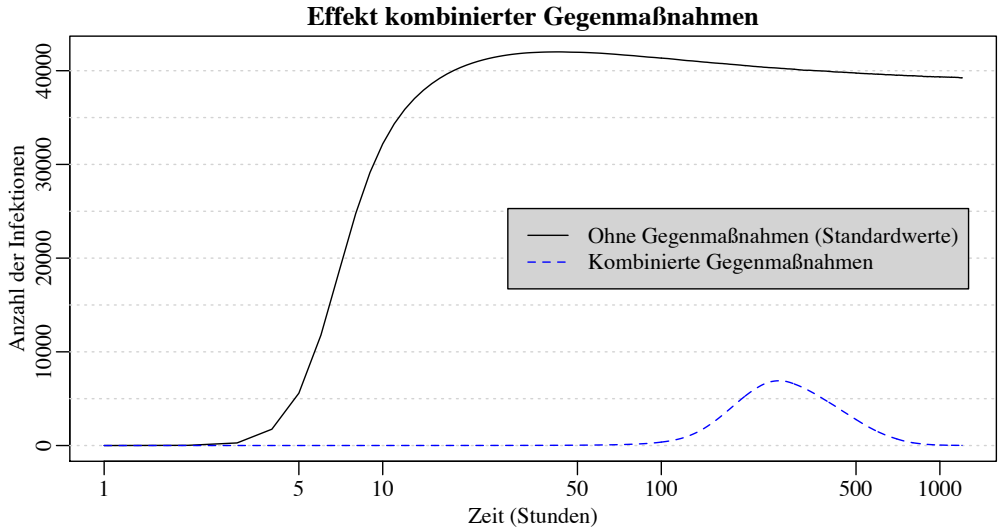


Abbildung 6.6.: Kombinierte Gegenmaßnahmen
Der gestrichelte Verbreitungsverlauf zeigt den Effekt von kombinierten Maßnahmen gegen die Verbreitung des simulierten E-Mail-Wurms. Bei der Simulation des Verbreitungsverlaufs ohne Gegenmaßnahmen waren alle Simulationsparameter mit ihren Standardwerten belegt.

Die gestrichelte, blaue Linie zeigt hingegen den Verbreitungsverlauf des E-Mail-Wurms, wenn verschiedene Gegenmaßnahmen kombiniert eingesetzt werden. Während der Simulation, aus der dieser Verbreitungsverlauf resultiert, wurden Rechner mit einer Wahrscheinlichkeit von $p_{immune} = 0{,}1$ nach einer Infektion immun gegen den simulierten Wurm, alle Rechner mit mindestens $d_{immune} = 50$ Kontakten in ihrem Adressbuch waren von Beginn der Simulation immun gegen den Wurm und die Zustellung der E-Mails wurde $ch_{delay} = 3$ Stunden künstlich hinausgezögert. Alle anderen Parameter enthielten bei dieser Simulation ihren Standardwert.

Die beiden Verbreitungsverläufe der Abbildung 6.6 verdeutlichen den Effekt von kombinierten Gegenmaßnahmen. Durch die künstliche Verzögerung in der Zustellgeschwindigkeit der E-Mails findet bei dem Vebreitungsverlauf mit Gegenmaßnahmen der Übergang von der Startphase in die Wachstumsphase erst wesentlich später statt als bei dem Verbreitungsverlauf ohne Gegenmaßnahmen. Die gezielte Immunisierung

bewirkt, dass das Maximum des Verbreitungsverlaufs wesentlich geringer ist als bei dem Verbreitungsverlauf ohne Gegenmaßnahmen, und die Immunisierung nach einer Infektion lässt den Verbreitungsverlauf nach dessen Maximum wieder stark abfallen. Die Verbreitung des simulierten E-Mail-Wurms kann also durch eine gezielte Kombination von Gegenmaßnahmen stark eingedämmt werden. Würden zudem noch eine Spam-Erkennung und eine Vorabüberprüfung der Verweise eingesetzt, würde der Verbreitungsverlauf noch flacher ausfallen.

6.4. Limitierungen und Ideen für weiterführende Arbeiten

Ein wesentlicher Punkt der Arbeit ist die Simulation der Verbreitung von Social Malcode. Für diesen Zweck wurde durch verschiedene Experimente und Skripte das Verhalten von menschlichen Computerbenutzern gemessen, damit dieses unmittelbar in die Simulationen einfließen kann. An dieser Stelle müssen wir darauf hinweisen, dass das gemessene Verhalten nicht das Verhalten einer einzigen homogenen Zielgruppe von Menschen ist. Vielmehr setzt es sich aus dem Verhalten unterschiedlicher Gruppen zusammen, bei denen jeweils das Verhalten kleinerer Teilaspekte im Bezug auf die Verbreitung von Social Malcode untersucht wurde. Eine Übersicht über die Zielgruppen der einzelnen Experimente/Skripte ist in Tabelle 6.3 zu finden.

Tabelle 6.3.: Zielgruppen der durchgeführten Experimente und Skripte
Dies ist eine Auflistung der einzelnen Gruppen, die im Fokus der von uns durchgeführten Experimente und Skripte standen.

Experiment/Skript	Zielgruppe
Spam-Experimente	Mitarbeiter von Unternehmen/Banken
Zeitspannen zwischen dem Eintreffen und dem Lesen einer E-Mail	Studenten und Lehrstuhlmitarbeiter
Zeitspannen zwischen der Infektion und der Reinigung eines Rechners	Opfer zweier Botnetze

So wurden mögliche Reaktionen auf E-Mails, also ob einem Verweis gefolgt wird oder ob zum Beispiel ein Dateianhang ausgeführt wird, nur bei Mitarbeitern von Unternehmen und Instituten aus dem Finanzdienstleistungssektor durchgeführt. Da diese die E-Mails in ihrem beruflichen Umfeld lesen, verhalten sie sich wahrscheinlich defensiver oder vorsichtiger beim Erhalt von E-Mails mit unbekannten Absenderadressen als beispielsweise Privatpersonen in einem häuslichen Umfeld. Des Weiteren sind Unternehmen, aufgrund gesetzlicher Auflagen, durch Rechenzentren, externe Dienstleister oder die eigene IT-Abteilung in der Regel besser abgesichert als eine private IT-Infrastruktur. Damit sind die Ergebnisse in Abschnitt 4.2.5 sehr charakteristisch für eben genau die in der Tabelle angegebene Zielgruppe.

Die Zeitspannen zwischen dem Eintreffen und dem Lesen von E-Mails wurden hingegen nur bei Postfächern von Studenten oder Mitarbeitern einer Fakultät für Wirtschaftsinformatik und Wirtschaftsmathematik ausgelesen. Es ist davon auszugehen, dass diese Zielgruppe ihr E-Mail-Postfach im Schnitt häufiger auf neue E-Mails hin überprüft als dies andere E-Mail-Nutzer tun. Somit sind auch die Ergebnisse des Abschnitts 4.3.4 eher typisch für eine Gruppe von Studenten und Mitarbeitern eines technischen Studiengangs als dass sie allgemeingültig sind.

Das simulierte Verhalten eines menschlichen Computerbenutzers in den Simulationen des Kapitels 5 entspricht deshalb nicht dem Verhalten einer in sich homogenen Gruppe, sondern ist eine Kombination aus dem Verhalten vieler unterschiedlicher Gruppen. Nichtsdestotrotz lässt sich damit die Verbreitung von Social Malcode simulieren. Unserem Wissen nach ist dies auch die erste Arbeit mit einer solchen Verbreitungssimulation, bei der so ausgiebig vorher das involvierte menschliche Verhalten experimentell bestimmt wurde.

Dennoch ist die Heterogenität des gemessenen Verhaltens ein Ansatzpunkt für weiterführende Arbeiten, bei denen tatsächlich das Verhalten einer einzelnen Gruppe verwendet wird. Des Weiteren sehen wir folgende Ansatzpunkte, um Social Malcode und/oder dessen Verbreitung noch eingehender zu untersuchen:

- **Plug-in für E-Mail-Clients**: Wie die Diskussion in Abschnitt 4.3.3 zeigt, liefert unser Skript zum Auslesen der Zeitspannen zwischen dem Eintreffen und dem Lesen einer E-Mail nur Näherungswerte und nicht die genauen Zeitspannen. Es wäre denkbar, ein Plug-in für erweiterbare E-Mail-Clients, wie zum Beispiel *Mozilla Thunderbird*, zu entwickeln, das diese Zeitspannen bestimmt. Dadurch, dass am Client an sich der Lesezeitpunkt exakt bestimmt werden kann, würde man so auch die exakten Zeitspannen ermitteln können.
- **Ermitteln, ob ein Dateianhang geöffnet wurde**: In den Ausführungen zu den von uns durchgeführten Spam-Experimenten des Abschnitts 4.2 haben wir bereits erörtert, dass es gewisse Schwierigkeiten gab, zu bestimmen, ob ein nicht ausführbarer Dateianhang geöffnet wurde oder nicht. Hier könnte man Überlegungen anstellen, wie dieser Umstand besser und präziser zu ermitteln ist – beispielsweise mittels Dokumenten mit zertifiziertem, aktivem Inhalt oder auch, wenn es der rechtliche Rahmen der Experimente zulässt, durch das Ausnutzen einer Schwachstelle in den Anwendungsprogrammen und eine dadurch bedingte Code-Ausführung.
- **Simulation eines realen Wurms**: Anstatt eines abstrakten E-Mail-Wurms könnte man auch ein reales Schadprogramm auswählen und dessen Verbreitung simulieren – etwa *Waledac* oder *Mydoom*. Das Vorgehen ist jedoch das gleiche wie hier in der Arbeit beschrieben: Man muss zuerst die Verbreitungsroutinen des realen Schadprogramms analysieren, daraus ein parametrisiertes Modell erstellen, die darin gekapselten menschlichen Handlungen identifizieren und anschließend kann man dies als Input für die Simulationen verwenden.
- **Experimentelle Verbreitungsverläufe**: Neben einer Simulation könnte man die Verbreitungsverläufe auch experimentell berechnen. Machbar wäre dies, indem man mehrere native oder virtuelle Systeme aufsetzt, innerhalb derer man einen menschlichen Computerbenutzer durch entsprechende Automatisierungs-

software simuliert. Für die Automatisierung kann man etwa *Windows Script Host* [Ait01] oder *AutoIt* [BB05] verwenden. Mit Hilfe der zuletzt genannten Lösung haben wir im Rahmen eines Lehrstuhlprojektes eine Software namens *SimUser* [Int08] entwickelt. Diese erlaubt es, eine Windows-Automatisierung auch ohne große Programmierkenntnisse zu realisieren. SimUser könnte beispielsweise für die experimentelle Bestimmung der Verbreitungsverläufe verwendet werden. Aufgrund der realen Systeme und Bedingungen, sind experimentell ermittelte Verbreitungsverläufe noch realitätsnäher. Sie ermöglichen somit eine Verifikation der simulierten Verbreitung.

- **Mathematisches Modell der Verbreitungsverläufe**: Statt einer reinen simulativen oder auch experimentellen Berechnung der Verbreitungsverläufe, könnte man auch versuchen, diese durch ein mathematisches Modell darzustellen. Es ist denkbar, bestehende Modelle, wie etwa das *Analytical Active Worm Propagation* Modell von Chen, Gao und Kwiat oder das *Two-Factor Worm* Modell von Zou, Gong und Towsley, zu erweitern, so dass diese auch typische Charakteristika von Social Malcode unterstützen. Dies sind im Wesentlichen die von uns extrahierten Simulationsparameter.
- **Charakteristische Merkmale der Verbreitungsverläufe**: Ebenfalls könnte man versuchen, die Zusammenhänge zwischen den charakteristischen Merkmalen der Verbreitungsverläufe eingehender zu untersuchen. Wie am Anfang des Abschnitts 6.2 erwähnt, kann man das Maximum der Verbreitung aus anderen Merkmalen herleiten. Auch ist auffällig, dass die Dauer der Wachstumsphase bei unseren Simulationen berechenbar scheint:

$$\Delta_W \approx \frac{E(\Delta_L) + E(\Delta_R)}{2}$$

  Dabei ist Δ_W die Dauer der Wachstumsphase, $E(\Delta_L)$ der Erwartungswert der Zeitspannen zwischen dem Eintreffen und dem Lesen einer E-Mail und $E(\Delta_R)$ entspricht dem Erwartungswert der Zeitspannen zwischen der Infektion und der Reinigung eines Rechners. Ob dies nur bei den von uns durchgeführten Simulationen so ist oder ob dies ein allgemeingültiger Zusammenhang ist, müsste jedoch eingehender untersucht werden.

6.5. Zusammenfassung

In dem vorangegangenen Kapitel wurde die Verbreitung eines E-Mail-Wurms simuliert und dabei der Einfluss der verschiedenen Simulationsparameter untersucht. In diesem Kapitel wurden die erzielten Ergebnisse mit den Ergebnisse anderer Arbeiten verglichen.

Der erste Abschnitt dieses Kapitels stellt dementsprechend verwandte Arbeiten vor, die inhaltliche Parallelen zu unserer Arbeit aufweisen. Neben einer inhaltlichen Vorstellung der Arbeiten werden diese ebenfalls gegenüber der vorliegenden Arbeit abgrenzt und/oder die Gemeinsamkeiten aufgezeigt. Dabei haben wir die verwandten Arbeiten in Gruppen unterteilt, die jeweils einem bestimmten, inhaltlichen Aspekt unserer Arbeit entsprechen: Arbeiten bezüglich des Verhaltens von Computerbenutzern,

Arbeiten, die sich generell mit der Verbreitung von Schadprogrammen beschäftigen, und Arbeiten, in deren Fokus speziell benutzerabhängige Schadprogramme stehen.

Anschließend haben wir in Abschnitt 6.2 die Ergebnisse unserer Arbeit mit den Ergebnissen der verwandten Arbeiten verglichen. Dabei haben wir nicht nur einen reinen Vergleich durchgeführt, sondern auch die Unterschiede zwischen den Ergebnissen erläutert und diskutiert. Dadurch wurden spezielle Charakteristika von Social Malcode identifiziert, durch die die Verbreitung des E-Mail-Wurms auf eine typische Art und Weise beeinflusst wird. Beispielsweise kommt es durch die Zeit, in der E-Mails ungelesen im Posteingang der Empfänger liegen, zu Verzögerungen, die die Verbreitung von Social Malcode im Vergleich zur Verbreitung eines autonomen Computerwurms extrem verlangsamen. Ebenfalls haben wir drei Phasen bei der Verbreitung von Social Malcode identifiziert: Eine Start-, eine Wachstums- und eine Kontinuitätsphase. Des Weiteren haben wir gezeigt, dass es Parallelen zwischen der Hit-List eines autonomen Wurms und den Knoten gibt, die bei unserer Simulation initial eine E-Mail in ihrem Posteingang haben. Die von uns analysierte künstliche Verzögerung in der Zustellgeschwindigkeit von E-Mails ist hingegen vergleichbar mit der Dauer, die ein autonomer Computerwurm zur Infektion eines einzelnen Rechners benötigt. Beide Faktoren weisen die gleiche Auswirkung auf die Verbreitung des Schadprogramms aus. Letztendlich haben wir noch die gezielte Immunisierung unserer Arbeit mit einer Arbeit von Zou et al. verglichen, die diesen Effekt gleichermaßen bei der Verbreitung eines E-Mail-Wurms untersucht haben. Dabei ist festzuhalten, dass bei unseren Simulationen eine gezielte Immunisierung eines wesentlich kleineren Anteils der beteiligten Rechner bereits zu einem deutlich wahrnehmbaren Eindämmungseffekt führt.

Inwieweit sich kombinierte Gegenmaßnahmen auf die Verbreitung des simulierten E-Mail-Wurms auswirken, zeigt Abschnitt 6.3. Hier haben wir den Verbreitungsverlauf einer Simulation mit den Standardwerten der Simulationsparameter einem Verbreitungsverlauf gegenübergestellt, bei dessen Simulation die E-Mails nur verzögert zugestellt werden, zusätzlich zentrale Knoten immunisiert sind und Rechner nach einer Infektion immun gegen den Wurm werden können. Durch diese Gegenmaßnahmen ist der Verbreitungsverlauf wesentlich flacher. Zudem wird, im Gegensatz zu einer Simulation komplett ohne Gegenmaßnahmen, die maximale Anzahl an infizierten Rechnern auch erst sehr viel später erreicht.

Abgeschlossen wir dieses Kapitel durch einen Hinweis auf Limitierungen dieser Arbeit. Dabei steht die Tatsache im Mittelpunkt, dass die von uns durchgeführten Experimente zur Bestimmung des Benutzerverhaltens keine homogene Zielgruppe hatten. Dementsprechend ist das von uns simulierte Benutzerverhalten kein Abbild des Verhaltens einer realen Gruppe, sondern eine Kombination aus dem Verhalten unterschiedlicher Gruppen. Daraus ergeben sich einige Verbesserungsansätze, wie man das Verhalten von Computerbenutzern noch besser messen oder quantifizieren könnte. Diese sind neben den Limitierungen ebenfalls in Abschnitt 6.4 zu finden. Auch wäre es denkbar, die Verbreitungsverläufe von Social Malcode experimentell oder mathematisch zu bestimmen.

KAPITEL VII

Zusammenfassung

In diesem finalen Kapitel der Dissertation werden noch einmal alle Kapitel zusammengefasst. Es werden die wesentlichen, inhaltlichen Aspekte dieser Arbeit rekapituliert und in Zusammenhang mit den zuvor gesteckten Zielen gesetzt.

Im Fokus dieser Arbeit steht eine besondere Sorte von Schadprogrammen – nämlich Schadprogramme, die auf eine gewisse *Mithilfe* ihrer Opfer angewiesen sind, damit sie neue Rechner infizieren können. Warum gerade solche Schadprogramme eingehender analysiert werden sollten, motivierte **Kapitel 1**. Frühe Schadprogramme verbreiteten sich fast ausschließlich durch das automatische Ausnutzen von Sicherheitslücken in Betriebssystemen oder Netzdiensten. Im Laufe der Jahre wurden diese jedoch immer sicherer. Zudem wurden immer wieder neue Schutztechnologien entwickelt, wie zum Beispiel DEP oder ASLR. Autoren von Schadsoftware mussten aus diesem Grund neue Verbreitungswege für ihre Schadprogramme finden. Diese versuchen neuerdings nicht mehr Sicherheitslücken in Programmen oder Betriebssystemen auszunutzen, sondern stattdessen das oftmals schlechte Sicherheitsbewusstsein der Computerbenutzer. Durch Techniken aus dem Bereich des Social Engineering versuchen Autoren der Schadprogramme Computerbenutzer derart zu täuschen und psychologisch zu manipulieren, dass diese ohne ihr Wissen die Schadprogramme selbst zur Ausführung bringen. Beispielsweise verschicken die Autoren massenhaft E-Mails mit einem Schadprogramm im Anhang, das als Rechnung getarnt ist, oder bieten die Schadprogramme unter aktuell populären Dateinamen in Internet-Tauschbörsen an. Aufgrund der Tatsache, dass Social Engineering bei der Verbreitung dieser Schadprogramme solch einen wichtigen Faktor darstellt, haben wir diese Sorte Schadprogramme *Social Malcode* genannt.

Dass Social Malcode immer mehr Bedeutung zukommt, zeigen auch Statistiken zur aktuellen Gefahrenlage bezüglich Schadprogrammen. In der Auflistung der häufigsten Bedrohungen im Februar 2013 sind ausschließlich Schadprogramme zu finden, die bei ihrer Verbreitung einen Eingriff der potentiellen Opfer benötigen. Alle aktuellen Bedrohungen dieser Liste fallen somit in die Kategorie von Social Malcode.

Neben einer motivierenden Einleitung wird im ersten Kapitel, anhand der Struktur der vorliegenden Dissertation, ebenfalls deren inhaltlicher roter Faden erläutert. Des

Weiteren gibt das 1. Kapitel einen Überblick über die erzielten Ergebnisse der Dissertation. Dieser Überblick wird unmittelbar nach der Definition der Ziele gegeben, die sich folgendermaßen zusammenfassen lassen:

1. Der zentrale Begriff *Social Malcode* soll formalisiert werden.
2. Da das Verhalten eines Benutzers in die Verbreitung von Social Malcode involviert ist, soll als zweites Ziel auch dieses Verhalten innerhalb des Verbreitungsprozesses formal beschreibbar sein. Jede menschliche Handlung, die für die Verbreitung von Social Malcode entscheidend ist, soll in Form von Verhaltensmodellen beschreibbar sein.
3. Um später den Effekt der Handlungen auf die Verbreitung zu untersuchen, müssen diese Handlungen natürlich ebenfalls näher untersucht werden. Das dritte Ziel der Dissertation ist also, das menschliche Verhalten in Situationen zu messen oder zu quantifizieren, die zu einer Infektion mit Social Malcode führen können.
4. Durch eine abschließende Simulation, die das vierte Ziel darstellt, sollen Verbreitungsverläufe eines Social Malcode untersucht und visuell dargestellt werden. Dabei soll auch der Einfluss der einzelnen Simulationsparameter, die unter anderem das simulierte, menschliche Verhalten steuern, auf den Verbreitungsverlauf beleuchtet werden.

Bevor wir jedoch mit der Umsetzung der Ziele anfangen, stellen wir in **Kapitel 2** die Grundlagen vor, die zum weiteren Verständnis der Arbeit notwendig bzw. hilfreich sind. Social Malcode ist eine spezielle Kategorie von Schadprogrammen. Aus diesem Grund wird als erstes erklärt was ein Schadprogramm an sich ist: Schadprogramme sind ganz normale Anwendungen, die jedoch mindestens eine schadhafte Funktionalität besitzen, die so nicht von einem Computerbenutzer gewünscht ist. Unter anderem werden die Schadfunktionen eines Schadprogramms auch benutzt, um Schadprogramme in Gruppen einzuteilen. Daneben kann auch die Art und Weise, wie sich ein Schadprogramm verbreitet, genutzt werden, um diese in Klassen einzuteilen. Sechs typische Klassen von Schadprogrammen stellen wir in diesem Kapitel näher vor und stellen ihre charakteristischen Merkmale heraus. Dies sind die Klassen *Viren*, *Würmer*, *Bots*, *Trojanische Pferde*, *Spyware* und *Rootkits*

Ein wichtiger Punkt bei der Klassifizierung eines Schadprogramms als Social Malcode ist die Art und Weise, wie sich dieses Schadprogramm verbreitet, d. h. wie dieses neue Rechner infiziert. Dementsprechend widmet sich der zweite Teil des Grundlagenkapitels den Methoden und Techniken, die Schadprogramme benutzen, um weitere Rechner zu befallen. Frühe Computerwürmer machten dies noch vollständig autonom, durch das *Ausnutzen von Sicherheitslücken in den angebotenen Diensten* eines Rechners. Dadurch wurde Code auf dem fremden Rechner ausgeführt und eine neue Instanz des Wurms konnte nachgeladen werden. Allerdings wurde dieses Vorgehen, wie in der Einleitung bereits erklärt, durch zunehmende Sicherheitsmaßnahmen immer schwieriger. Autoren von Schadprogrammen versuchten deshalb, Computerbenutzer dazu zu bewegen, die Schadprogramme unwissentlich selbst zu installieren oder auszuführen. Gängige Verbreitungsmethoden nutzen dazu das *World Wide Web*, *E-Mails*, *Instant Messaging*, *Soziale Netzwerke*, *Filesharing*, *Netzfreigaben* oder auch *Wechseldatenträger*. Bei diesen Verbreitungsmethoden ist jeweils eine Benutzerinter-

aktion für eine erfolgreiche Infektion eines neuen Systems notwendig. Beispielsweise muss ein E-Mail-Empfänger bei der Verbreitung eines E-Mail-Wurms einem Verweises folgen oder einen Dateianhang herunterladen und ausführen. Auch bei der Verbreitung über Filesharing-Programme müssen die Anwender dazu gebracht werden, bestimmte Dateien herunterzuladen und auszuführen.

Im Normalfall hilft natürlich kein Computerbenutzer freiwillig dabei, seinen eigenen Rechner mit einem Schadprogramm zu infizieren. Damit ein potentielles Opfer dennoch die vom Schadprogrammautor gewünschte Benutzerinteraktion durchführt, versucht er die potentiellen Opfer durch Täuschung und Manipulation dazu zu bewegen. Am besten so, dass die Benutzer gar nicht bemerken, dass sie gerade dazu beigetragen haben, ein Schadprogramm zu installieren oder auszuführen. Diesen Vorgang bezeichnet man als *Social Engineering* und ist folglich Thema des dritten Abschnitts des Grundlagenkapitels.

Es gibt zahlreiche Definitionsansätze, von denen einige vorgestellt werden. Generell sind die Motive eines Social Engineers vielfältig. Neben einem reinen finanziellen Interesse des Angreifers können dies auch ein persönliches Selbstinteresse, z. B. bedingt durch Eifersucht, oder auch Rache sein. Die Ziele eines Social Engineers bilden all jene Dinge, die diese Motive befriedigen. Dabei ist es nicht unbedingt notwendig, dass er diese Ziele direkt erreicht, sondern er kann sie auch über mehrere Zwischenziele erreichen. Um dies zu verwirklichen, kann sich ein Angreifer verschiedener Methoden bedienen: Zum Beispiel kann er die Methode des *Dumpster Diving* benutzen, bei dem der Abfall einer Zielperson oder -organisation nach hilfreichen Dingen durchsucht wird, oder das *Shoulder Surfing*, bei dem heimlich vertrauliche Informationen mitgelesen werden. Auch *Phishing* ist eine Methode des Social Engineering. Dabei wird versucht, über nachgebaute Internetseiten vertrauliche Daten der Opfer zu stehlen. Die dafür nachgebauten Seiten sehen den originalen Internetseiten zum Verwechseln ähnlich. In der Literatur finden sich etliche psychologische Erklärungsversuche dafür, warum Social Engineering erfolgreich sein kann oder erfolgreich ist. Dabei werden immer wieder menschliche „Schwächen“ als Gründe genannt. Dies können zum Beispiel das grundsätzliche *Vertrauen* in andere Menschen, *Autorität* in Verbindung mit *Angst*, *Pflichtgefühl* gegenüber seinen Mitmenschen oder aber auch ein allgemeines Verlangen nach *Konformität* im täglichen Leben sein.

Nachdem alle relevanten Grundlagen der Arbeit erörtert wurden, befasst sich der Inhalt des **3. Kapitels** damit, eine formale Definition des Begriffs *Social Malcode* herzuleiten. Dies ist auch gleichzeitig das erste Arbeitsziel der Dissertation. Wie in dem Grundlagenkapitel bereits deutlich wurde, sind Techniken des Social Engineering immer mit einer gewissen Art von Täuschung und Manipulation verbunden. Ein Angreifer hat also die böswillige Absicht, seine Opfer zur Installation eines Schadprogramms zu bewegen. Jedoch ist es schwierig, die Absichten eines Angreifers zu formalisieren und in die Definition von Social Malcode einfließen zu lassen. Um diesem Problem zu entgehen, lautet eine erste Definition folgendermaßen: Social Malcode ist Code, der einem Angreifer nicht-autorisierten Zugriff auf eine oder mehrere Ressourcen eines Opfers ermöglicht, und der zur Weiterverbreitung eine Benutzerinteraktion erfordert. Durch die erste Bedingung ist der schadhafte Aspekt von Social Malcode ausgedrückt. Die zweite Bedingung hingegen bewirkt, dass die Absicht des Angreifers nicht forma-

lisiert werden muss, sondern nur dessen Ziel, die Opfer dazu zu bewegen, aktiv bei der Verbreitung des Schadprogramms mitzuwirken.

Nach dieser Definition werden einige Schadprogramme als Beispiel und Gegenbeispiel für Social Malcode angeführt. Alle Beispiele haben gemeinsam, dass für eine erfolgreiche Verbreitung der entsprechenden Schadprogramme das Eingreifen der potentiellen Opfer notwendig ist – etwa durch das Ausführen eines Dateianhangs oder dem Besuch einer präparierten Internetseite. Dieses Eingreifen entspricht genau der durch die Definition geforderten Benutzerinteraktion. Die angeführten Gegenbeispiele bestehen hingegen ausschließlich aus Schadprogrammen, die sich autonom verbreiten. In den meisten Fällen handelt es sich dabei um Computerwürmer, die sich durch das vollautomatische Ausnutzen einer Schwachstelle verbreiten. Sie benötigen also keinerlei menschliche Interaktion, um neue Rechner zu kompromittieren, und fallen somit auch nicht unter den Begriff Social Malcode.

Das bereits beschriebene Phishing ist, aufgrund einer optisch perfekt nachgebauten Internetseite zum Stehlen von Zugangsdaten, intuitiv als Paradebeispiel eines Social-Engineering-Angriffs anzusehen. Jedoch bereitet gerade Phishing bei der bisherigen Definition von Social Malcode noch Probleme, da sich bei solch einem Angriff kein Schadprogramm verbreitet. Aus diesem Grund diskutieren wir im nächsten Abschnitt noch bestehende Probleme und Schwächen der Definition. Im Laufe der Diskussion wird ersichtlich, dass zum Beispiel die Verbreitungs- und Schadroutinen eines Schadprogramms nicht immer klar voneinander abzugrenzen sind oder dass etwa nicht ersichtlich ist, ob Social Malcode nach erster Definition immer noch als Social Malcode bezeichnet werden kann, wenn sich dieser als Payload eines autonomen Computerwurms verbreitet.

Somit ist klar, dass die bisherige Definition noch weiter präzisiert und überarbeitet werden muss. Nachdem zuerst die Begriffe *Schadcode* und *Verbreitungsroutine* definiert wurden, lautet die endgültige Definition von Social Malcode nun folgendermaßen: Als Social Malcode wird Schadcode bezeichnet, der sich entweder nicht verbreitet oder der sich verbreitet und mindestens eine Verbreitungsroutine enthält, in der sich eine *Herausforderung*, eine *Benutzerinteraktion* und eine *Code-Ausführung* kausal bedingen. Alle Schadprogramme, deren erfolgreiche Verbreitung auf einer provozierten Benutzerinteraktion beruhen, sind dank der geforderten Sequenz der Zustandsübergänge, definitionsgemäß immer noch als Social Malcode zu bezeichnen. Zusätzlich fallen nun aber auch Schadcodes unter den Begriff Social Malcode, die sich nicht verbreiten – wie etwa eine Phishing-Seite. Die Ungenauigkeiten der ersten Definition werden vor allem dadurch beseitigt, dass die zweite Definition auf Modellen von Verbreitungsroutinen beruht. In diesen Modellen wird das Verhalten von Benutzern gekapselt und sie stellen somit die Erfüllung des zweiten Arbeitsziels der vorliegenden Arbeit dar.

Die in der zweiten Definition geforderte Benutzerinteraktion ist eine menschliche Handlung. Entscheidend für die Verbreitung von Social Malcode ist also das Verhalten der potentiellen Opfer in Situationen, die möglicherweise zu einer Infektion führen. Wir müssen also wissen, wie beispielsweise Empfänger einer E-Mail reagieren, ob sie diese lesen oder ob sie sogar einem möglichen Verweis innerhalb der E-Mail folgen. Da wir im weiteren Verlauf der Arbeit die Verbreitung von Social Malcode auch simulieren möchten, ist es notwendig, das Verhalten der beteiligten Personen modellieren zu können. Aus diesem Grund stellen wir in **Kapitel 4** Experimente und Techniken

vor, mit deren Hilfe das Verhalten einer Person in genau solchen Situationen, die zur Infektion mit Social Malcode führen können, erfasst und quantifiziert werden kann. Dies wiederum ist das dritte Arbeitsziel der Dissertation.

Das klassische Instrument der empirischen Sozialwissenschaften zum Quantifizieren von Verhaltensweisen sind Umfragen und Interviews. In deren Zusammenhang existieren jedoch einige Probleme, die es erschweren, den Wahrheitsgehaltes der Antworten zu beurteilen. Zum Beispiel nehmen die *soziale Erwünschtheit*, der *Reihenfolgeneffekt* und die *Formulierung der Fragen* oftmals einen verfälschenden Einfluss auf die Antworten der befragten Personen.

Um dennoch das Verhalten der Benutzer analysieren zu können, haben wir in Zusammenarbeit mit einem Unternehmen aus der Branche der IT-Sicherheit Spam-Experimente durchgeführt. Diese haben den Vorteil, dass sie die Probleme und Schwierigkeiten von Interviews und Umfragen dadurch umgehen, dass die Benutzer gar nicht wissen, dass ihr Verhalten untersucht wird. Natürlich wurden diese Experimente nur nach Beauftragung der Kunden des kooperierenden Unternehmens durchgeführt und waren somit rechtlich abgesichert. Im Rahmen dieser Spam-Experimente wurden E-Mails an die Mitarbeiter der beauftragenden Firmen geschickt. Über einen Rückkanal wurde anschließend gemessen, wie hoch der Anteil der Mitarbeiter ist, die einem Verweis folgen, die einen Dateianhang herunterladen und ausführen, die einen angeblichen Sicherheitspatch ausführen und auch wie hoch der Anteil der Mitarbeiter ist, die Daten auf einer von uns erstellten Phishing-Seite eingeben.

Neben den durch die Spam-Experimente gemessenen Raten, spielen bei der Verbreitung von Social Malcode natürlich auch noch andere Faktoren eine entscheidende Rolle. So ist es natürlich ein Unterschied, ob die Empfänger der E-Mails diese recht zügig lesen oder ob die E-Mails vor dem Lesen noch einige Tage unbemerkt im Posteingang der Empfänger liegen. In erstem Fall verbreitet sich das entsprechende Schadprogramm natürlich wesentlich schneller als im zweiten Fall. Somit müssen wir auch die Zeitspannen zwischen dem Eintreffen und dem Lesen einer E-Mail erfassen. Dafür haben wir ein Python-Skript geschrieben, das diese Zeitspannen direkt am Mail-Server der Universität Mannheim auslesen kann. Voraussetzung dafür war jedoch, dass wir das Skript nur auf die Mailboxen derjenigen Benutzer anwenden, die zuvor eine entsprechende Einverständniserklärung unterschrieben haben. Dabei ist herausgekommen, dass die Verteilung der ausgelesenen Zeitspannen einem Potenzgesetz folgt. Der Großteil der Benutzer liest neue E-Mails also recht schnell (innerhalb einer Stunde) und einige wenige Benutzer lesen neue E-Mails erst recht spät nach dem Eintreffen im Posteingang.

In der Regel ist ein infiziertes System natürlich nicht für immer mit einem Schadprogramm befallen. Sobald die Benutzer die Infektion bemerken, versuchen sie Gegenmaßnahmen einzuleiten und das Schadprogramm von dem betroffenen Rechner zu entfernen. Im schlimmsten Fall müssen sie dafür das entsprechende System neu aufsetzen. Solch eine Rechnersäuberung bzw. -reinigung möchten wir ebenfalls in unsere Simulationen integrieren. Um die zeitlichen Aspekte einer Rechnerreinigung zu modellieren, haben wir gestohlene Daten zweier großer Keylogger-Familien analysiert. Innerhalb dieser Daten ist jeder Datensatz mit einer Opfer-ID und einem Zeitstempel gekennzeichnet. Als Dauer einer Infektion haben wir die Zeitspanne zwischen dem

ersten und dem letzten Zeitstempel eines Opfers festgelegt. Bei den von uns auf diese Weise analysierten Daten waren die gemessenen Zeitspannen exponentialverteilt.

Neben dem Verhalten der potentiellen Opfer ist auch das Verhalten eines Angreifers für die Verbreitung von Social Malcode interessant. Welche Angriffsvektoren werden zum Beispiel genutzt und wie häufig finden Angriffe statt? Um dies eingehender zu untersuchen, wollten wir auf Spamtraps zurückgreifen, die wir für ein von uns zuvor durchgeführtes Projekt aufgesetzt hatten. Aufgrund eines Hardware-Defektes standen diese aber zum benötigten Zeitpunkt nicht mehr zur Verfügung. Freundlicherweise konnten wir stattdessen einen Spam-Korpus aus den USA analysieren. Dieser besteht aus über 17 Millionen Spam-Mails, die in einem Zeitraum von vier Wochen an 12 unterschiedlichen Domains gesammelt wurden. Insgesamt sind in dem Korpus mehr als eine Million unterschiedliche Empfänger adressiert. Innerhalb der Mails wurden fast $5{,}5$ Millionen unterschiedliche Verweise verwendet und knapp über $62\,000$ unterschiedliche Dateianhänge eingebettet. Da der Anteil der Mails mit einem Verweis wesentlich höher als der Anteil der Mails mit einem Dateianhang ist, haben wir uns dazu entschlossen, bei unseren Simulationen auch genau diesen Angriffsvektor zu benutzen.

Das Ergebnis all dieser Experimente und Skripte sind Kennzahlen, die das Verhalten von Computerbenutzern in Situationen beschreiben, die zu einer Infektion mit Social Malcode führen können. Diese Ergebnisse werden nun wiederum genutzt, um die Verbreitung von Social Malcode mit Hilfe einer Simulation darzustellen und näher zu untersuchen. Diese ist das vierte Arbeitsziel der Dissertation und gleichzeitig Inhalt von **Kapitel 5**.

Zur Simulation von Social Malcode haben wir *OMNeT++* verwendet. Dies ist ein Simulations-Rahmenwerk, d. h. es dient nicht direkt zur Simulation eines bestimmten, realen Sachverhalts – wie zum Beispiel der Verbreitung eines Schadprogramms. Vielmehr stellt es Mittel und Werkzeuge zur Verfügung, um die unterschiedlichsten Simulationen zu erstellen. Beispielsweise bietet OMNeT++ eine eigene Auszeichnungssprache, um die Topologie einer Simulation festzulegen. Das Verhalten einer Simulation wird in C++-Code spezifiziert.

Mittels OMNeT++ haben wir eine Simulation erstellt, die die Verbreitung von Social Malcode nachbildet. Damit die Ergebnisse der einzelnen Simulationsdurchläufe besser zu interpretieren sind, haben wir zuvor jedoch die Rahmenbedingungen der Simulation genau abgesteckt – also welche Modell-Struktur wir verwendet haben, wie das Verhalten der einzelnen Komponenten ist und welche Sorte von Social Malcode wir genau simulieren möchten. Folgende drei Beschreibungen bilden die Rahmenbedingungen unserer Simulationen:

- **E-Mail-Wurm**: Zur Analyse der Verbreitung von Social Malcode haben wir ein Schadprogramm simuliert, das sich über das Medium E-Mail verbreitet. Nach einer Infektion durchsucht der E-Mail-Wurm den befallene Rechner nach E-Mail-Adressen. An jede gefundene E-Mail-Adresse verschickt der Wurm eine E-Mail, in der sich ein Verweis auf eine Internetseite befindet. Besucht ein Empfänger der E-Mails diese Seite, wird durch einen Drive-By-Download eine neue Instanz des Wurms auf dessen Rechner geladen. Voraussetzung für eine Infektion ist, dass der Empfänger die Internetseite mit einem entsprechend ver-

wundbaren Browser besucht. Anschließend beginnt die Verbreitungsroutine auf dem neu infizierten Rechner wieder von vorn.

- **Modell-Struktur**: Nicht nur die Funktionsweise des simulierten Schadprogramms hat einen Einfluss auf dessen Verbreitung. Auch die Modell-Struktur, in unserem Fall also die Struktur des zugrundeliegenden Netzes, hat Einfluss auf die Verbreitung des Wurms. Für die vorliegende Arbeit haben wir ein E-Mail-Netz simuliert, das aus $50\,000$ Rechnern besteht. Zur Generierung des Netzes haben wir einen Algorithmus von Holme und Kim verwendet. Dieser erweitert ein bekanntes Modell von Barabási und Albert um einen zusätzlichen Schritt, der dafür sorgt, dass in dem generierten Netz Cluster entstehen. Dies sind Teilbereiche des Netzes, deren Knoten untereinander sehr stark miteinander vernetzt sind, zu Knoten anderer Cluster aber nur recht wenige Kanten aufweisen. Die Verteilung aller Verzweigungsgrade des auf diese Weise generierten Netzes genügt einem Potenzgesetz.
- **Modell-Verhalten**: Das Modell-Verhalten haben wir anhand zweier Programmablaufpläne beschrieben. Diese beschreiben zum einen das globale Verhalten, also in welcher Art und Weise die einzelnen Rechner des Simulationsnetzes miteinander interagieren, und zum anderen das lokale Verhalten der Rechner. Dies Rechner-lokale Simulationsverhalten orientiert sich stark an der Funktionsweise des zuvor beschriebenen E-Mail-Wurms.

Insbesondere fließen die Ergebnisse des 4. Kapitels in dieses Rechner-lokale Verhalten ein und entsprechen somit dem Verhalten der simulierten Computerbenutzer. Für jedes zuvor beschriebene Experiment zur Analyse des Benutzerverhaltens existiert in der Simulation mindestens ein Simulationsparameter. Beispielsweise enthält ein Parameter die Wahrscheinlichkeit, mit der Benutzer dem Verweis der E-Mail folgen, und über einen anderen Parameter lässt sich die Rate der Exponentialverteilung steuern, die die Zeitspannen zwischen Infektion und Reinigung eines Rechners beschreibt. Mit Hilfe dieser Parameter kann also der Verlauf der Simulation gesteuert und kontrolliert werden. Bevor die Simulationsergebnisse beschrieben werden, haben wir in Abschnitt 5.2.4 nochmals alle Simulationsparameter in Form eines tabellarischen Überblicks zusammengefasst. In dieser Tabelle haben wir jedem einzelnen Parameter ebenfalls einen Standardwert zugewiesen. Dieser entspricht exakt den Ergebnissen des entsprechenden Experiments zur Bestimmung des Benutzerverhaltens.

Das Hauptaugenmerk der Simulationen liegt auf der Untersuchung des Einflusses der verschiedenen Parameter. Um dieses Ziel zu erreichen, haben wir für jeden Parameter mehrere Simulationen durchgeführt. Dabei haben wir den zu untersuchenden Parameter sukzessive mit verschiedenen Werten belegt, während den anderen Parametern ihr fester Standardwert zugewiesen ist. Für jede Belegung des zu untersuchenden Parameters haben wir 40 Simulationsdurchläufe, jeweils mit einem anderen Startwert des Zufallsgenerators, durchgeführt und daraus den durchschnittlichen Verbreitungsverlauf berechnet. Anschließend haben wir alle so berechneten Verbreitungsverläufe eines Parameters entsprechend dessen Belegung nebeneinander angeordnet und dadurch eine 3-dimensionale Darstellungsform der Verläufe erhalten. Anhand dieser 3-dimensionalen Darstellungen kann man den Einfluss der acht untersuchten Parameter auf den Verbreitungsverlauf des simulierten E-Mail-Wurms sehr schnell visuell erfassen.

Der erste Parameter, den wir auf diese Weise untersucht haben, enthält die *Anzahl der Rechner, die initial eine Spam-Mail in ihrem Posteingang haben*. Wenig überraschend verbreitet sich der simulierte E-Mail-Wurm besser, je höher die Anzahl dieser Startknoten oder -rechner gewählt wird. Anhand der erzeugten Verbreitungsverläufe sieht man, dass die Anzahl der maximal im Verlauf der Simulation infizierten Systeme proportional von der Anzahl der Rechner abhängt, die solch eine initiale Spam-Mail in ihrem Posteingang haben. Auch ist zu erkennen, dass die Verbreitungsverläufe mit einer hohen Anzahl an Startknoten schneller ansteigen als die Verläufe, bei denen nur wenige Rechner mit einer initialen Spam-Mail existieren.

Die nächsten beiden Parameter, deren Einfluss von uns untersucht wurde, enthalten zum einen die *Wahrscheinlichkeit, dass eine neue E-Mail gelesen wird*, und zum anderen die *Wahrscheinlichkeit, dass ein Benutzer dem in der E-Mail enthaltenen Verweis folgt*, wenn er diese liest. Bei beiden Parametern steigt mit der zugewiesenen Wahrscheinlichkeit auch die Anzahl der durch den Wurm infizierten Rechner. Bei dem zweiten Parameter, der die Wahrscheinlichkeit enthält, dass auf den Verweise geklickt wird, ist dieser Anstieg jedoch wesentlich schneller zu beobachten. Der Grund dafür ist der höhere Standardwert des Parameters, der die Lese-Wahrscheinlichkeit enthält.

Bei der Analyse der *Zeitspannen zwischen dem Eintreffen einer E-Mail im Posteingang und dem Lesen durch den Benutzer* enthält der entsprechende Simulationsparameter den Exponenten der Verteilung nach einem Potenzgesetz. Bei steigenden Werten dieses Parameters lesen die simulierten Benutzer schneller ihre E-Mails. Entsprechend lässt sich auch hier mit steigenden Werten des Parameters eine steigende Anzahl an infizierten Rechnern beobachten. Zudem ist in den Verbreitungsverläufen eine auffällige Senke zu erkennen. Diese ergibt sich dadurch, dass in den entsprechenden Wertebereichen des Parameters Rechner teilweise schneller bereinigt als infiziert werden.

Des Weiteren haben wir untersucht, inwieweit sich eine *künstliche Verzögerung in der Zustellgeschwindigkeit der E-Mails* auf den Verbreitungsverlauf des E-Mail-Wurms auswirkt. Dabei zeigt sich, dass sich durch eine langsamere Zustellung der E-Mails ebenfalls eine langsamere Verbreitung des Wurms ergibt. Die Zeit zwischen dem Ausbruch des Wurms und dem Zeitpunkt, ab dem er anfängt, sich rasant zu verbreiten, hängt proportional mit der Verzögerung der E-Mails zusammen. In den Verbreitungsverläufen ist dies an einem fast linearen Verlauf dieses Zeitpunktes, bezogen auf sämtliche Parameterbelegungen, zu erkennen. Ebenfalls ist ein Flattern der Verläufe bei langen Verzögerungen zu beobachten.

Die *Zeitspannen zwischen der Infektion und der Säuberung eines Rechners* sind exponentialverteilt. Folglich enthält der entsprechende Simulationsparameter die Rate, die diese Exponentialverteilung beschreibt. Die Analysen haben ergeben, dass mit steigenden Raten die Verbreitung des Wurms eingedämmt wird. Dies liegt daran, dass bei höheren Raten der Exponentialverteilung die befallenen Rechner auch schneller bereinigt werden. Interessanterweise spiegelt sich die Exponentialverteilung in einem Querschnitt durch den Verbreitungsverlauf nach 600 Stunden wider.

Die letzten beiden Parameter, die wir untersucht haben, steuern beide eine mögliche Immunität der Rechner gegen den simulierten E-Mail-Wurm. Durch den ersten Parameter lässt sich die *Wahrscheinlichkeit, mit der ein Rechner nach einer Wurminfektion*

immun gegen diesen werden kann, festlegen. Mit einer steigenden Wahrscheinlichkeit sinkt natürlich die Anzahl der Rechner, die aktiv an der Wurmverbreitung teilnehmen, da diese wesentlich schneller immun gegen den Wurm sind. Entsprechend fällt die Anzahl der infizierten Rechner. Aber auch bei einem festen Wert des Parameters, sinkt die Anzahl der nicht immunen Rechner im Verlauf der Simulation: Bei mehrmaliger Infektion eines Rechners wird immer wieder neu entschieden, ob dieser immun gegen den Wurm wird. Also sinkt die Anzahl der infizierten Rechner sowohl mit einem steigenden Parameterwert als auch unabhängig von diesem im Verlauf der Simulation. Somit ergibt sich wieder eine Senke in dem 3-dimensionalen Verbreitungsverlauf – diesmal jedoch eine wesentlich ausgeprägtere als dies bei der Analyse der Zeitspannen zwischen dem Eintreffen und dem Lesen einer E-Mail der Fall war.

Anhand des letzten Simulationsparameters lässt sich der Effekt einer gezielten Härtung oder Immunisierung zentraler Knoten des Netzes analysieren. Dabei enthält der Parameter einen minimalen Verzweigungsgrad. Jeder Knoten des Netzes, dessen Verzweigungsgrad mindestens so hoch ist wie der aktuelle Parameterwert gilt ab Beginn der Simulation als immun gegen den Wurm. Das Ergebnis der Untersuchung ist, dass eine gezielte Immunisierung zentraler Knoten des E-Mail-Netzes nur in einem bestimmten Umfang Sinn macht. Härtet man zu wenige zentrale Knoten, verbreitet sich der Wurm dennoch relativ gut. Härtet man viele Knoten, dämmt dies zwar sehr gut die Verbreitung des Wurms ein, jedoch ist der entsprechende Aufwand, den man durch die vielen Härtungen betreiben muss, extrem hoch. Unabhängig davon kann eine Immunisierung eines Großteils der Knoten nicht mehr als gezielt bezeichnet werden.

Um die Verbreitungsverläufe und die beobachteten Effekte der Simulationsparameter besser einordnen zu können, vergleichen wir diese in **Kapitel 6** mit anderen Arbeiten auf diesem Themengebiet. Dabei geben wir in einem ersten Abschnitt einen Überblick über verwandte Arbeiten, also Arbeiten, die inhaltliche Parallelen zu der vorliegenden Arbeit aufweisen. Neben einer kurzen inhaltlichen Beschreibung dieser Arbeiten, stellen wir auch Gemeinsamkeiten oder Unterschiede zur vorliegenden Arbeit heraus. Die vorgestellten Arbeiten haben wir in Gruppen eingeteilt, die jeweils einem inhaltlichen Aspekt unserer Arbeit entsprechen. Diese Gruppen enthalten Arbeiten, die ebenfalls das Verhalten von Computerbenutzern untersuchen, Arbeiten, die sich mit der Verbreitung von Schadprogrammen beschäftigen, und Arbeiten, in deren Fokus speziell benutzerabhängige Schadprogramme stehen.

Nach der Vorstellung und einer ersten inhaltlichen Abgrenzung der Arbeiten, untersucht der zweite Abschnitt die Gemeinsamkeiten und Unterschiede zwischen den Arbeiten genauer. Dieser Vergleich konzentriert sich hauptsächlich auf die Verbreitungsverläufe, die den Verbreitungsverläufen der anderen Arbeiten gegenübergestellt werden. Dadurch wurden spezielle Charakteristika von Social Malcode identifiziert, durch die die Verbreitung des E-Mail-Wurms auf eine typische Art und Weise beeinflusst wird. Ein Beispiel hierfür sind die Verzögerungszeiten, die bei dem E-Mail-Wurm dadurch entstehen, dass neu zugestellte E-Mails erst eine gewisse Zeit ungelesen im Posteingang der Empfänger liegen, bevor sie zur Verbreitung des Wurms beitragen können. Somit verbreitet sich der von uns simulierte E-Mail-Wurm auch wesentlich langsamer als ein autonomer Computerwurm wie zum Beispiel Code Red. Neben diesem zeitlichen Unterschied in der Verbreitung existieren aber auch Gemeinsamkeiten bezüglich der Verbreitung des E-Mail-Wurms und eines autonomen Computerwurms: So ist die

grundsätzliche Form der Verbreitungsverläufe gleich. Beginnend mit einer recht langsamen Verbreitung (Startphase) folgt eine Periode, in der die Anzahl der infizierten Rechner nahezu exponentiell ansteigt (Wachstumsphase). Dieser Anstieg wird dann allmählich immer flacher, bis sich eine Art Gleichgewicht zwischen Rechnerinfektionen und -säuberungen einstellt. Danach bleibt das Niveau des Verbreitungsverlaufs relativ konstant (Kontinuitätsphase).

Genauso ist der Einfluss einer Hit-List auf die Verbreitung eines autonomen Wurms vergleichbar mit dem Einfluss, den die Anzahl der Rechner, die eine initiale Spam-Mail in ihrem Posteingang haben, auf die Verbreitung des simulierten E-Mail-Wurms nimmt. Beide Größen bestimmen den Zeitpunkt, ab dem der Verbreitungsverlauf von der Start- in die Wachstumsphase übergeht. Die von uns analysierte künstliche Verzögerung in der Zustellgeschwindigkeit von E-Mails ist ebenfalls mit einem Phänomen eines autonomen Computerwurms vergleichbar: mit der Dauer, die ein solcher Wurm zur Infektion eines einzelnen Rechners benötigt. Beide Faktoren weisen die gleiche Auswirkung auf die Verbreitung des Schadprogramms aus. Als letztes haben wir die gezielte Immunisierung zentraler Knoten des Simulationsnetzes mit einer anderen Arbeit verglichen, in der dies auch analysiert wurde. Dabei kam heraus, dass bei unseren Simulationen bereits eine gezielte Immunisierung eines wesentlich kleineren Anteils der beteiligten Rechner ausreicht, um einen wahrnehmbaren Eindämmungseffekt zu erzielen.

Ebenfalls im Zuge der Diskussion haben wir den Effekt von kombinierten Gegenmaßnahmen auf die Verbreitung des simulierten E-Mail-Wurms untersucht. Dafür haben wir den Verbreitungsverlauf einer Simulation, bei der alle Simulationsparameter mit ihren Standardwerten belegt waren, einem Verbreitungsverlauf gegenübergestellt, bei dessen Simulation folgende Gegenmaßnahmen getroffen wurden: Alle Rechner mit einer Adressbuchgröße von mindestens 50 Kontakten sind ab Beginn der Simulation immun gegen den Wurm, Rechner werden nach einer Infektion mit einer Wahrscheinlichkeit von 0,1 immun gegen den Wurm und die Zustellung der E-Mails dauert drei Stunden. Es zeigt sich, dass durch diese Gegenmaßnahmen die Verbreitung des E-Mail-Wurms stark eingedämmt werden kann. Durch die verzögerte Zustellung der E-Mails verbreitet sich der Wurm zudem wesentlich langsamer.

Das letzte Thema der Diskussion besteht aus den Limitierungen und einem Ausblick auf weiterführende Arbeiten. Die größte Schwierigkeit bei der Erstellung der Arbeit war das zuverlässige Messen der Verhaltensweise von menschlichen Computerbenutzern. Wir haben dafür verschiedene Experimente durchgeführt und Skripte geschrieben, die uns beim Erfassen der menschlichen Verhaltensweisen unterstützen. Jedoch hatten diese alle eine unterschiedliche Zielgruppe. Dementsprechend ist das simulierte Verhalten auch eine Kombination des Verhaltens einzelner, unterschiedlicher Gruppen von Computerbenutzern. Daraus ergeben sich einige Verbesserungsansätze, wie man das Verhalten von Computerbenutzern noch besser messen oder quantifizieren könnte. Nichtsdestotrotz haben die von erzielten Ergebnisse ausgereicht, um den Einfluss der einzelnen Simulationsparameter zu untersuchen und die Verbreitung von Social Malcode im Detail zu beleuchten.

Literaturverzeichnis

[8co11] 8com GmbH & Co. KG. Internetseite der 8com GmbH & Co. KG, Dezember 2011. Internet: *http://www.8com.de/* (Zugriffsdatum: 26.12.2011).

[AB02] Réka Albert und Albert-László Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74:47–97, Jan 2002.

[ACS10] Sherly Abraham und InduShobha Chengalur-Smith. An overview of social engineering malware: Trends, tactics, and implications. *Technology in Society*, Juli 2010.

[Ait01] Peter G. Aitken. *Windows Script Host*. Prentice Hall Professional Technical Reference, 2001.

[All06] Malcolm Allen. Social Engineering: A Means To Violate A Computer System. Technischer Bericht, SANS Institute, Juni 2006. InfoSec Reading Room.

[ANFM05] Ant Allan, Kristen Noakes-Fry und Rich Mogull. Management Update: How Businesses Can Defend against Social Engineering Attacks. White paper, Gartner Inc., März 2005.

[Ano97] Anonymer Autor „bernz“. The Complete Social Engineering FAQ!, Januar 1997. Internet: *http://www.morehouse.org/hin/blckcrwl/hack/soceng.txt* (Zugriffsdatum: 17.05.11).

[App09] Scott D. Applegate, Major. Social engineering: Hacking the wetware! *Information Security Journal: A Global Perspective*, 18 Issue 1:40–46, Januar 2009. Taylor & Francis, Inc.

[ASI11] Adobe Systems Incorporated. Adobe – Pagemaker 7: Homepage, 2011. Internet: *http://www.adobe.com/de/products/pagemaker* (Zugriffsdateum: 17.05.11).

[BA99] Albert-László Barabási und Réka Albert. Emergence of Scaling in Random Networks. *Science (New York, N.Y.)*, 286:509–512, Oktober 1999.

[Bac09] Daniel Bachfeld. Schädliche Werbebanner bei der New York Times. heise Security (Internet), September 2009. *http://www.heise.de/security/meldung/Schaedliche-Werbebanner-bei-der-New-York-Times-788720.html* (Zugriffsdatum; 15.01.2011).

[Ban01] Karen J Bannan. Internet World, Januar 2001.

[Bar13] Aram Bartholl. Dead Drops – Un-cloud your files in cement! 'Dead Drops' is an anonymous, offline, peer to peer file-sharing network in public space. Internet, Januar 2013. *http://deaddrops.com/* (Zugriffsdatum: 22.01.13).

[BB05] Jason Brand und Jeff Balvanz. Automation is a breeze with autoit. In Cynthia A. Murnan, Kelly Wainwright und Chris Jones, editors, *SIGUCCS*, Seiten 12–15. ACM, 2005.

[BBvH+10] Beat Balzlii, Matthias Bartsch, Konstantin von Hammerstein, Dietmar Hipp, Christian Reiermann, Marcel Rosenbach, René Pfister und Barbara Schmid. Steuerbetrug: Die geheimen Eichkater. *Der Spiegel*, 6, 2010.

[BCF09] Jonell Baltazar, Joey Costoya und Ryan Flores. Infiltrating WALEDAC Botnet's Covert Operations: Effective Social Engineering, Encrypted HTTP2P Communications, and Fast-Fluxing Network. White paper, Trend Micro, June 2009. http://us.trendmicro.com/imperia/md/content/us/pdf/threats/securitylibrary/infiltrating_the_waledac_botnet_v2.pdf (Zugriffsdatum: 25.11.2010).

[BCJ+05] Michael Bailey, Evan Cooke, Farnam Jahanian, David Watson und Jose Nazario. The blaster worm: Then and now. *IEEE Security and Privacy*, 3:26–31, 2005.

[BEFF10] Petra Bornhöft, Sebastian Erb, Markus Feldenkirchen und Katharina Fuhrin. Datenschutz: Unter Ahnungslosen. *Der Spiegel*, 33, 2010.

[Ben01] Colin J. Bennett. Cookies, web bugs, webcams and cue cats: Patterns of surveillance on the world wide web. *Ethics and Information Technology*, 3:195–208, 2001.

[BGLL] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte und Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008.

[BHJ09] Mathieu Bastian, Sebastien Heymann und Mathieu Jacomy. Gephi: An open source software for exploring and manipulating networks. *International AAAI Conference on Weblogs and Social Media*, 2009.

[BI11] Bebo, Inc. Bebo Homepage. Internet, Januar 2011. *http://www.bebo.com/* (Zugriffsdatum: 28.01.11).

[Bis00] Matt Bishop. *Analysis of the ILOVEYOU Worm*, Mai 2000. Department of Computer Science, University of California at Davis.

[Blo04] Michel Blomgren. *Introduction to Shellcoding: How to exploit buffer overflows*. tigerteam.se, 2004.

[BM10] K. Beaver und S. McClure. *Hacking For Dummies*. For Dummies. John Wiley & Sons, 2010.

[Bra83] N.M. Bradburn. *Response effects*. Academic Press, New York, 1983.

[Bra05] Matthew Braverman. Win32/blaster: A case study fron microsoft's perspective. In *Proceedings of the 15th Virus Bulletin International Conference*, Seiten 200–205, 2005.

[Bro10] Joshua Brower. Which Disney© Princess are YOU? – (Web 2.0) Social Engineering on Social Networks. Technischer Bericht, SANS Institute, März 2010. InfoSec Reading Room.

[Bur91] William Kent Burtner. Hidden pressures. *Notre Dame Magazine*, Winter 1991-92, Seiten 29–32, 1991.

[bwG10] bwGRiD (http://www.bw-grid.de/). Member of the German D-Grid initiative, funded by the Ministry of Education and Research (Bundesministerium für Bildung und Forschung) and the Ministry for Science, Research and Arts Baden-Wuerttemberg (Ministerium für Wissenschaft, Forschung und Kunst Baden-Württemberg). Technischer Bericht, Universities of Baden-Württemberg, 2007-2010.

[Cam09] Camarillo, G. and IAB. *Peer-to-Peer (P2P) Architecture: Definition, Taxonomies, Examples, and Applicability*. RFC 5694 (Informational), November 2009. *http://www.ietf.org/rfc/rfc5694.txt*.

[Car11a] Carnegie Mellon University. Project Cyrus, Juli 2011. Internet: *http://cyrusimap.web.cmu.edu/* (Zugriffsdatum: 01.07.2011).

[Car11b] Carnegie Mellon University. Project Cyrus: Dokumentation – Database Formats, 2011. Internet: *http://www.cyrusimap.org/docs/cyrus-imapd/2.4.8/internal/database-formats.php* (Zugriffsdatum: 23.04.2011).

[CGK03] Zesheng Chen, Lixin Gao und Kevin A. Kwiat. Modeling the spread of active worms. In *INFOCOM*, 2003.

[CJM05] Evan Cooke, Farnam Jahanian und Danny McPherson. The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets. In *Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, 2005.

[CL98] Lorrie Faith Cranor und Brian A. LaMacchia. Spam! *Communications of the ACM*, 41:74–83, August 1998.

[Con70] Philip E. Converse. Attitudes and non-attitudes. continuation of a dialogue. In Edward R. Tufte, editor, *The Quantitative Analysis of Social Problems*, Seiten 168–189. Addison-Wesley, Reading u. a., 1970.

[Con09] Lucian Constantin. Fake Windows Security Bulletin Notifications Link to Malware. Internet, Dezember 2009. http://news.softpedia.com/news/Fake-Windows-Security-Bulletin-Notifications-Link-to-Malware-129023.shtml (Zugriffsdatum: 01.12.2010).

[CPM+98] Crispin Cowan, Calton Pu, Dave Maier, Heather Hintony, Jonathan Walpole, Peat Bakke, Steve Beattie, Aaron Grier, Perry Wagle und Qian Zhang. StackGuard: automatic adaptive detection and prevention of buffer-overflow attacks. In *Proceedings of the 7th conference on USENIX Security Symposium - Volume 7*, Berkeley, CA, USA, 1998. USENIX Association.

[CVE00] Common Vulnerabilities und Exposures. IIS 4.0 and 5.0 allows remote attackers to read documents outside of the web root, and possibly execute arbitrary commands, CVE-2000-0884. Internet, 2000. *http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0884* (Zugriffsdatum: 12.10.2010).

[CVE01] Common Vulnerabilities und Exposures. Buffer overflow in ISAPI extension (idq.dll), CVE-2001-0500. Internet, 2001. *http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0500* (Zugriffsdatum: 12.10.2010).

[CVE02] Common Vulnerabilities und Exposures. Buffer overflow in AOL Instant Messenger (AIM), CVE-2002-0005. Internet, 2002. *http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2002-0005* (Zugriffsdatum: 17.01.2011).

[CVE10] Common Vulnerabilities und Exposures. Microsoft Outlook SMB Attachment Vulnerability, CVE-2010-0266. Internet, 2010. *http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-0266* (Zugriffsdatum: 31.01.2011).

[Dah08] Frédiéric Dahl. *Der Storm-Worm*. Diplomarbeit, Lehrstuhl für Praktische Informatik I, Universität Mannheim, Deutschland, März 2008.

[Dav76] James A. Davis. *Are Surveys Any Good, And If So, For What?* Champaign: Pendleton, 1976.

[DGG01] D.J. Daley, J. Gani und J.J.M. Gani. *Epidemic Modelling: An Introduction*. Cambridge Studies in Mathematical Biology. Cambridge University Press, 2001.

[DLFMT04] G. A. Di Lucca, A. R. Fasolino, M. Mastoianni und P. Tramontana. Identifying cross site scripting vulnerabilities in Web applications. *Evolution*, Seiten 71–80, 2004.

[DTH06] Rachna Dhamija, J. D. Tygar und Marti Hearst. Why phishing works. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, CHI '06, Seiten 581–590, New York, NY, USA, 2006. ACM.

[EBB04] Donald L. Evans, Phillip J. Bond und Arden L. Bement. *FIPS Pub 199: Standards for Security Categorization of Federal Information and Information Systems*. National Institute of Standards and Technology, Februar 2004.

[Eck07] Claudia Eckert. *IT-Sicherheit: Konzepte, Verfahren, Protokolle*. R. Oldenbourg, München, Januar 2007.

[EFG+09a] Markus Engelberth, Felix Freiling, Jan Göbel, Christian Gorecki, Thorsten Holz, Ralf Hund, Konrad Rieck, Philipp Trinius und Carsten Willems. Das Internet Malware Analyse System (InMAS): NE1 – Gesamtsystem, Dezember 2009. Internet: *http://mwanalysis.org/inmas/inmas.pdf* (Zugriffsdatum: 20.05.2011).

[EFG+09b] Markus Engelberth, Felix Freiling, Jan Göbel, Christian Gorecki, Thorsten Holz, Ralf Hund, Konrad Rieck, Philipp Trinius und Carsten Willems. Internet Malware Analyse System: InMAS – CW5: Analysis of Malicious Office Files (Part1), Juni 2009.

[EFG+10] Markus Engelberth, Felix C. Freiling, Jan Göbel, Christian Gorecki, Thorsten Holz, Ralf Hund, Philipp Trinius und Carsten Willems. The InMAS Approach. In *Proceedings of the 1st European Workshop on Internet Early Warning and Network Intelligence (EWNI)*, Januar 2010.

[EGGT09] Markus Engelberth, Jan Göbel, Christian Gorecki und Philipp Trinius. Mail-shake. In *Proceedings of the 2009 20th International Workshop on Database and Expert Systems Application*, DEXA '09, Seiten 43–47, Washington, DC, USA, 2009. IEEE Computer Society.

[EL13] Gerhard Eschelbeck und James Lyne. Security threat report 2013. Technischer Bericht, Sophos GmbH, 2013.

[Eva08] David Evans. The economics of the online advertising industry. *Review of Network Economics*, 7(3):359–391, September 2008.

[EWH09] Markus Engelberth, Carsten Willems und Thorsten Holz. MalOffice – Analysis of various application data files. In *Proceedings der 19. Virus Bulletin International Conference (VB2009), Genf, Schweiz*, September 2009.

[FC10] F-Secure Corporation. Informationen über Schadprogramme: *Worm: W32/Mydoom*. Internet, 2010. Internet: http://www.f-secure.com/v-descs/novarg.shtml (Zugriffsdatum: 21.11.2010).

[FF00] Cormac Flanagan und Stephen N. Freund. Type-based race detection for Java. In *Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation*, PLDI '00, Seiten 219–232, New York, NY, USA, 2000. ACM.

[FHW05] Felix Freiling, Thorsten Holz und Georg Wicherski. Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks. In *10th European Symposium On Research In Computer Security (ESORICS)*, 2005.

[FI11a] Facebook Inc. Facebook Homepage. Internet, Januar 2011. *http://www.facebook.com/* (Zugriffsdatum: 28.01.11).

[FI11b] Friendster, Inc. Friendster Homepage. Internet, Januar 2011. *http://www.friendster.com/* (Zugriffsdatum: 28.01.11).

[Fin08] Finjan. Malicious Page of the Month. *http://www.finjan.com/Content.aspx?id=1367* (Zugriffsdatum: Oktober 2008), April 2008.

[Fox09] Dirk Fox. Zero day exploits. *Datenschutz und Datensicherheit - DuD*, 33:250–250, 2009.

[FTD11] Financial Times Deutschland. Unternehmenswert bei 50 Mrd. Dollar: Facebook-Hype kennt keine Grenzen. Internet, Januar 2011. http://www.ftd.de/it-medien/medien-internet/:unternehmenswert-bei-50-mrd-dollar-facebook-hype-kennt-keine-grenzen/50210862.html (Zugriffsdatum: 28.01.2011).

[GMY04] Michel L. Goldstein, Steven A. Morris und Gary G. Yen. Problems with Fitting to the Power-Law Distribution. *European Physical Journal B*, 41(2):4, 2004.

[Gra01] Sarah Granger. Social Engineering Fundamentals, Part I: Hacker Tactics. Security Focus, Infocus, 18. Dezember 2001. *http://www.securityfocus.com/infocus/1527* (Zugriffsdatum: 12.03.2011).

[Gra02] David Gragg. A Multi-Level Defense Against Social Engineering. Technischer Bericht, SANS Institute, Dezember 2002. InfoSec Reading Room.

[GSN+07] Julian B. Grizzard, Vikram Sharma, Chris Nunnery, Brent ByungHoon Kang und David Dagon. Peer-to-Peer Botnets: Overview and Case Study. In *Hot Topics in Understanding Botnets (HotBots)*, 2007.

[Har13] David Harley. Global threat report february 2013: Who gets scammed? Technischer Bericht, ESET, spol. s r. o., 2013.

[HC03] Neal Hindocha und Eric Chien. Malicious Threats and Vulnerabilities in Instant Messaging. White paper, Symantec Security Response, September 2003.

[HEF09] Thorsten Holz, Markus Engelberth und Felix C. Freiling. Learning more about the underground economy: A case-study of keyloggers and dropzones. In *ESORICS*, Seiten 1–18, 2009.

[HGRF08] Thorsten Holz, Christian Gorecki, Konrad Rieck und Felix C. Freiling. Measuring and detecting fast-flux service networks. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2008.

[His11] Hispasec Sistemas. VirusTotal - Free Online Virus, Malware and URL Scanner, 2011. Internet: *http://www.virustotal.com/* (Zugriffsdatum: 27.07.2011).

[HK02] Petter Holme und Beom Jun Kim. Growing scale-free networks with tunable clustering. *Phys. Rev. E*, 65:026107, Jan 2002.

[Hon07] Honeynet Project & Research Alliance. Know Your Enemy: Fast-Flux Service Networks - An Ever Changing Enemy. Technischer Bericht, Juli 2007.

[Hor77] Susan Dadakis Horn. Goodness-of-fit tests for discrete data: a review and an application to a health impairment scale. *Biometrics*, 33(1):237–47, 1977.

[HSD+08] Thorsten Holz, Moritz Steiner, Frédéric Dahl, Ernst W. Biersack und Felix C. Freiling. Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm. In *LEET'08: 1st USENIX Workshop on Large-Scale Exploits and Emergent Threats, April 15, 2008, San Francisco, USA*, 04 2008.

[Int99] ECMA International. *ECMA-262: ECMAScript Language Specification*. ECMA (European Association for Standardizing Information and Communication Systems), 3. Edition, Dezember 1999. *http://www.ecma-international.org/publications/standards/Ecma-262.htm* (Zugriffsdatum: 18.01.2011).

[Int08] Internet Malware Analyse System (InMAS). CW3: Prototyp eines Moduls zur Nutzer-Simulation. Dezember 2008.

[Int10] Internet Corporation For Assigned Names and Numbers (ICANN). Top-Level Domains (gTLDs), August 2010. Internet: *http://www.icann.org/en/tlds/* (Zugriffsdatum: 28.07.2011).

[Itz07] Laura Anna Itzel. Eine Infrastruktur zur Einschätzung des aktuellen Gefährdungslevels durch Malware. Master's thesis, Universität Mannheim, November 2007.

[JJJM07] Tom N. Jagatic, Nathaniel A. Johnson, Markus Jakobsson und Filippo Menczer. Social phishing. *Communications of the ACM*, 50:94–100, October 2007.

[JM06] Markus Jakobsson und Steven Myers. *Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft*. Wiley-Interscience, Dezember 2006.

[JMR09] Owen Jones, Robert Maillardet und Andrew Robinson. *Introduction to Scientific Programming and Simulation Using R*. Chapman & Hall/CRC, Boca Raton, FL, 2009.

[Jus03] Just Landed. Die Social Security Number – Warum Sie in den USA eine SSN benötigen, 2003. Internet: *http://www.justlanded.com/deutsch/Vereinigte-Staaten/Artikel/Finanzen/Die-Social-Security-Number* (Zugriffsdatum: 14.05.11).

[KAG06] Andrew Kalafut, Abhinav Acharya und Minaxi Gupta. A study of malware in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, IMC '06, Seiten 327–332, New York, NY, USA, 2006. ACM.

[Kee01] James E. Keeling. Social Engineering – For the Good Guys. Technischer Bericht, SANS Institute, Juli 2001. InfoSec Reading Room.

[KK06a] Josephine D. Korchmaros und David A. Kenny. An evolutionary and close-relationship model of helping. *Journal of Social and Personal Relationships*, 23(1):21–43, Februar 2006.

[KK06b] Gregory P. Kruck und S. E. Kruck. Spoofing – a look at an evolving threat. *The Journal of Computer Information Systems*, Oktober 2006.

[KKL+08] Chris Kanich, Christian Kreibich, Kirill Levchenko, Brandon Enright, Geoffrey M. Voelker, Vern Paxson und Stefan Savage. Spamalytics: an empirical analysis of spam marketing conversion. In *Proceedings of the 15th ACM conference on Computer and communications security*, CCS '08, Seiten 3–14, New York, NY, USA, 2008. ACM.

[KLE+08] Chris Kanich, Kirill Levchenko, Brandon Enright, Geoffrey M. Voelker und Stefan Savage. The Heisenbot Uncertainty Problem: Challenges in Separating Bots from Chaff. In *First Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2008.

[KSTW07] Chris Karlof, Umesh Shankar, J. D. Tygar und David Wagner. Dynamic pharming attacks and locked same-origin policies for web browsers. In *Proceedings of the 14th ACM conference on Computer and communications security*, CCS '07, Seiten 58–71, New York, NY, USA, 2007. ACM.

[KW91] Jeffrey O. Kephart und Steve R. White. Directed-graph epidemiological models of computer viruses. In *IEEE Symposium on Security and Privacy*, Seiten 343–361, 1991.

[KWK+07] Imran Khan, Bridget Weishaar, Vasily Karasyov, Lev Polinsky und Joseph Boushelle. The Rise of Ad Networks: An In-Depth Look at Ad Networks. Technischer Bericht, JPMorgan Chase & Company, Oktober 2007. i*http://www.mediamath.com/docs/JPMorgan.pdf* (Zugriffsdatum: 15:01.2011).

[LAHR10] Michael Ligh, Steven Adair, Blake Hartstein und Matthew Richard. *Malware Analyst's Cookbook and DVD: Tools and Techniques for Fighting Malicious Code*. John Wiley & Sons, 2010.

[Lan07] Mary Landesman. Security viewpoint artikel - malware revolution: A change in target, März 2007. Internet: *http://technet.microsoft.com/en-us/library/cc512596.aspx* (Zugriffsdatum: 08.12.2012).

[LH08] Jure Leskovec und Eric Horvitz. Planetary-Scale Views on a large Instant-Messaging Network. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, Seiten 915–924, New York, NY, USA, 2008. ACM.

[Lin11] Christopher Linnemann. Statistik: E-Mail Nutzung im Jahr 2010 (E-Mail Marketing Blog | DoYouMail.de), August 2011. Internet: *http://www.doyoumail.de/statistiken/statistik-e-mail-nutzung-im-jahr-2010/* (Zugriffsdatum: 03.01.2012).

[LM06] Amy N. Langville und Carl D. Meyer. *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press, Princeton, NJ, USA, 2006.

[Mas51] Frank J. Massey, Jr. The Kolmogorov-Smirnov Test for Goodness of Fit. *Journal of the American Statistical Association*, 46(253):68–78, 1951.

[MC11a] Microsoft Corporation. Encyclopedia entry: TrojanDownloader:Win32/Bredolab.AB - Learn more about malware - Microsoft Malware Protection Center, April 2011. Internet: http://www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?Name=TrojanDownloader:Win32/Bredolab.AB (Zugriffsdatum: 27.07.2011).

[MC11b] Microsoft Corporation. Encyclopedia entry: TrojanDownloader:Win32/Chepvil - Learn more about malware - Microsoft Malware Protection Center, März 2011. Internet: http://www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?Name=TrojanDownloader:Win32/Chepvil (Zugriffsdatum: 27.07.2011).

[MC11c] Microsoft Corporation. Encyclopedia entry: TrojanDownloader:Win32/Oficla - Learn more about malware - Microsoft Malware Protection Center, Mai 2011. Internet: *http://www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?Name=Win32/Oficla* http://www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?Name=Win32/Oficla (Zugriffsdatum: 27.07.2011).

[MI11] Myspace, Inc. Myspace Homepage. Internet, Januar 2011. *http://www.myspace.com/* (Zugriffsdatum: 28.01.11).

[Min09] Martin Mink. *Vergleich von Lehransätzen für die Ausbildung in IT-Sicherheit*. Dissertation, Universität Mannheim, 2009. Internet:

http://madoc.bib.uni-mannheim.de/madoc/volltexte/2010/2943/pdf/dissertation_mink.pdf.

[Moz11] Mozilla Foundation. Mozilla Thunderbird, Dezember 2011. Internet: *http://www.mozilla.org/en/thunderbird/* (Zugriffsdatum: 26.12.2011).

[MR04] Jelena Mirkovic und Peter Reiher. A taxonomy of ddos attack and ddos defense mechanisms. *SIGCOMM Comput. Commun. Rev.*, 34:39–53, April 2004.

[MRRV01] Andrew Mackie, Jensenne Roculan, Ryan Russell und Mario Van Velzen. Incident Analysis Report: Nimda Worm Analysis – Version 1. Technischer Bericht, SecurityFocus, September 2001.

[MS06] Kevin D. Mitnick und William Simon. *Die Kunst der Täuschung: Risikofaktor Mensch*. Mitp-Verlag, 1. Edition, 2006.

[MSC02] David Moore, Colleen Shannon und K. Claffy. Code-Red: a case study on the spread and victims of an internet worm. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, IMW '02, Seiten 273–284, New York, NY, USA, 2002. ACM.

[MvO05] Mohammad Mannan und Paul C. van Oorschot. On instant messaging worms, analysis and countermeasures. In *Proceedings of the 2005 ACM workshop on Rapid malcode*, WORM '05, Seiten 2–11, New York, NY, USA, 2005. ACM.

[Nel02] Rick Nelson. Methods of Hacking: Social Engineering, Oktober 2002. Internet: http://th3-0utl4ws.com/Database/E-books/Methods of Hacking - Social Engineering.pdf (Zugriffsdatum: 13.05.11).

[New05] M. E. J. Newman. Power laws, pareto distributions and zipf's law. *Contemporary Physics*, 46(5):323–351, 2005.

[New06] M E Newman. Modularity and community structure in networks. *Proc Natl Acad Sci U S A*, 103(23):8577–8582, June 2006.

[NI11] hi5 Networks, Inc. hi5 Homepage. Internet, Januar 2011. *http://hi5.com/* (Zugriffsdatum: 28.01.11).

[NL13] Netcraft Ltd. June 2013 Web Server Survey. Internet, Juni 2013. http://news.netcraft.com/archives/2013/06/06/june-2013-web-server-survey-3.html (Zugriffsdatum: 21.06.2013).

[NM92] Robert H. B. Netzer und Barton P. Miller. What are race conditions?: Some issues and formalizations. *ACM Lett. Program. Lang. Syst.*, 1:74–88, März 1992.

[One96] Aleph One. Smashing The Stack For Fun And Profit. *Phrack*, 7(49), November 1996.

[Orm03] Hilarie Orman. *The Morris Worm: A Fifteen-Year Perspective*. *IEEE Security and Privacy*, 1:35–43, 2003.

[Pel06] Thomas R. Peltier. Social Engineering: Concepts and Solutions. *Information Systems Security*, 15(5):13–21, 2006.

[PLR07] Tao Peng, Christopher Leckie und Kotagiri Ramamohanarao. Survey of network-based defense mechanisms countering the dos and ddos problems. *ACM Comput. Surv.*, 39, April 2007.

[PMM+07] Niels Provos, Dean McNamee, Panayiotis Mavrommatis, Ke Wang und Nagendra Modadugu. The Ghost in the Browser: Analysis of Web-based Malware. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, Berkeley, CA, USA, 2007. USENIX Association.

[Poh09] Hartmut Pohl. Zero-day und less-than-zero-day vulnerabilities und exploits. In Christoph Zacharias, Klaus W. Horst, Kurt-Ulrich Witt, Volker Sommer, Marc Ant, Ulrich Essmann und Laurenz Mülheims, editors, *Forschungsspitzen und Spitzenforschung*, Seiten 113–123. Physica-Verlag HD, 2009.

[PP01] M. Parker und J. Parish. *The age of anxiety: conspiracy theory and the human sciences*. Sociological review monograph. Blackwell/Sociological Review, 2001. Seiten 17–30.

[Pro04] The Honeynet Project. *Know Your Enemy: Learning About Security Threats*. Addison-Wesley Longman, 2. Edition, 2004.

[QI11] Quark Inc. QuarkXPress 9 – Desktop-Publishing-Software – Seitenlayout-Software; Homepage, 2011. Internet: *http://www.quark.com/Products/QuarkXPress* (Zugriffsdateum: 17.05.11).

[Ras07] Tinku Rasheed. Wireless Mesh Network Simulation Framework for OMNeT++. Technischer Bericht CN-TR-200700016, CREATE-NET, 2007.

[Res08] P. Resnick. *Internet Message Format*. RFC 5322 (Standards Track), Oktober 2008. *http://tools.ietf.org/rfc/rfc5322.txt*.

[RL13] Sara Radicati und Justin Levenstein. Email Statistics Report, 2013–2017. Technischer Bericht, The Radicati Group, Inc., April 2013.

[RLZ08] Paruj Ratanaworabhan, Benjamin Livshits und Benjamin Zorn. Nozzle: A Defense Against Heap-spraying Code Injection Attacks. Technischer Bericht MSR-TR-2008-176, Microsoft Research, 2008.

[Rob07] Shane W. Robinson. Corporate Espionage 201. Technischer Bericht, SANS Institute, Dezember 2007. InfoSec Reading Room.

[RZMT06] Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose und Andreas Terzis. A Multifaceted Approach to Understanding the Botnet Phenomenon. In *6th Internet Measurement Conference (IMC)*, 2006.

[SCO10] SCO Group, Inc. *Internetseite der SCO Group, Inc.* Internet, 2010. *http://www.sco.com/* (Zugriffsdatum: 28.11.2010).

[Sec07] SecureWorks. PRG Trojan. *http://www.secureworks.com/research/threats/prgtrojan/* (Zugriffsdatum: Oktober 2008), June 2007.

[Sec08] SecureWorks. Coreflood Report. *http://www.secureworks.com/research/threats/coreflood-report/* (Zugriffsdatum: Oktober 2008), August 2008.

[SGE+09] Ben Stock, Jan Göbel, Markus Engelberth, Felix C. Freiling und Thorsten Holz. Walowdac - Analysis of a Peer-to-Peer Botnet. In *2009 European Conference on Computer Network Defense (EC2ND)*, Seiten 13–20, November 2009.

[She08] Sergei Shevchenko. ThreatExpert Blog: One Tricky Banking Trojan, November 2008. Internet: *http://blog.threatexpert.com/2008/11/one-tricky-banking-trojan.html* (Zugriffsdatum: 15.05.11).

[SJB06] Seungwon Shin, Jaeyeon Jung und Hari Balakrishnan. Malware Prevalence in the KaZaA File-Sharing Network. In *Internet Measurement Conference (IMC)*, Rio de Janeiro, Brazil, Oktober 2006.

[Spi10] Spiegel Online. Bundespräsident: Ramsauer lehnt Köhlers Spritpreis-Vorstoß ab. *http://www.spiegel.de/politik/deutschland/0,1518,684889,00.html* (Zugriffsdatum: Oktober 2010), März 2010.

[SPW02] Stuart Staniford, Vern Paxson und Nicholas Weaver. How to 0wn the Internet in Your Spare Time. In *Proceedings of the 11th USENIX Security Symposium*, San Francisco, CA, August 2002.

[SS87] Howard Schuman und Jacqueline Scott. Problems in the Use of Survey Questions to Measure Public Opinion. *Science*, 236(4804):957–959, 1987.

[Sta07] Mika Stahlberg. The Trojan Money Spinner. In *Virus Bulletin Conference*, 2007.

[Str94] Fritz Strack. *Zur Psychologie der standardisierten Befragung - Kognitive und kommunikative Prozesse*. Springer, 1994.

[Sva09] Vanja Svajcer. Beware of fake Microsoft updates coming through email. Internet, Dezember 2009. *http://nakedsecurity.sophos.com/2009/12/04/beware-fake-microsoft-updates-coming-email/* (Zugriffsdatum: 03.12.2010).

[Sym06] Symantec. Evolving Shell Code. In *Proceedings of the 16th Virus Bulletin International Conference*, 2006.

[Sym08] Symantec. Global Internet Security Threat Report, April 2008. Trends for July – December 07.

[Sym10] Symantec. Informationen über Schadprogramme: *W32.Mydoom.A@mm*. Internet, 2010. http://www.symantec.com/security_response/writeup.jsp?docid=2004-012612-5422-99 (Zugriffsdatum: 21.11.2010).

[SZ04] Ed Skoudis und Lenny Zeltser. *Malware: Fighting Malicious Code*. Prentice Hall series in computer networking and distributed systems. Prentice Hall PTR, November 2004.

[Szo05] Peter Szor. *The art of computer virus research and defense*. Symantec Press Series. Addison-Wesley, 2005.

[TI11] Twitter, Inc. Twitter Homepage. Internet, Januar 2011. *http://twitter.com/* (Zugriffsdatum: 28.01.11).

[TN10] Kurt Thomas und David M. Nicol. The Koobface botnet and the rise of social malware. In *Proceedings of the 5th International Conference on*

Malicious and Unwanted Software (MALWARE), Seiten 63–70. IEEE, Oktober 2010.

[Var10] András Varga. *OMNeT++ User Manual*. OpenSim Ltd., 4.1 Edition, 2010. *http://www.omnetpp.org/doc/omnetpp41/Manual.pdf* (Aug. 2010).

[VH08] András Varga und Rudolf Hornig. An overview of the OMNeT++ simulation environment. In *Simutools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, Seiten 1–10, Brüssel, Belgien, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[VTM09] Prashanth Ramagopal Vinoo Thomas und Rahul Mohandas. The Rise of AutoRun-Based Malware. White paper, McAfee, Inc., Juni 2009.

[Wan09] Jie Wang. *Computer Network Security: Theory and Practice*. Springer, 1. Edition, März 2009.

[WBW+06] Yi-Min Wang, Doug Beck, Jeffrey Wang, Chad Verbowski und Brad Daniels. Strider typo-patrol: discovery and analysis of systematic typo-squatting. In *Proceedings of the 2nd conference on Steps to Reducing Unwanted Traffic on the Internet - Volume 2*, Seiten 31–36, Berkeley, CA, USA, 2006. USENIX Association.

[Whe03] David A. Wheeler. *Secure Programming for Linux and Unix HOWTO*. v3.010 Edition, März 2003.

[WHF07] Carsten Willems, Thorsten Holz und Felix Freiling. Toward Automated Dynamic Malware Analysis Using CWSandbox. *IEEE Security & Privacy Magazine*, 5(2):32–39, März 2007.

[Wor07] Michael Workman. Gaining Access with Social Engineering: An Empirical Study of the Threat. *Information Security Journal*, 16:315–331, November 2007.

[WPB04] Matthew M. Williamson, Alan Parry und Andrew Byde. Virus Throttling for Instant Messaging. In *Proceedings der 14. Virus Bulletin International Conference (VB2004), Chicago, United States of America*, September 2004.

[WSL09] Websense Security Labs. Waledac Independence Day Theme - New Campaign In The Wild. Internet, März 2009. *http://securitylabs.websense.com/content/Alerts/3431.aspx* (Zugriffsdatum: 02.12.2010).

[XFZ09] Yang Xiang, Xiang Fan und Wen Tao Zhu. Propagation of active worms: A survey. *Comput. Syst. Sci. Eng.*, 24(3), 2009.

[ZGT02] Cliff Changchun Zou, Weibo Gong und Don Towsley. Code red worm propagation modeling and analysis. In *Proceedings of the 9th ACM conference on Computer and communications security*, CCS '02, Seiten 138–147, New York, NY, USA, 2002. ACM.

[ZTG04] Cliff C. Zou, Don Towsley und Weibo Gong. Email worm modeling and defense, 2004.

Anhang

ANHANG A

Werbetext zur Vorstellung der Spam-Experimente

Durch die immer häufiger bekannt werdenden Vorfälle des Datendiebstahls/-verlusts, steigt bei Unternehmen das Bewusstsein für eine sichere IT-Infrastruktur. Neben den teilweise hohen finanziellen Schäden durch das Abhandenkommen von sensiblen Daten, tragen sicherlich auch vermehrt die dadurch entstehenden Imageschäden zu dieser Bewusstseinssteigerung bei.

Als eine Folge dessen steigen die Investitionen in die Absicherung der IT-Infrastruktur. So werden beispielsweise teure Firewall-Konzepte, Antiviren-Lösungen oder Intrusion-Detection-Systeme installiert. Doch was passiert, wenn Angreifer diese Absicherungen umgehen und versuchen, das möglicherweise schlechte Sicherheitsbewusstsein Ihrer Mitarbeiter auszunutzen, um darüber in Ihr Firmennetzwerk einzudringen? Die besten und teuersten Sicherheitskonzepte bieten in diesen Fällen gewöhnlich keinen Schutz. Aktuelle Trends zeigen, dass sich das Angreiferverhalten immer mehr in diese Richtung verlagert. Erhalten Mitarbeiter Ihres Unternehmens etwa E-Mails mit bösartigen Dateianhängen oder Verweisen auf bösartig präparierte Webseiten, stellt dies bei einem unbedachten Mitarbeiter eine große Gefahr dar. Werden diese Anhänge heruntergeladen und ausgeführt oder ein Mitarbeiter folgt einem der Verweise, ist die Gefahr groß, dass danach der Rechner dieses Mitarbeiters mit Schadsoftware infiziert ist. Dieses kleine Beispiel macht bereits deutlich, wie wichtig das Gefahrenbewusstsein der Mitarbeiter für die gesamte Sicherheitsstrategie Ihres Unternehmens sein kann und dass keineswegs eine alleinige Absicherung der IT-Infrastruktur einen ausreichenden Schutz vor IT-Gefahren bietet.

Wenn Sie wissen möchten, wie hoch das Gefahrenbewusstsein Ihrer Mitarbeiter ist, bieten wir Ihnen folgende Dienstleistung an: Wir bieten Ihnen eine kostenlose und OPDV-zertifizierte Software, die in Ihrem Institut die Vorgehensweise eines Angreifers simuliert. Mit den dabei verschickten E-Mails können wir messen, wie hoch das Sicherheitsbewusstsein Ihrer Mitarbeiter ist. Selbstverständlich geht von diesen E-Mails keinerlei Gefahr aus, sie dienen lediglich zum Messen der Schwachstelle Mensch. Die von uns angebotene Software liegt in zwei Versionen vor:

1. Eine einfache Version, die Ihnen einen schnellen Überblick darüber liefert wie anfällig Ihr Institut für Angriffe ist, die mit konventioneller Hardware nicht oder nur kaum verhindert werden können. Bei dieser Version werden keinerlei Informationen über die Empfänger (Ihre Mitarbeiter) gespeichert. Als Ergebnis werden sie beispielsweise wissen, wie viele Ihrer Mitarbeiter einen Anhang heruntergeladen haben oder diesen sogar ausführten.
2. Die zweite Version arbeitet fast identisch zur ersten Version, nur dass sie in einer pseudonymisierten Art und Weise Ergebnisse speichert. Dadurch ermöglichen wir Ihnen, den Erfolg einer möglichen Fortbildung Ihrer Mitarbeiter zu quantifizieren, indem diese Version unserer Software zu einem späteren Zeitpunkt erneut ausgeführt wird. Wir versichern Ihnen, dass bei dieser Version ebenfalls keinerlei Daten über Ihre Mitarbeiter gespeichert werden, die Rückschlüsse auf einzelne Identitäten zulassen.

ANHANG B

Einverständniserklärung

Diese Seite ist absichtlich leer, damit die Einverständniserklärung auf der nächsten Seite größer skaliert dargestellt werden kann.

FAKULTÄT FÜR MATHEMATIK UND INFORMATIK
Lehrstuhl für Praktische Informatik 1

UNIVERSITÄT MANNHEIM

Sekretariat des Lehrstuhls:
Telefon 06 21 / 1 81-2540
Telefax 06 21 / 1 81-3577
braak@informatik.uni-mannheim.de
http://pi1.informatik.uni-mannheim.de/

Markus Engelberth
A 5, 6, 68131 Mannheim
Telefon 06 21 / 1 81-2669
engelberth@informatik.uni-mannheim.de

Einverständniserklärung

Im Rahmen seiner Promotionsarbeit hat Herr Markus Engelberth vom Lehrstuhl für Praktische Informatik 1 ein Programm erstellt, das auf einem Mailserver die Zeitstempel des Eintreffens und des Lesens der E-Mails eines registrierten Benutzers ermitteln und ausgeben kann. Die Ausgaben sind nicht personenbezogen und lassen weder Rückschlüsse auf den Empfänger, den Absender, den Inhalt oder den Betreff der E-Mails zu.

Hiermit erkläre ich mich damit einverstanden, dass das von Herrn Markus Engelberth erstellte Programm auf meine unten angegebene Mailbox der Universität Mannheim angewandt werden darf.

Ebenfalls erkläre ich mich damit einverstanden, dass die dabei produzierten Ausgaben des Programms Herrn Markus Engelberth zum Zwecke seiner Dissertation zur Verfügung gestellt werden.

Vorname: | | | | | | | | | | | | | | | | | | |

Nachname: | | | | | | | | | | | | | | | | | | |

Benutzerkennung (achtstellig): | | | | | | | |

Ort, Datum

Unterschrift

Abbildung B.1.: Einverständniserklärung
Durch das Ausfüllen dieser Erklärung haben sich Studenten und Mitarbeiter der Universität Mannheim damit einverstanden erklärt, dass Informationen über ihr Leseverhalten von E-Mails aus ihrer Mailbox extrahiert und im Rahmen dieser Arbeit verarbeitet werden dürfen.

ANHANG C

Skript zum Auslesen von E-Mail-Zeitinformationen

```
#!/usr/bin/python
import sys, os, time, struct, stat, email, re, subprocess, hashlib

seenDir = "/var/lib/imap/user/"
mbpath_error_praefix = "Invalid mailbox name"

seen = None
rectypes = {
  1: 'INORDER',
  2: 'ADD',
  4: 'DELETE',
  255: 'COMMIT',
  257: 'DUMMY',
}


# mailbox.seen -> dict[folderID] = [ ( timestamp, seen-Pattern ), ... ]
# Based on http://www.majid.info
class Skiplist(dict):
  def __init__(self, name):
    global rectypes
    size = os.stat(name)[stat.ST_SIZE]
    f = open(name)
    assert f.read(20) == "\241\002\213\015skiplist file\0\0\0"
    major, minor, maxlevel, curlevel, listsize, log_start, last_recovery = struct.unpack('!7I', f.
        read(28))
    assert major == 1
    i = 0
    while True:
      i += 1
      rectype = f.read(4)
      if not rectype: break # EOF
      rectype = struct.unpack('!I', rectype)[0]
      rectype = rectypes[rectype]
      if rectype == 'COMMIT':
        continue
      if rectype == 'DELETE':
        delptr = struct.unpack('!I', f.read(4))[0]
        continue
      keysize = struct.unpack('!I', f.read(4))[0]
      key = f.read(keysize)
      f.read(((keysize + 3) & 0xFFFFFFFC) - keysize)
      datasize = struct.unpack('!I', f.read(4))[0]
      data = f.read(datasize).split()
      f.read(((datasize + 3) & 0xFFFFFFFC) - datasize)
      if key not in self.keys():
        self[key] = []
      try:
        if (data[3], data[4]) not in self[key]:
          self[key].append((data[3], data[4]))
      except:
        pass
      while True:
        skipptr = struct.unpack('!I', f.read(4))[0]
        if skipptr == 0xFFFFFFFFL: break
    f.close()


```

```
58  # seen-Pattern -> set( mailID_1, mailID_2, .... )
59  def decode_seen(encoded_seen):
60    seenset = set()
61    try:
62      if encoded_seen:
63        for interval in encoded_seen.split(','):
64          if ':' in interval:
65            begin, end = interval.split(':')
66            seenset.update(range(int(begin), int(end) + 1))
67          else:
68            seenset.add(int(interval))
69      return seenset
70    except:
71      return set()
72
73
74  # folderID -> dict[mailID] = ( timestamp_NICHT_gelesen, timestamp_gelesen )
75  # Based on http://www.majid.info
76  def getReadTimes(folderID):
77    result = {}
78    processed = decode_seen(seen[folderID][0][1])
79    lastTimestamp = seen[folderID][0][0]
80    for entry in seen[folderID][1:]:
81      seenMails = decode_seen(entry[1])
82      currentTimestamp = entry[0]
83      diffMails = seenMails
84      diffMails.difference_update(processed)
85      for mailID in diffMails:
86        result[mailID] = (lastTimestamp, currentTimestamp)
87      lastTimestamp = currentTimestamp
88      processed.update(seenMails)
89    return result
90
91
92  # Based on http://www.majid.info
93  def email_received(filename):
94    m = email.message_from_file(open(filename))
95    received = m.get_all('received')
96    if received:
97      received = received[0].split(';')[-1].strip()
98      ts_received = time.mktime(email.Utils.parsedate(received))
99    else:
100     ts_received = None
101   if 'date' in m:
102     ts_date = time.mktime(email.Utils.parsedate(m['date']))
103   else:
104     ts_date = None
105   try:
106     ts_ctime = os.stat(filename)[stat.ST_CTIME]
107   except:
108     ts_ctime = None
109   return (ts_received, ts_date, ts_ctime)
110
111
112 def header2ID(filename):
113   header_magic = """\241\002\213\015Cyrus mailbox header
114 "The best thing about this system was that it had lots of goals."
115 \t--Jim Morris on Andrew
116 """
117   f = open(filename)
118   assert f.read(len(header_magic)) == header_magic
119   line = f.readline()
120   f.close()
121   return line.split('\t')[1].strip()
122
123
124 def getTimes(mbpath, char, mailbox):
125   global seen
126   print "SHA1(mailbox,folderID,mailID) (TS_RECEIVED, TS_DATE, TS_CTIME)\tNOT_READ\tREAD"
127   reSubpath = re.compile("/(\w)/user/(.*?)/(.*?)cyrus.header$")
128   ID2folder = {}
129   for (root, dirs, files) in os.walk(mbpath):
130     for filename in files:
131       if filename == "cyrus.header":
132         matSubpath = reSubpath.search(os.path.join(root, filename))
133         if matSubpath:
134           subpath = matSubpath.group(3)
135           if subpath.endswith("/"):
136             subpath = subpath[:-1]
137           ID2folder[header2ID(os.path.join(root, filename))] = subpath
138   seen = Skiplist(os.path.join(seenDir, char, '%s.seen' % mailbox))
139   for folderID in seen.keys():
140     if len(seen[folderID]) < 2:
141       del seen[folderID]
142   for folderID in seen.keys():
143     if folderID not in ID2folder.keys():
144       continue
145     if len(seen[folderID]) > 1:
146       readTimes = getReadTimes(folderID)
147       for mailID in readTimes.keys():
```

```
148         sha1MailID = hashlib.sha1("%s%s%s" % (mailbox, folderID, mailID)).hexdigest()
149         if not os.path.exists(os.path.join(mbpath, ID2folder[folderID], str(mailID) + '.')):
150           pass
151         else:
152           ctime = email_received(os.path.join(mbpath, ID2folder[folderID], str(mailID) + '.'))
153           print "%s\t%s\t%s\t%s" % (sha1MailID, ctime, readTimes[mailID][0], readTimes[mailID][1])
154
155
156 def getPath(mbname):
157   procMbpath = subprocess.Popen("/usr/lib/cyrus/bin/mbpath user.%s" % mbname, stdout=subprocess.PIPE
          , stdin=subprocess.PIPE, stderr=subprocess.PIPE, shell=True)
158   procMbpath.wait()
159   if len(procMbpath.stderr.read()) == 0:
160     output = procMbpath.stdout.read().strip()
161     if output.lower().startswith(mbpath_error_praefix.lower()):
162       return None
163     else:
164       return output
165   else:
166     return None
167
168
169 # Hauptfunktion
170 if __name__ == '__main__':
171   try:
172     mailbox = sys.argv[1]
173   except:
174     print "USAGE: python %s <mailbox name>" % sys.argv[0]
175     sys.exit(1)
176   mbpath = getPath(mailbox)
177   if mbpath == None:
178     print 'ERROR: No path found for mailbox name "%s"' % mailbox
179     sys.exit(2)
180   reChar = re.compile("/imap/(.)/user/")
181   matChar = reChar.search(mbpath)
182   try:
183     if matChar:
184       char = matChar.group(1)
185     else:
186       raise
187   except:
188     print "ERROR: Cannot find hash character for mailbox."
189     sys.exit(3)
190   try:
191     print 'time      :', time.time()
192     print 'altzone   :', time.altzone
193     print 'localtime : %s (%s)' % (repr(time.localtime()), time.mktime(time.localtime()))
194     print 'ctime     :', time.ctime()
195     print 'timezone  : %s\n' % repr(time.tzname)
196   except:
197     print 'ERROR: Exception while examination of time informations.\n'
198   if os.path.isdir(mbpath):
199     getTimes(mbpath, char, mailbox)
```

Code-Block C.1: Python-Skript zum Auslesen der Eintreff- und Lesezeiten von E-Mails