**Reihe Informatik. TR-2007-010**

# Characterizing the IRC-based Botnet Phenomenon

Jianwei Zhuge[1], Thorsten Holz[2], Xinhui Han[1],
Jinpeng Guo[1], and Wei Zou[1]

[1] Peking University
Institute of Computer Science
and Technology
Beijing, China

[2]University of Mannheim
Laboratory for
Dependable Distributed Systems
Mannheim, Germany

December 3, 2007

**Abstract**

Botnets, networks of compromised machines that can be remotely controlled by an attacker, are one of the most common attack platforms nowadays. They can, for example, be used to launch distributed denial-of-service (DDoS) attacks, steal sensitive information, or send spam emails. A long-term measurement study of botnet activities is useful as a basis for further research on global botnet mitigation and disruption techniques. We have built a distributed and fully-automated botnet measurement system which allows us to collect data on the botnet activity we observe in China. Based on the analysis of tracking records of 3,290 IRC-based botnets during a period of almost twelve months, this paper presents several novel results of botnet activities which can only be measured via long-term measurements. These include. amongst others, botnet lifetime, botnet discovery trends and distributions, command and control channel distributions, botnet size and end-host distributions. Furthermore, our measurements confirm and extend several previous results from this area.

Our results show that the botnet problem is of global scale, with a scattered distribution of the control infrastructure and also a scattered distribution of the victims. Furthermore, the control infrastructure itself is rather flexible, with an average lifetime of a Command & Control server of about 54 days. These results can also leverage research in the area of botnet detection, mitigation, and disruption: only by understanding the problem in detail, we can develop efficient counter measures.

## 1 Introduction

The term *botnet* can be defined as a network of end-hosts infected by *bots*, a certain type of malware. These networks are commonly organized through an one-to-many communication channel and all compromised machines are under the control of human operators, called *botherders*. Botnets first appeared more than eight years ago with PrettyPark in 1999 as one of the first malware samples with an IRC-based backdoor. Since then, botnets have developed into the first-choice attack platform for network-based attacks. With the help of a botnet, an attacker has control over all compromised machines and this remote control structure helps him to carry out his attacks more efficiently. Botnets are commonly used for distributed denial-of-service (DDoS) attacks, identity theft, installation of additional malware on the compromised machines, or additional mischief [18]. Usually, botnets use Internet Relay Chat (IRC [11]) as communication protocol, but also other protocols like HTTP or Peer-to-Peer-based protocols can be used by the botherder to command the clients. Since IRC is still the common communication protocol used by botnets today, we focus in this study on IRC-based botnets.

In this paper, we present the results of a long-term, large-scale measurement study on IRC-based botnets in the wild. With the help of *honeypots*, a certain form of network decoys [14], we collect a

large number of autonomous spreading malware. Based on an analysis of these malware samples, we can detect botnets in an automated way. Furthermore, we also track these networks by sending a *snoop* into the botnets that monitors all actions happening in the botnet from the inside. In total, our study is based on a collection of about 90,000 samples of autonomous spreading malware, of which 5,645 are bots that successfully connected to the botnet control channel during our analysis. By analyzing these samples, we detected a total of 3,290 IRC-based botnets which we tracked during the measurement period between June 2006 and June 2007. The results of our study characterize the current extent of the botnet problem. We present typical features of botnet Command & Control (C&C) servers like the geographical location, the TCP ports used for control channels, and the server software used. Furthermore, we measure botnet properties like average lifetime, average size of a botnet, overlap between different botnets, and diurnal patterns. We also describe botnet activities we have observed in the wild and present detailed results for spreading, DDoS, and download activities caused by botnets.

These results can be used as a basis to develop techniques for detecting and mitigating IRC-based botnets: only by understanding the problem in detail, we can develop efficient counter-mechanisms. Furthermore, we studied the botnet phenomenon for about twelve months and can thus also infer long-term trends in the development of botnets. These results can also be used to get a deeper understanding of the phenomenon, which in turn can be used to develop tools and techniques for disrupting these networks. During our study, we found evidence of about half a million infected IP addresses and therefore effective counter measures are needed to stop this threat.

This paper is outlined as follows: Section 2 provides an overview of related work which studies the global botnet phenomenon. In Section 3, we describe our measurement setup and give a brief background on the tools and methods used during our study. We present the analysis results in Section 4, where we focus on four different aspects of botnets. Finally, we conclude the paper in Section 5.

# 2 Related Work

In the last few years, the botnet phenomenon got the general attention of the security research community. One of the first systematic studies was published in March 2005 by the Honeynet Project, that studied about 100 botnets during a period of four months [18]. A more methodical approach was introduced by Freiling et al., who used the same amount of botnet data for their study [8]. Cooke et al. outlined the origins and structure of botnets and present some results based on a distributed network sensor system and honeypots [6]. They do not give detailed results that characterize the extent of the botnet problem. Rajab et al. used DNS data and monitoring of C&C control activity to get a better understanding of botnets [15]. Their study was based on data collected by monitoring 192 botnets during a period of more than three months. Compared to all these studies, our study is based on data of one magnitude more botnets (3,290 botnets) collected in a period of about one year. We can observe trends and long-term effects of the botnet phenomenon like the average lifetime of a botnet not possible with previous studies.

A transport layer-based botnet detection approach was introduced by Karasaridis et al. [12]. They use passive analysis based on flow data to characterize botnets and were able to detect several hundred controllers over a period of seven months. However, such a flow-based approach can not provide insight into the botnet and the control structure itself. In our study, we can also observe the commands issued by botherders, the malware binary executables used, and similar inside effects of a botnet. Canavan [4] and Barford and Yegneswaran [2] presented an alternative perspective on IRC-based botnets based on in-depth analysis of bot source code. We also analyzed the source code of several bot families such as SdBot and Agobot, which can be freely downloaded from the Internet, to get a better understanding of some of the effects we monitored during our observations.

In our study, we focus only on IRC-based bots. Botnets that use Peer-to-Peer based protocols have also been observed in the wild [9]. The general approach outlined in this paper can be extended to also study this kind of botnets, which we plan as part of our future work.

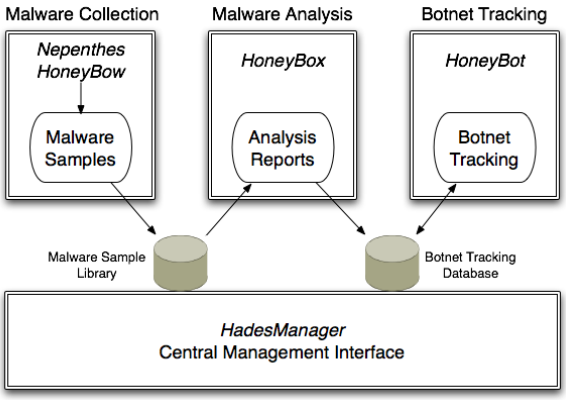



Figure 1: General overview of measurement approach for botnet study

# 3 Measurement Setup

Our methodology follows a standard botnet measurement method based on honeypot technology and botnet snooping, similar to previous studies in this area [8, 15, 18]. We extend this methodology with an additional approach for malware collection (Section 3.1) and malware analysis (Section 3.2). Furthermore, our approach is fully automated and has proven to run stable for several months. On the one hand, this allows us to perform a long-term study of the botnet phenomenon, and on the other hand we can perform such a measurement study on a large-scale basis.

As depicted in Figure 1, our botnet measurement methodology consists of three phases:

1. *Malware collection*: with the help of nepenthes [1] and HoneyBow (Section 3.1), we collect samples of bots in a fully automated way.

2. *Malware analysis*: a behavior-based analysis engine called HoneyBox is used to extract information about the botnet command and control infrastructure from the captured bot samples (Section 3.2).

3. *Botnet tracking*: the extracted information is used to send a snoop into the control channel of the botnet who tracks what is happening within the botnet (Section 3.3).

All malware collected during these phases is stored in a central malware sample library. Furthermore, all extracted information and all botnet tracking records are stored in a central database. We developed a web interface to access the library and the database. In addition, this interface provides a central management console for all phases, and it enables the monitoring of the status of all sensors, browsing of the collected information, and similar tasks.

## 3.1 Malware Collection

In the first phase, we collect samples of bot binaries with a honeynet approach. A common method for bots to propagate further is exploitation of remote vulnerabilities. With the help of a honeypot, we pretend to be a vulnerable system, but in reality, we are just a decoy, waiting to be exploited. We use two different types of honeypots. As a *low-interaction honeypot*, we use nepenthes [1]. A low-interaction honeypot provides only limited interaction for an attacker. By design, it is not meant to represent a fully featured system and usually cannot be completely exploited. Nevertheless, such a honeypot allows us to capture known exploitation attempts and quantitative information like the top attacked ports on a given network. Nepenthes implements this idea and emulates known vulnerabilities in network services: the tool detects incoming exploitation attempts, and sends data back to the attacker similar to a vulnerable machine. Based on an automated analysis of the attack payload, nepenthes extracts enough information

to actually download a binary copy of the attacking malware. The complete process is fully automated and we can collect autonomous spreading malware which spreads by exploiting common vulnerabilities in network services. We use a default setup of nepenthes with 21 different vulnerability modules.

As a second sensor for malware collection, we developed the *high-interaction honeypot* HoneyBow. A high-interaction honeypot is a conventional computer system, deployed to be probed, attacked, and compromised. Such a system has no production task in the network and no regularly active users. Thus it should neither have any unusual activities on the system nor generate any network traffic. These assumptions aid in attack detection: every interaction with the honeypot is suspicious by definition. HoneyBow uses this idea and is an approach to collect malware with high-interaction honeypots. Compared to nepenthes, this has the advantage that we do not need any signatures: we can use a conventional machine, patch it to an arbitrary patch-level, deploy the honeypot, and wait for successful compromises. The key concept is that a malware binary usually installs a binary copy of itself after a successful compromise. If we thus monitor the changes of the filesystem, we can detect an infection attempt and also obtain a binary copy of the malware sample itself. We observe the filesystem activity in real-time by monitoring all changes to the filesystem at the operating system level with the help of a tool we developed. Changes can for example include addition, deletion, or modification of files.

HoneyBow consists of the following three components: *MwWatcher* is the first part of the malware collecting tool. It is based on the essential feature of honeypots – no production activity – and watches the filesystem for suspicious activity caused by malware infections in real time. The tool itself is executed within a virtual machine (*virtual honeypot*), so that we can stop the virtual machine and then operate on an image of the filesystem: once MwWatcher detects a change to the filesystem, this is a clear sign of an intrusion attempt and the virtual machine is stopped. The disk image of the machine is then examined by the next tool: *MwFetcher*. This tool first generates a listing of all files from a hard disk image of an infected system detected by MwWatcher. Then this listing is compared to a file list from a clean system and all modified files are extracted since they could be an artifact of a successful intrusion. All binaries detected by MwFetcher are then submitted by *MwSubmitter* to the central malware sample library. The biggest advantage of such a high-interaction approach is that we do not need signatures of attacks. For example, we could capture samples of bots that use new attack vectors (e.g., Mocbot which uses MS06-040 for propagation [17]) which were not caught by nepenthes. This combined setup of two different honeypot sensor thus allows us to collect on the one hand samples that use well-known vulnerabilities (nepenthes) and on the other hand we can also collect malware samples that use new exploits (HoneyBow).

Because malware for the Windows operating system constitutes the vast majority of malware in the wild, we implemented HoneyBow only for Windows. On other platforms such as Linux or FreeBSD, the mechanism of real-time filesystem monitoring behind MwWatcher can also be implemented. The implementation details differ, but the principle is the same.

## 3.2 Malware Analysis

In the second phase, we want to automatically analyze the malware samples we have collected during the first phase. We need automation since we need to react on new threats in a timely manner. Two aspects are interesting to us: information about the botnet communication infrastructure and detection rates of common antivirus engines. To achieve these goals, we implemented *HoneyBox*, a fully automated malware analysis platform. The first component is *MwSniffer*, a behavior-based analysis tool similar to CWSandbox [20] and TTAnalyze [3]. The malware binary is executed and during runtime, the behavior of the application is observed with the help of a technique called *API hooking*: we intercept all API calls we want to monitor and insert our hooking function, which instruments the execution flow [10]. The technical implementation of API hooking is achieved via *inline code overwriting*, a common technique for this kind of analysis. The observations by MwSniffer include for example changes to the filesystem, access to the Windows registry, and network communication. Furthermore, we also limit some activities of the malware behavior (e.g., excessive network communication) within the hooking function to contain malicious behavior of the malware sample during the analysis process. One of the main aspects of MwSniffer is the identification of botnet communication channels. From the analysis reports, we extract all IRC-related information like username, channel name, and passwords together with the DNS- and IP-

related information and store it in the botnet tracking database. Please note that we let the bot connect to the actual C&C server, since this allows us to get detailed information about the communication protocol, e.g., we can also monitor botnets that use non-standard IRC servers.

The second component of HoneyBox is *MwScanner*, a tool that examines the detection rates of common antivirus (AV) engines for the collected malware samples. Nine major AV engines (e.g., Kaspersky, Trend Micro, and some local vendors such as Rising) are used for this test. Each collected sample is scheduled to be scanned several times: immediately after collection, after 1 day, after 3 days, after 2 weeks, and finally after 1 month. These results allow us to study the response rates of common AV engines to this kind of threat.

### 3.3 Botnet Tracking

In the third phase, we use the information extracted by MwSniffer to actually track the botnet. We implemented *HoneyBot* to perform this snooping on botnets. This tool is comparable to drone [18], botspy [13] and IRC Tracker [15]. Similar to these tools, we also join the channel used for Command & Control and monitor what is happening inside the botnet. Furthermore, the tool collects statistics about the botnet server (e.g., number of connected clients as reported by the server), the botherder (e.g., which commands were issued by whom), and status reports by bots. All collected information is stored in the botnet tracking database for further analysis.

### 3.4 Sensor Deployment

In total, we have deployed 17 malware collection nodes in 16 provinces of China. This unique network of sensors in one of the fastest growing networks allows us to study the botnet phenomenon from a completely different point of view compared to previous studies in this area. The distributed nodes have the following configuration: each node runs one nepenthes sensor, and two or three high-interaction Windows honeypots with Win2000 Pro, Windows XP, or Windows 2003 Server as an operating system. In total, we have thus about 50 sensors. The high-interaction honeypots are *virtual honeypots*, i.e., they are executed as virtual machines within VMware. Typically, only one IP address is assigned to each honeypot. However, on several nodes with extra IP resource, we assign 2-4 IP addresses from different ISPs to the honeypots. Furthermore, each sensor node also contains a *Honeywall*. A Honeywall is a transparent bridge used for containment purposes in order to minimize the risk associated with honeypots [19]. We run a customized version based on `Roo v1.1-hw1`, the official version released by the Honeynet Project. The first node is in production mode since June 19, 2006 and since July 2006, the complete distributed sensor network is up and running. We have two HoneyBot instances running to track IRC-based botnets. Every instance is able to track about four to five hundred IRC-based C&C channels and stores all collected raw information in the botnet tracking database for further statistical analysis.

## 4 Results

With the help of the botnet measurement setup described in the previous section, we have discovered 3,290 unique botnets on the China public Internet during the twelve month period between June 19, 2006 and June 10, 2007. *Uniqueness* is defined in this context as a unique combination of DNS name, port number and channel name. We have tracked these botnets throughout their lifetime, starting with the point in time where we first detected them. Based on these tracking data, we present several measurement results in the following paragraphs to give a detailed overview of the botnet phenomenon nowadays.

These results are complete within the scope our measurement setup was able to observe or capture. It reflects the current botnet phenomenon to a certain degree, but it does not represent a global snapshot of all botnets, bots, spreading mechanisms, or DDoS attacks found on the Internet. An open research question is still how many sensor nodes need to be deployed, and at which locations they should listen for what kind of malicious traffic [5]. For the future, we plan to increase the range and density of the distributed honeynet, and integrate it with a backbone threats measurement system. However, we think

that the sensor locations within one of the fastest growing, but traditionally opaque networks, gives our results a unique position not reported in any botnet-related study before. Furthermore, our results are based on almost one year of data, thus our measurements are based on solid data sources.

## 4.1 Malware Discovery Trends

Using nepenthes and our HoneyBow malware collecting tool, we are able to capture on average about 2,800 malware samples per day with the help of the 50 malware sensors distributed at 17 nodes. Figure 2 illustrates the number of collected samples per day for the period of almost twelve months. In the first few weeks, the complete system was still in the setup phase, and thus the number of collected samples was below 1,000. Since middle of July 2006, all sensor nodes are up and running. Between 2,000 and 4,000 samples are usually captured per day, with peak points of more than 7,000 samples. These peaks are mainly caused by outbreaks of new malware variants or polymorphic worms, as shown in later statistics in this section. On average, nepenthes collects 1925 binaries per day and HoneyBow 856.

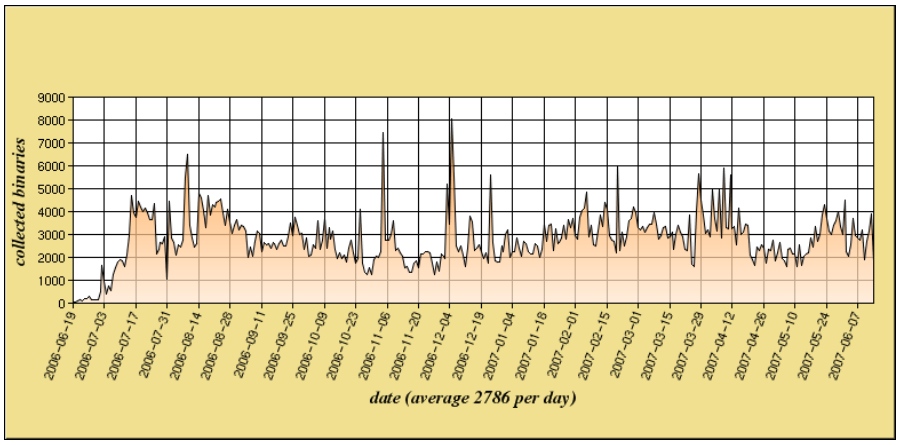



Figure 2: Temporal analysis of total number of collected malware binaries with all malware sensors between June 19, 2006 and June 10, 2007

Based on unique MD5 hash value, we have collected about 90,000 samples during the overall measurement period. The distribution of unique binaries collected per day is depicted in Figure 3. The spikes in the figure are mainly caused by polymorphic worms: in each iteration, such a worm changes certain parts of itself and thus the MD5 hash value is different. On average, we collect about 45 unique samples with nepenthes and 208 unique samples with HoneyBow every day. HoneyBow thus yields a higher number of unique malware samples, mainly because it does not rely on signatures.

All binaries were analyzed with MwScanner to collect further information with the help of common antivirus engines. In general, the detection rates are rather low. Even the best engine in our test detected only 92.8% of the samples. The detection rates vary between 50.4% and 92.8% for the nine engines. We also analyzed the distribution of malware families, as detected by the best antivirus engine in our test. Table 1 provides an overview of the results. Polymorphic worms like Virut and Allaple dominate our collection set. This is mainly due to our weak measurement of uniqueness: by using MD5 hash values, even slight differences in two binaries cause a completely different hash value. Thus we collect a large amount of cuplicates for polymorphic worms: for example, we collected 37,342 samples of Virut.a/Virut.b and 28,859 samples of Allaple, two common polymorphic malware families. All these samples have a unique MD5 hash value since each iteration of these worms is different.

For the four antivirus engines that detected most samples, we also analyzed the long-term trends in malware detection. We scan a collected malware binary after fixed time intervals with MwScanner as explained in Section 3.2. Figure 4 shows the temporal trend in detection rates for these four engines. With time, the detection rates grow, but none of the engines ever reaches complete detection.

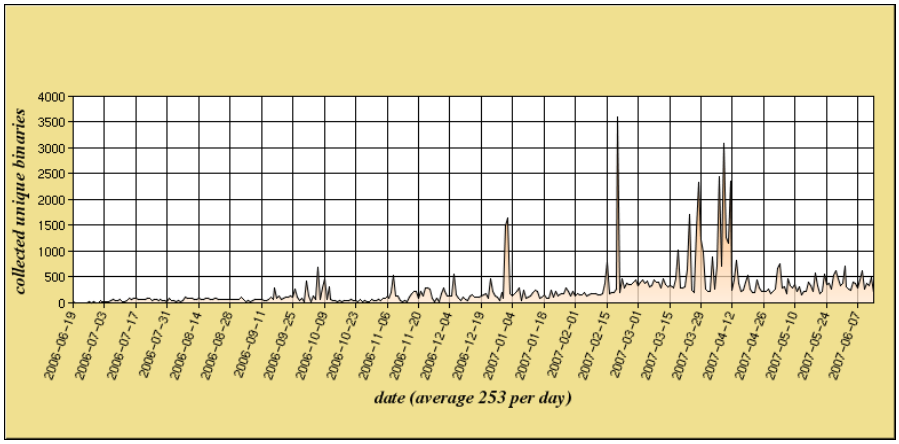



Figure 3: Temporal analysis of unique number of collected malware binaries with all malware sensors between June 19, 2006 and June 10, 2007

| Malware Family | Number of Samples | Percentage |
|---|---|---|
| Virut | 37,342 | 41.4 % |
| Allaple | 28,859 | 32.0 % |
| Rbot | 2,946 | 3.3 % |
| Parite | 1,861 | 2.1 % |
| SdBot | 1,101 | 1.2 % |
| Tenga | 922 | 1.0 % |
| Agent | 888 | 1.0 % |
| Expiro | 673 | 0.8 % |
| Others | 15,534 | 17.2 % |

Table 1: Distribution of malware families spreading in the wild

Since the malware samples we collected comprise worms, bots, and other forms of autonomous spreading malware, we need to identify all bots in order to study the corresponding botnets. Since our study focuses on IRC-based bots, we can identify these bots during the behavior-based analysis with MwSniffer: if we detect a successful connection to an IRC server during the analysis process, we have identified a bot and its corresponding botnet. In total, we could identify 5,645 IRC-based bots this way, a rate of 6.3% of the overall samples we collected. We also found many bot samples that could not establish a connection to the C&C server, mainly because mitigation took already place or other causes.

| Bot Family | Number of Samples | Percentage |
|---|---|---|
| Rbot | 1174 | 26.4 % |
| Virut | 664 | 15.9 % |
| SdBot | 335 | 7.5 % |
| Parite | 187 | 4.2 % |
| Bobic | 149 | 3.6 % |
| IRCBot | 134 | 3.0 % |
| PoeBot | 127 | 2.9 % |

Table 2: Overview of the top seven bot families spreading in the wild with alive control infrastructure

We analyzed the bot families with the help of the antivirus output to get a rough overview of the distribution of bot families in the wild. Table 2 presents the top seven bot families identified this way. Rbot, Virut and SdBot are responsible for about 50 percent of the botnets we found in the wild. Since the

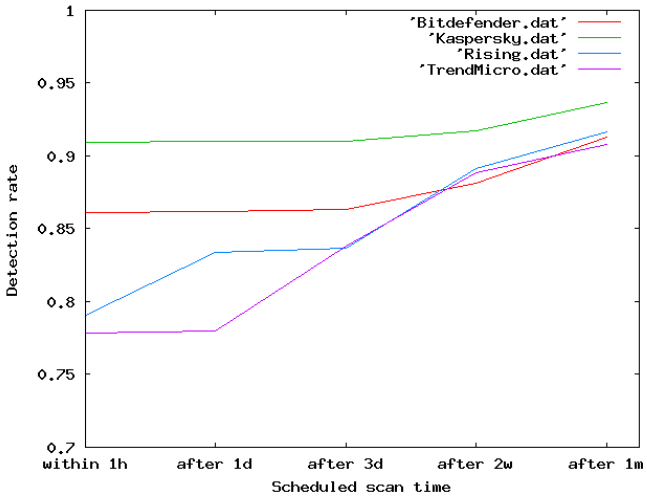



Figure 4: Trend in detection rate for four antivirus engines

source code of these bot families is publicly available, it is easy for an attacker to customize a bot and begin to spread a new variant. Therefore, thousand of variants exist for these bots and antivirus engines have a hard time to detect all of them. The discrepancy in Table 1 and 2 for families like Virut or Rbot is caused if a malware sample can not connect to the C&C server during the analysis phase with MwSniffer.

## 4.2 Botnet Command and Control Server Distribution

All 5,645 IRC-based bot samples were analyzed with MwSniffer to extract information about the communication channel used by botnets. In total, we could extract information about 3,290 unique botnet C&C channels this way. Figure 5 presents the trend in detection of botnet C&C channels for the whole measurement period. In the following, we present several statistics in order to characterize the distribution of the botnet control structure.

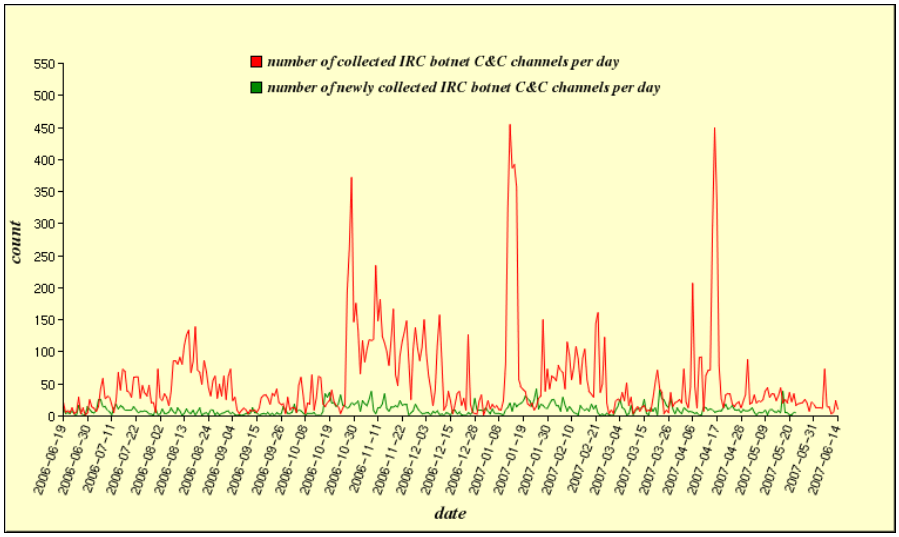
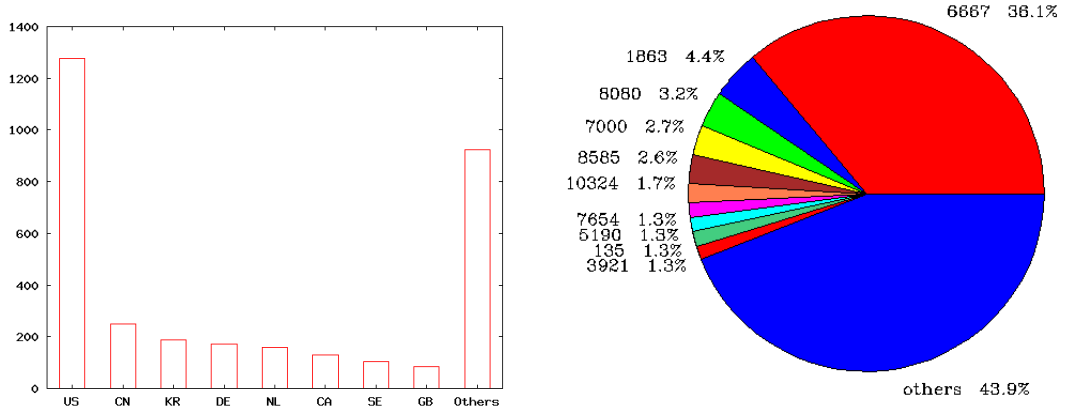



Figure 5: Temporal analysis of the number of detected and new botnet control channels for the period between June 19, 2006 and June 10, 2007

Figure 6a depicts the geographical distribution of discovered botnet C&C servers, generated with the help of the Geo-IP solution *IP2Location*. About 38.8% of them are hosted in the United States, taking the leading place. China, Korea, Germany and the Netherlands follow, and their percents range between 7.5% and 4.9%. The geographical diversity of the control servers is rather high, with more than 37.8% of all C&C servers hosted in other countries. These measurements are accurate to the extend of the Geo-IP software, which claims to have an accuracy of more than 99% on country level.




(a) Geographical distribution of C&C servers

(b) Distribution of network ports used for botnet C&C channel

Figure 6: Characteristics of 3,290 detected IRC-based botnet Command and Control servers

As shown in Figure 6b, about 36.1% of the discovered botnets use the standard IRC port 6667 to host the C&C channel. This confirms the trend that botnets move more and more away from using a standard configuration. Instead, most botnets nowadays use other TCP port for the communication channel. This helps the botherders to evade simple port-based detection and inspection tools. About 1.3% of the botnets we monitored even use TCP port 135, a port commonly used by Windows for file sharing.

Table 3 illustrates the distribution of the software used by the botherders to host the C&C server. The majority of botnets use `unreal`, a well known open source IRC server. Commonly, the tool was used in versions between v3.2.0 and v3.2.6, with v3.2.5 being the most used version in the wild. We also found other free IRC daemons such as `ircu`, `bahamut` and `hybrid`, but their frequencies are quite small compared to `unreal`. About 16% of active botnet C&C servers did not provide information about the software tool they were running. Benign IRC servers typically uses other IRC servers, e.g., *freenode* uses `hyperion`, *IRCnet* uses `IRCD`, and *QuakeNet* uses `Asuka`.

| C&C Server Software | Number | Percentage |
|---|---|---|
| unreal | 947 | 57.5 % |
| ircu | 128 | 7.8 % |
| bahamut | 117 | 7.1 % |
| hybrid | 39 | 2.4 % |
| ratbax | 15 | 0.9 % |
| Others | 143 | 8.1 % |
| N/A | 266 | 16.2 % |

Table 3: Distribution of IRC server software used by botnets

Our long-term measurement also allows us to study the *lifetime* of a botnet, i.e., how long a C&C server is typically used by the botherders. This time is mainly influenced by two events: *mitigation* and *alteration*. Mitigation means that the C&C server is taken offline, e.g., with the help of hosting providers or law enforcement. Alteration means that the botherder himself changes the structure of his botnet, e.g.,

by "moving" the bots to another server. The lifetime of a C&C server is an indicator of the flexibility of the command infrastructure. Figure 7 presents the distribution of C&C server lifetime we observed during our study. We start to count the lifetime once HoneyBot enters the control channel and starts with tracking. If we detect either mitigation or alteration of the server, we stop our measurement. The figure shows that a large percentage of servers last for a long time, i.e., several months or even longer. We found an average lifetime of about 54 days for the C&C servers we monitored. From the 3,290 botnets we found during our study, 378 are still alive in the middle of June 2007.

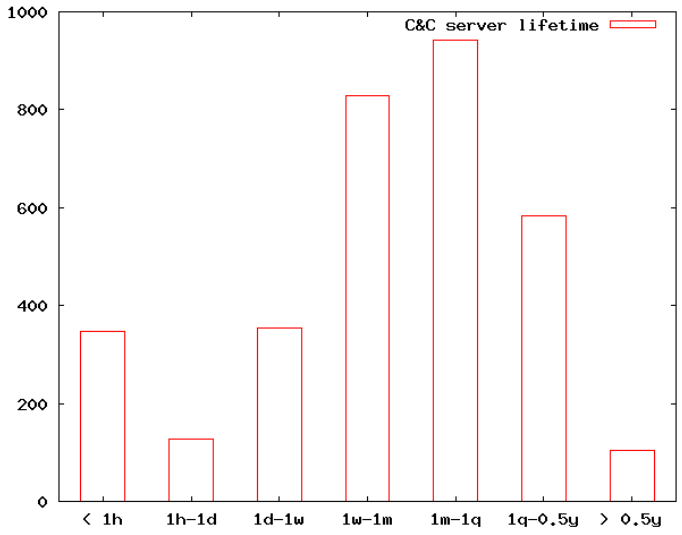



Figure 7: Measured lifetime of C&C servers

## 4.3 Botnet Size Distribution

The size of a botnet is an important metric to evaluate the threat posed by it. In general, it is hard to measure the real size of a botnet since attackers can obfuscate this information by modifying the C&C server or other means [16]. To obscure the size of the botnet, botherders can modify the C&C server to discard the responses of the IRC `/list users` and `/list #channel` commands. In addition, botherders can set the user mode of the bots to be invisible: the status messages of the bots are then not visible for all members of a channel. In such situations, the size of these botnets is hard to estimate.

The status information was not obscured by 1,904 of the 3,290 (57.9%) botnets we tracked in the wild. For these servers, we can estimate the number of online visible users on a C&C channel by keeping track of joining and leaving bots. In total, we observed about 1,520,000 distinct bot IDs in all botnet channels. From the 1,904 botnets, another 1,110 botnets (33.7% of total number) did not obfuscate the IP address by displaying an ID, but showed the actual IP address of a bot joining the control channel. This allows us to estimate the number of infected machines. In total, we observed about 700,700 distinct IP addresses. The biggest botnet we tracked during this period has controlled more than 50,000 hosts. Please note that these numbers do not take churn effects caused by DHCP or NAT into account and are thus only a rough estimate. Nevertheless, these numbers confirm the threat posed by botnets. An interesting finding is that we observed about 32,000 of the total 420,000 IP addresses (7.6%) in at least two different botnets, implying that these IP addresses are infected by at least two different bots.

We also actively queried the botnet C&C server to determine the botnet size. This measurement may not be accurate since the botherder can obfuscate the size information returned by the server. Table 4 shows the reported number of bots for the queries `/list users` and `/list #channel` and the status information reported by the C&C server on connect. Not all C&C servers responded to these queries, but we can get an overview of the typical size of a botnet with the help of such a query.

| | /list users *current global users* | /list #channel *online visible users in channel* | status information *reported by C&C server* |
|---|---|---|---|
| 5K+ | 64 | 4 | 57 |
| 2K - 5K | 107 | 20 | 57 |
| 501 - 2K | 214 | 65 | 164 |
| 101 - 500 | 121 | 141 | 257 |
| 11 -100 | 93 | 338 | 428 |
| 2 - 10 | 16 | 259 | 623 |
| 1 | 6 | 79 | 319 |

Table 4: Botnet size determined by actively querying C&C server

To determine the overlap between bot IP addresses and black lists, i.e., a list of misbehaving IP addresses which for example send out spam mails, we used the black list from Spamhaus[1]. In total, about 61,500 of the botnet IPs were also listed in the blacklist, corresponding to a coverage of about 8.8%.

Again, we determined the countries / regions that the individual bots belong to. The geographical distribution of all bots is shown in Figure 8. For our measurement setup, most bots are located in Brazil (15.1%). Following, several Asian countries / regions including China, Malaysia, and Chinese Taiwan host most of the compromised hosts. This indicates to some degree that the network security situation of developing countries is worse compared to developed countries. In addition, we see a scattered distribution and a wide diversification of infected hosts all over the world with 197 countries / regions total.

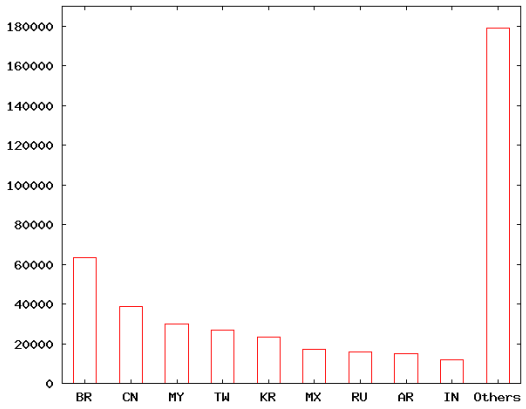



Figure 8: Geographical location of 420,000 bots, corresponding to the location of the victims

Figure 9 plots the number of online visible bots versus time for a typical botnet. As shown in the figure, the size of a botnet is changing over time. It ranges from only about 1,000 bots to more than 8,000 in this example. These dynamic changes can be observed in all botnets we tracked. Another observation is the strong daily diurnal pattern: the size of botnets follows daily patterns, caused by infected machines being powered on and off. This observation is consistent with the bot propagation model by Dagon et al. [7] and implies that the size of a botnet is constantly changing.

## 4.4 Botnet Activities Analysis

With the help of our botnet tracking tool HoneyBot, we monitored the C&C channels of the tracked botnets during their lifetime, and observed all commands issued by the botherders. Based on the observed

[1] `http://www.spamhaus.org/`

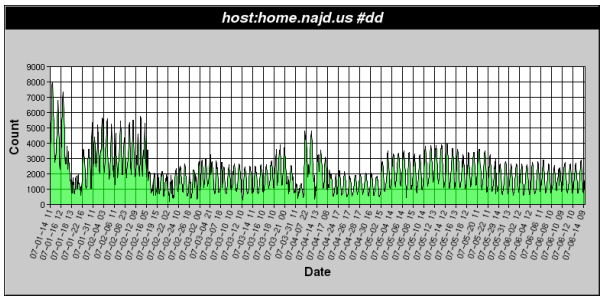



Figure 9: Diurnal patterns observed for estimated size of a given botnet

botnet activities, we classify botnet commands into the following eight command categories: *spreading*, *download and update*, *DDoS*, *information theft*, *server hosting*, *bot login*, *bot control*, and *botnet cloning*.

The absolute number of commands issued across all tracked botnet is summarized in Table 5. Spreading commands are the most popular since botherders commonly try to compromise additional machines to grow the size of the botnet. The DDoS category includes flooding-related commands used to attack other computers on the network. These attacks are very efficient with botnets and the second most commonly command issued. Botnet cloning is a special form of DDoS attacks where the controller orders each bot to connect a large number of clones to the victim IRC network. This kind of attack is efficient against other IRC servers. Download and update commands are used to distribute new malware or upgrade bots to new software versions. These four categories of botnet commands are the most commonly used in the wild, thus we give further results based on our analysis in the following sections.

| Command Category | Number of Events |
|---|---|
| spreading | 10,891 |
| DDoS attacks | 9,755 |
| botnet cloning | 5,621 |
| download/update | 5,583 |
| information theft | 3,809 |
| bot login | 1,863 |
| server hosting | 398 |
| bot control | 780 |
| Other | 107 |

Table 5: Distribution of commands issued by botherders for 3,290 monitored botnets

### 4.4.1 Spreading Activities

A typical botnet spreading command looks like `.advscan asn1smb 200 5 0 -r`. The first dot is called *command prefix*, it is used to identify commands issued by a botherder. `advscan` is a frequently seen spreading command, and the parameters includes the module name used to exploit target hosts, the number of concurrent scanning threats, amount of time between scanning threats, the time to scan (`0` means forever) and the scanning tactic (`-r` indicates random propagation attempts).

60.8% of the 10,891 spreading events we observed use *advscan* and its abbreviation *asc* as the spreading command. This observation is consistent with our finding from the source code analysis of several popular bot families. Furthermore, about 9.7% of the botnets use `.root.start` or `.ntscan` (5.5%) / `.scan` (4.5%) as spreading command.

The top six most commonly used spreading modules by our tracked botnets are shown in Table 6. The module names indicate what vulnerabilities are exploited during the propagation attempts. Vulnerabilities like *asn1* (MS04-007), *dcom* (MS03-026), *lsass* (MS04-011), *wks* (MS03-049/MS05-12), and

*pnp* (MS05-039) are rather old and patches are available for a long time, but nevertheless these exploits are still very popular for botnet spreading. In addition, weak passwords are another main cause of bot infections and used by the botherders to infected users.

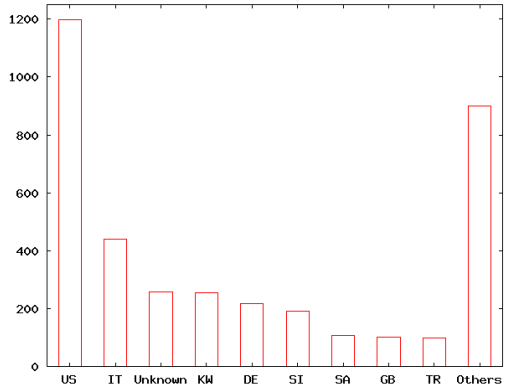
| Module Name | Number | Percentage |
|---|---|---|
| asn1 | 3310 | 30.4 % |
| dcom | 1213 | 11.1 % |
| lsass | 759 | 7.0 % |
| ntscan (weak password) | 643 | 5.9 % |
| wks | 40 | 0.3 % |
| pnp | 33 | 0.3 % |

Table 6: Distribution of exploitation attempts used by botherders

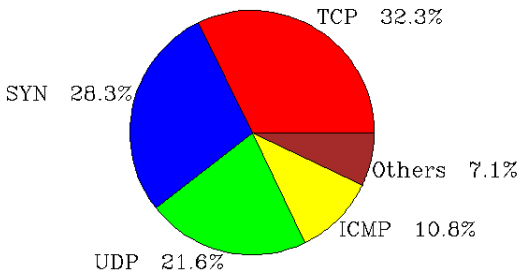
### 4.4.2 DDoS Activities

When the size of botnet increases to a certain range, it is often used by the botherders to launch attacks, e.g., to perform DDoS attacks against other systems. A typical DDoS command issued by a botherder looks like: `.syn 76.X.X.X 6667 5 20000` This command performs a SYN flood attack on TCP port 6667 for an amount of 20,000 seconds with 5 threads in parallel against the specified target.

In total, we observed 3,766 distinct victim IP addresses located in 76 countries which were attacked via DDoS commands. Figure 10a shows the geographical distribution of these victims. Most victims are located in the United States (31.8%) or Italy (11.7%). The other top countries include Kuwait, Germany, Slovenia, Saudi Arabia, and Great Britain. There are only six victim IP addresses located in China. Further analysis showed that the DDoS attacks targeting victims located in Kuwait, Slovenia, and Saudi Arabia are mainly launched by a small number of botnets. This distribution result cannot really reflect the general situation of DDoS attacks, but is an artifact of our monitoring scope.




(a) Geographical location of DDoS victims




(b) Distribution of DDoS attacks

Figure 10: Characteristics of observed Distributed Denial-of-Service attacks

Figure 10b presents the distribution of DDoS attacks types. TCP flooding, SYN flooding (a special form of attacks against TCP), and UDP flooding have quite close frequencies with 32.3%, 28.3% and 21.6%, respectively. The commands we observed most commonly are *!tcp* (8.8%), *.tcpflood* (8.3%), and *.wisdom.udp* (5.2%). The most active botherder attacked 335 distinct victims in 18 different countries during a period of about two months. On the other hand, we observed a victim that was attacked by nine different botnets during the whole measurement period.

### 4.4.3 Download and Update Activities

A typical download and update command is for example:

```
!d0wnl0ad http://.../sp4mb0ts/XXX.jpg C:\hook.exe 1
```

The command word is the obfuscated word *download* and the download files are commonly uploaded to free webspace providers. The third parameter is the local filename used for the downloaded file and the last parameter instruct the bot to also execute the binary once the downloaded has finished.

The most common download and update commands include *download* / *dl* (71.1%), *update* / *upd* (10.9%), and *wget* (8.7%). In total, we observed 5,583 events, which resulted in 2,219 unique URLs that we also downloaded. The geographical distribution of these update URLs is as follows: 41.8% are located within the US, 22.0% could be observed in China itself, and 5.9% are located in Russia.

Via this update mechanism, a botherder can for example install a new version of the bot. This new version commonly includes only small modification, e.g., a different executable packer to evade detection of antivirus engines or a new C&C server to move the bots to a new infrastructure. Furthermore, the botherder can install additional malware like keyloggers or SOCKS proxies on the victim's machine to use it for other nefarious purposes. We also monitored 129 *visit* commands, which causes a bot to visit a certain website. These commands are typically used for click-fraud.

# 5 Conclusion and Future Work

Botnets have become the first-choice attack platform for network-based attacks during the last few years. These networks pose a severe threat to normal operations of the public Internet and affect many Internet users. With the help of a distributed and fully-automated botnet measurement system, we were able to discover and track 3,290 botnets during a period of almost twelve months. Based on the analysis of the collected information, we gave a broad overview of the current botnet phenomenon and presented several characteristics and features of IRC-based botnets observed in the wild. This information can be used to get a deeper understanding of the threat posed by these networks. Furthermore, these results can also leverage research in the area of botnet detection, mitigation, and disruption: only by understanding the problem in detail, we can develop efficient countermeasures.

Our results show that the botnet problem is of global scale, with a scattered distribution of the control infrastructure and also a scattered distribution of the victims. Furthermore, the control infrastructure itself is rather flexible, with an average lifetime of a C&C server of about 54 days. We thus need automated counter measures to stop this threat.

In the future, we need to extend such a measurement study for non-IRC-based botnets to also understand that phenomenon in more detail. Botnets seem to shift away from IRC to protocols like HTTP, Peer-to-Peer-based protocols, or custom protocols. We also need to study these types of remote control networks in order to be able to develop efficient counter measures for these threats.

Measuring the size of a botnet is still an open research problem. By counting the joining and leaving bots within a channel or querying the IRC server for this information, we can at least estimate the actual size as presented in this study. But if the botherder modifies the server, this kind of information is often not available. Measurements like DNS snooping can be used to get a rough overview of the botnet size [15], but the reliability and accuracy of these measurements is still not clear.

# Acknowledgments

We would like to thank Christian Gorecki and Felix Freiling for valuable comments on previous versions of this paper.

# References


[1] P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. C. Freiling. The nepenthes platform: An efficient approach to collect malware. In *Proceedings of 9th Symposium on Recent Advances in Intrusion Detection (RAID'06)*, pages 165–184, 2006.

[2] P. Barford and V. Yegneswaran. *An Inside Look at Botnets*, volume 27 of *Advances in Information Security*, pages 171–191. Springer US, 2007.

[3] U. Bayer, C. Kruegel, and E. Kirda. TTAnalyze: A tool for analyzing malware. In *Proceedings of 15th Annual Conference of the European Institute for Computer Antivirus Research (EICAR)*, 2006.

[4] J. Canavan. The evolution of malicious IRC bots. In *Proceedings of the Virus Bulletin Conference*, 2005.

[5] E. Cooke, M. Bailey, Z. M. Mao, D. Watson, F. Jahanian, and D. McPherson. Toward understanding distributed blackhole placement. In *Proceedings of 2004 ACM Workshop on Rapid Malcode*, 2004.

[6] E. Cooke, F. Jahanian, and D. McPherson. The zombie roundup: Understanding, detecting, and disrupting botnets. In *Proceedings of SRUTI'05*, pages 39–44, 2005.

[7] D. Dagon, C. Zou, and W. Lee. Modeling botnet propagation using time zones. In *Proceedings of the 13th Annual Network and Distributed System Security Symposium (NDSS'06)*, February 2006.

[8] F. Freiling, T. Holz, and G. Wicherski. Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks. In *Proceedings of 10th European Symposium On Research In Computer Security (ESORICS05)*. Springer, July 2005.

[9] J. B. Grizzard, V. Sharma, C. Nunnery, B. B. H. Kang, and D. Dagon. Peer-to-peer botnets: Overview and case study. In *Proceedings of the Workshop on Hot Topics in Understanding Botnets*, April 2007.

[10] G. C. Hunt and D. Brubacker. Detours: Binary Interception of Win32 Functions. In *Proceedings of the 3rd USENIX Windows NT Symposium*, pages 135–143. Advanced Computing Systems Association, 1999.

[11] C. Kalt. Internet relay chat: Architecture, April 2000. Request for Comments: RFC 2810.

[12] A. Karasaridis, B. Rexroad, and D. Hoeflin. Wide-scale botnet detection and characterization. In *Proceedings of the Workshop on Hot Topics in Understanding Botnets*, April 2007.

[13] C. R. F. Overbeck. Efficient Observation of Botnets. Master's thesis, RWTH Aachen University, July 2007.

[14] N. Provos and T. Holz. *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*. Addison-Wesley, July 2007.

[15] M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A multifaceted approach to understanding the botnet phenomenon. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, pages 41–52, New York, NY, USA, 2006. ACM Press.

[16] M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis. My botnet is bigger than yours (maybe, better than yours). In *Proceedings of HotBots'07*, 2007.

[17] J. Stewart. Mocbot/MS06-040 IRC Bot Analysis, August 2007. Internet: http://www.secureworks.com/research/threats/mocbot-ms06040/.

[18] The Honeynet Project. Know Your Enemy: Tracking Botnets, March 2005. Internet: http://www.honeynet.org/papers/bots/.

[19] The Honeynet Project. Honeywall CDROM, March 2007. Internet: http://honeynet.org/tools/cdrom/.

[20] C. Willems, T. Holz, and F. Freiling. CWSandbox: Towards automated dynamic binary analysis. *IEEE Security and Privacy*, 5(2), 2007.