Technische Universität Berlin
Seminar Internet Measurements
Betreuer: Gregor Maier

A Multifaceted Approach to Understanding the Botnet Phenomenon

Abstract

The following text is a summary of the original paper: “A Multifaceted Approach to Understanding the Botnet Phenomenon” from Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose and Andreas Terzis from the Computer Science Department in the John Hopkins University [1]. The text tries to clarify mysteries within botnets. Its authors set up a measurement system, collected and tracked botnet activities and finally analysed the results.

Javier Zugasti Raposo
javier.zugasti@gmail.com
(Version from 2nd June 2007)

1. Introduction

The original paper is an approach to clarify some of the mysteries within botnets. There are two main unknown points when talking about botnets. These are their prevalence on the Internet and their life cycle. The investigators who developed this research constructed a multifaceted and distributed measurement infrastructure in order to solve these points. Throughout a period of three months they used this infrastructure to track 192 botnets, consisting some of them of up to a few thousand infected hosts. Amazing results appeared and these were that 27% of all malicious attempts observed could be attributed to botnet-related spreading activity. Evidence of botnet infections was discovered in 11% of the 800.000 DNS domains examined.

A simple definition for botnet is that they are networks of infected end-hosts, called bots, which are under the control of a person: the botmaster. Some other classes of malware were mostly used to demonstrate technical knowledge among hackers but botnets are used for malicious activities such as extortion of Internet businesses, identity theft, spamming and software piracy. Botnets recruit vulnerable machines to add them to their system. Some of the methods used by botmasters are similar to the ones used by other classes of malware. It can be said that its main characteristic is the use of *C&C*[1] channels, within an *Internet Relay Chat (IRC)*[2] protocol, for the spreading of commands to the bots.

The procedure followed in this measurement is firstly the development of an infrastructure to capture and track botnets and secondly an analysis of the measurements. The infrastructure developed consists of three different parts which interact with each other and are: malware collection points for the capturing of binaries, IRC trackers to analyse the phenomenon from the inside and DNS probing in order to asses the botnet prevalence on the net.

The following summary is divided into four sections. The first section gives a general explanation about botnets and their life cycle. The second one is an explanation of the measurement methodology done by the investigators. In the third part a summary of the main points from the analysis of the results is given. The author's conclusion is shown on the last section.

[1] *C&C* channel: command and control channel. They work over a large variety of topologies and use many different protocols for communication.
[2] IRC protocol: It is an open protocol that uses TCP and sometimes SSL which was thought for offering real-time Internet chat or synchronous conferencing. It supports many-to-many communication but also one-to-one communication.

2. Botnets

The following picture [1] shows the life-cycle of a botnet infection and the contact with the botmaster:

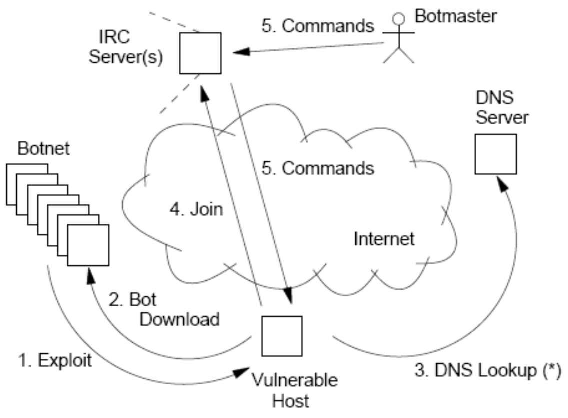


Figure 1: The life-cycle of a typical botnet infection.

Infection strategies used by the masters are often seen when examining other malware such as self-replicating worms, e-mail viruses, etc. but also can be spread by making a victim execute some form of malicious code on his machine. Many email attachments are simply these executable files.

Once the infection has taken place, the *shellcode*[3] is executed and the bot binary is downloaded from some location (usually from the same machine were the original code came from). The binary is then installed immediately in the background without the victim's knowledge. From this point on, every time a reboot occurs the binary restarts itself.

The next step taken by a new bot is to contact a DNS server for the resolving of the DNS name of the IRC server (the IRC server's name is given in the executable and a DNS query is made to acquire the server's IP address). This step allows the master to retain control of the net also if the IP address associated with the DNS name of the IRC server gets black-listed. The fact that IRC channels allow several forms of communication as well as data dissemination and that many open-source implementations are available [4] make this protocol just suitable for botmasters. As the *C&C* channel is also specified in the binary, the bot can now establish an IRC connection with the server and join the given channel. For this purpose, three steps of authentication are required: first of all the bot authenticates itself to the server with the PASS message, then it also has to authenticate itself with a password

[3] In my opinion the word shellcode is used on the original paper referring to a piece of executable code or script. A suitable definition on this context would be: Shellcode is the name given to small pieces of assembly language which are used to launch shells, typically as a result of a buffer overflow. (A buffer overflow is a technical term which is used to describe the act of filling a piece of memory with more data than it is designed to hold, this is a commonly used technique in security exploits). [http://shellcode.org]

to the master in order to join the channel. Lastly the botmaster also needs to authenticate himself to the bot before being able to send any command. The first two aim to keep outsiders away from the *C&C* channel, and the last one prevents the bots from being overtaken by other masters.

When the join has successfully occurred, the bot executes the channel's topic, which contains the default commands that every bot has to execute. Often it occurs that all bots on the channel are able to hear every exchanged message and this characteristic is used on this paper for the acquisition of insider information. However, sometimes broadcasting is not allowed to prevent saturation.

3. Measurements

There are three different phases in the measurements. The first one is the malware collection and its goal is the collection of as many as possible malicious binaries. For this goal a modified version of *Nepenthess Platform* [2], a honeynet[4] [3] and a download station are used. The second phase of the measurements is the binary analysis. In it the binaries collected on the first step are analysed. The last phase is the tracking of botnets, which is divided into IRC- and DNS-tracking, for obtaining results on prevalence, botnet traffic, botnet sizes, etc (results are shown on section 4).

The following diagram [1] shows the overall measuring structure allocated on a darknet and connected through a gateway.

[4] It is a net consisting of several honeypots dedicated usually for intrusion-detection systems. Traps are set to detect attemps of unauthorized use of services. A honeypot is a computer that appears to be part of a network but is protected and seems to contain important resources that would be of value to attackers.

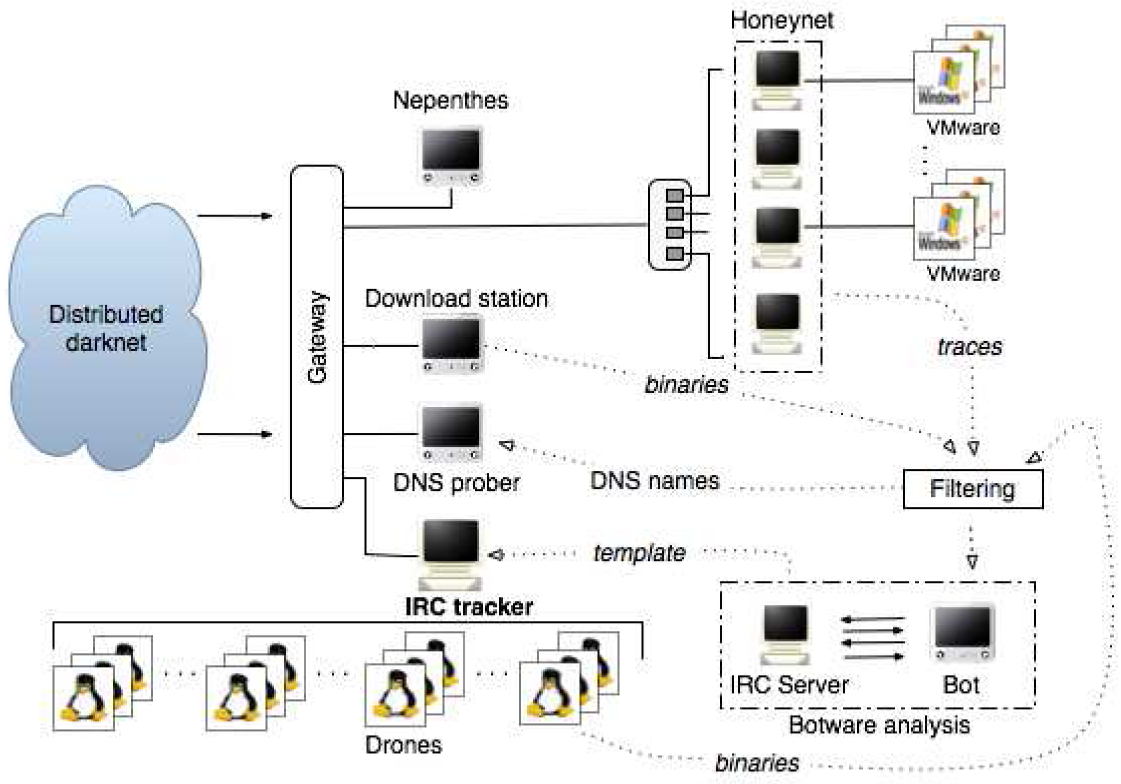


Figure 2: Overall data collection architecture

3.1 Malware Collection

The goal of this part is to collect as many malicious binaries as possible. In order not to miss any part of the IP space in which botnet activity occurs, a distributed darknet is used. The darknet resembles allocated but unused parts of the IP space. The results obtained at the darknet are then extrapolated to the whole internet. 14 distributed *PlanetLab test bed*[5] nodes with access to the darknet IP space are also required. In the distributed darknet a modified version of the *Nephenthess Platform [2]* was deployed. The platform mimics the answers generated by the victims to collect, when possible, *shellcodes.* For the retrieving of binaries a list of URLs to be downloaded is created and sent to a machine designed for this task (URLs to be accessed are contained in the shellcodes).

Complementing the tasks of *Nephenthess* a honeynet is also necessary for the catches that could have been missed. The honeynet consists of some honeypots which run an unpatched instance of WinXP each. The honeypot instances are in different IP spaces on a virtual LAN. These pots try to establish connections with the IRC servers. When the virtual machines are reimaged, the

[5] *PlanetLab* is a global research network that supports the development of new network services. More than 1000 researchers have used to develop technologies for distributed storage, network mapping, peer-to-peer systems and query processing.

disk contents are compared to clean images of Windows and binaries are retrieved.

As can be seen on the diagram a gateway is necessary to engage all parts of the measurement system. Its first task is to forward the incoming traffic to the different parts of the darknet. It does it to several /24 prefixes and keeps changing to fully cover all the IP space. Half of the prefixes are sent to *Nephenthess* and the other half to the honeynet. Network Address Translation (NAP) is used so that not so many honeypots are needed. Of course if there were no gateway the honeypots could infect each other; this means that it also acts as a firewall. As said before each honeypot is configured on a different VLAN and traffic across them is terminated at the gateway. After some filtering stages, the remaining traffic is queued to a detection process configured to allow the pots to follow the infection sequence.

Once a connection from a honeypot is established, the detection module just has to search at the application-level for the IRC protocol strings used at the handshake such as *NICK, JOIN, USER, etc.* When these results are found, the module creates a record on the IRC session and as the honeypot has already accomplished its duty, it can now be reseted (*RST*). An interesting detail to take into notice is that as the analyses are made at the application level, traffic on non-standard ports can also be detected.

Some other functions are also implemented by the gateway. These are re-imaging of the honeypots, pre-filtering and control for the download station and running a DNS server for the resolving of DNS queries from the pots.

3.2 Binary Analysis

When the binaries have been collected, a Graybox[6] Analysis Tool is used for the analysis and extraction of features of suspicious binaries. There are two different parts in this analysis which are: a network level analysis, made by deriving a network fingerprint of the binary, and an application level analysis in which IRC features are extracted.

To help the analysis, a part of the private network is set to contain a server and a virtual client.

Phase 1 (network level analysis)

All traffic logs are processed in the server we created for the extraction of a network fingerprint. This fingerprint contains the targets of the DNS requests, the destination IP addresses, the ports used, and whether or not a scanning activity had occurred[7].

[6]The Graybox Testing Methodology is a software testing method used to test applications. Graybox method can be applied in real-time using software executing on the target platform.

[7] Scanning is said to occur when there is an attempt to contact more than 20 different destinations on the same port during a given time.

Phase 2 (application level analysis)

At the server which was created, a modified version of *UnrealIRC daemon*[8] [4] is instantiated. As we collected the observed ports on phase 1, we can now set the IRC server to listen on all of them. When observing an IRC connection, an IRC-fingerprint is created. It consists of a password to establish the connection, the nickname and username chosen by the bot, the modes set and the channels to be joined. Having both these fingerprints, makes possible joining a botnet in the wild. The last step, necessary for the botmaster not to recognize the trap (it is a honeypot which is connecting to the net and not a real bot) and for being able to understand his commands, is learning the botnet's "dialect". The first thing to do is to make a bot connect to our server and force him to join a specific channel. Some commands are sent to the bot (these commands are taken from the honeynet traces that are known), and we then learn to mimic the bot's behaviour. As said above on the second point, often bots need an authentication coming from the master to respond to his commands. Sometimes it can be extracted from the log if the bot was observed on the honeynet. If not, as bots usually try to parse the server channel's topic, the IRC server is set to send this command on the fly so that bots parse it and no authentication is therefore required. At the end of the process the *template* with the bot's dialect is obtained.

3.3 Tracking of Botnets

The botnet tracking is developed in two different ways:

IRC Tracker

The functionality of the IRC tracker used is quite simple. From the template created at the binary analysis, we develop an IRC client (the tracker), which can join a specified IRC channel and answer queries. With both template and IRC fingerprint, the tracker instantiates a session to the IRC server. It then pretends to follow exactly all the commands sent by the botmaster and consequently sends him the correct replies. In order to appear real and succeeding on the goal, the IRC tracker must look real. For this duty, traffic must be pre-filtered before being sent to the master for the suppression of inappropriate information. This filtering is developed in real time while the software is executed. The IRC tracker must also be able to mimic the state changes for a correct responding. So that it is not suspicious, the tracker joins and leaves the channel at random intervals. It leaves the channel for a few minutes and then rejoins with a different *USER*.

DNS Tracking

Knowing that most bots send DNS requests to resolve IP addresses of the servers, a large number of caches from DNS servers are probed[9] [6]. As a

[8] UnrealIRCd is an open source IRC server. Its development started on 1999. For our concern, a modified version of it is used. Daemon is a computer program that runs on the background.

[9] It is not said on the paper how the DNS Servers are selected. References to other papers on DNS studies are done.

measure we take cache hits[10] into account. This is to be able to distinguish between servers contacted by infected clients and by not infected ones. For each IRC server, the caches of the DNS servers are probed and the cache hits are recorded. This kind of measurement is not quite exact because a cache hit only indicates that at least one bot sent a DNS request, but it does not tell how many bots did query the server within the TTL or if at all.

4. Results and Analysis

The results reported are based on data collected over a period of, more a less, 3 months. It includes traffic captured at the darknet, IRC logs gathered over this period of time and the results of DNS cache hits from tracking 65 IRC servers.

4.1 Result 1: Prevalence of the Botnet Phenomenon

Traffic share

The following figure shows the total incoming packets to the darknet versus the ones coming from known botnet spreading activity. Most of this traffic is thought to be coming from the scanning of new victims.

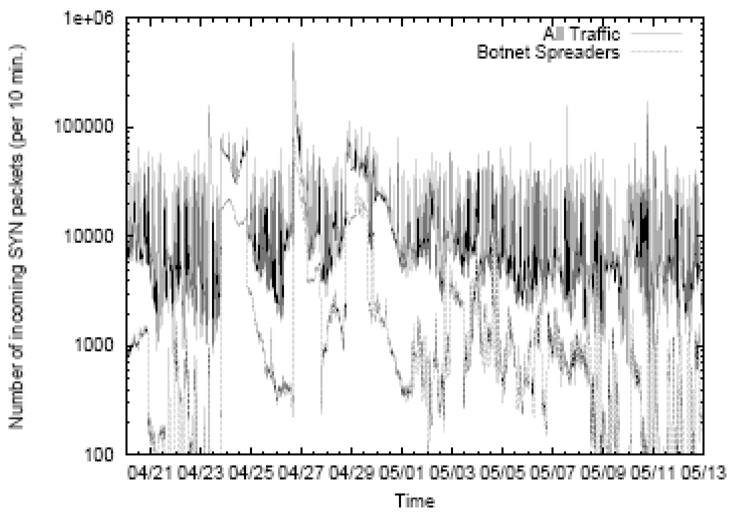


Figure 3: Time series of incoming SYN packets to the darknet

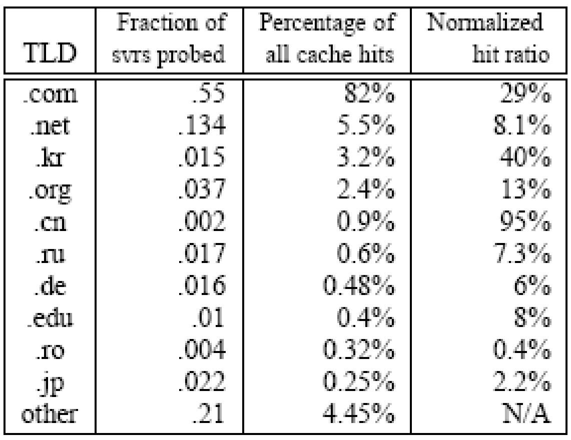
As a first result it can be said that 27% of the SYN packets received are supposed to come from botnet activity. If we instead take a look on the common ports used by botnet spreaders the percentage increases up to 76% (this fact cannot be taken as a very exact one because many other types of malware might also use these ports for their activity). We highlight the fact that traffic peaks are synchronized with bot traffic peaks which gives us a good measure of the amount of bandwidth and resources wasted by this damaging activity. Even

[10] A cache hit means that a client machine has queried the DNS server within the TTL of its DNS entry.

though it cannot be taken as an exact value we can say that botnet-related activity occupies at least a 27% of overall traffic in the darknet.

Prevalence

The prevalence results and analysis are extracted from the DNS probing experiment. As said before cache hits for 65 IRC servers were tracked over a large amount of DNS servers (800.000). Amazing results appeared, and these are that 85.000 severs presented at least one detected activity. This represents 11% of the total amount. Within the servers probed, the *.com* domain represents 55%, and it registered 82% of all cache hits detected. This means that 29% of *.com* domains in the dataset have registered botnet's activity with at least one hit. Other cases from other important domains are shown on the following table. The first column shows different domains, the second one shows the fraction of each server within the ones probed (this means for example that 55% of the domains probed were *.com* ones), the third column shows the percentage of all cache hits (this means for example that 82% of all hits detected occurred on a *.com* domain). The last column shows a normalized hit ratio (29% of the *.com* servers in the data had at least one cache hit).



TLD	Fraction of svrs probed	Percentage of all cache hits	Normalized hit ratio
.com	.55	82%	29%
.net	.134	5.5%	8.1%
.kr	.015	3.2%	40%
.org	.037	2.4%	13%
.cn	.002	0.9%	95%
.ru	.017	0.6%	7.3%
.de	.016	0.48%	6%
.edu	.01	0.4%	8%
.ro	.004	0.32%	0.4%
.jp	.022	0.25%	2.2%
other	.21	4.45%	N/A

Table 1: TLD statistics of DNS servers supporting clients envolved in at least one botnet

4.2 Result 2: Botnet's Spreading and Growth patterns

Email, web and active scanning are known to be the usual recruiting methods used on botnets. Between them, scanning is the most effective. However, different botnets show different methods of scanning but most can be classified into two types of means of scanning: The first group are worm-like botnets which continuously scan certain ports. The second group contains botnets that can vary their ways of scanning. This group uses different types of scanning algorithms (localized, uniform…). They also require a "start scanning" command from the master over the IRC channel. A minority of the bots studied belonged to the first type and they just kept scanning the IP space searching for new victims. The amount of victims has a continuous growth because of the constant scanning activity. Because of the intermittent and changing behaviour on the second type bots, their activity is much more accurate and difficult to track and to fight against. We can differentiate between localized scanning and targeted scanning[11]. Whereas localized scanning targets mostly /16 subnets, targeted scanning is aimed to /8 prefixes. Botmasters have set their armies to be able to select between several scanning settings. Scanning rates, number of threads, number of packets and scanning duration are some examples of these settings.

Localized scanning occupies a 66% of scanning activity and targeted one a 32%.

For the examination of growth patterns a first attempt is the plotting of DNS cache hits for individual botnets. Three different patterns were observed between different botnets. These are semi exponential growth, Staircase growth and Linear growth. The first one is due to worm-like infections. As an example of this type we can take a channel topic that continuously scanned a given port. As the number of victims kept increasing and the topic remained unchanged, a semi exponential pattern appeared. The second one is caused by bots that performed a scanning activity on a short period of time, then they stopped their activity and after a while resumed it again. Therefore this staircase pattern was shown. The last pattern was observed within bots that used a time-scoped scanning, targeting specific network prefixes and not others. It should be also noted that 52% of botnets broadcasted their messages to the *C&C* channel.

Results extracted from the IRC tracking showed concordant trends with the DNS analysis.

4.3 Result 3: Botnet Structures

Four different structures were detected in the experiment. To the first type belong 70% of the botnets studied and they are those whose bots always connect to a single IRC server. It is predominant among small botnets with population of no more than a few hundred of machines. The second type bases on the idea that different IRC servers can be connected to form an IRC network supporting a great number of users. They can be called bridged botnets and 30% of the botnets studied belong to this group. Half of them only bridged 2 servers at the same time. The third structure is quite not too clear and it is the one formed by apparently unrelated bots that later on were found to belong to the same botmaster. Finally some instances were seen in which bots downloaded the same binary which at the same time moved them to a different IRC server.

4.4 Result 4: Effective Botnet Sizes

Even though we said in the previous section that botnets can be thought to have even more than 15.000 bots, it is difficult to believe that the servers would be able to handle so many users. Within single- or few-servers-structures a distinction between botnet's fingerprints and the amount of bots connected to a channel at the same time has to be done. This last term we call it the botnet's

[11] The basic idea of localized scanning method is that if vulnerable hosts are clustered, an infected host searching for local hosts would have a higher probability to find a target than random guessing [7].

effective size. Figure 4 shows the number of simultaneous bots connected at different periods of time on three different servers.

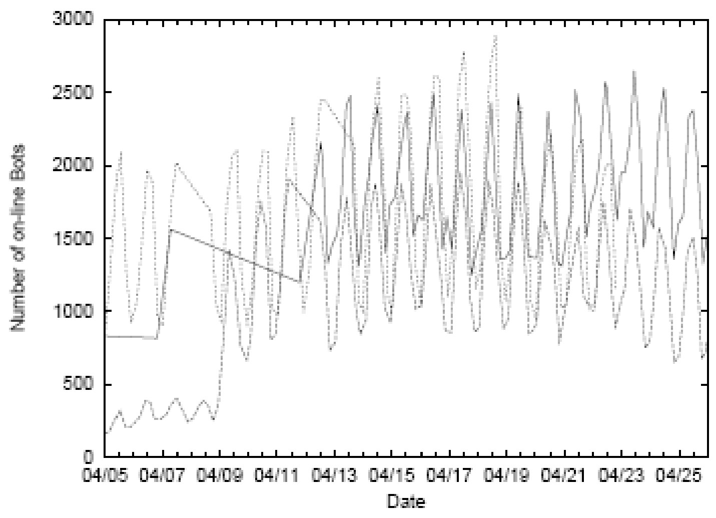


Figure 4: Evolution of effective size for three of the botnets in the study

The average footprint's size for the case shown on the figure was bigger than 10.000 but as can be seen, not more than 3.000 hosts were online at the same time. It can be deducted then that the maximum size of the online population is much smaller than the one read on the fingerprint. As the effects don't seem to decrease at any time, it can be said that botmasters have many different headquarters with different times.

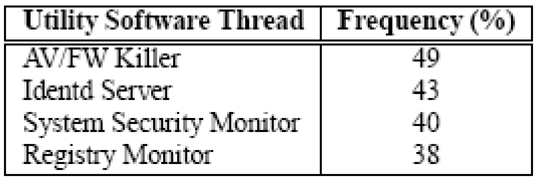
4.5 Result 5: Lifetime

The long lifetime of a typical botnet causes the big difference between the footprint and the effective size. Also the high *churn rate*[12] contributes to this fact. Botnet IRC channels show a high churn rate, meaning that bots do not stay long on the channels. The average staying time is 25 minutes even though 90% of them stayed for less than 50 minutes. The most likely reason for this is that masters may command their bots to leave every so often. The bots which actually make the average lifetime higher are the proper masters. Often it was also seen that bots that were shut down by the botmaster still remained their activity for more than a month before disappearing. After the three months of experiment 84% of the IRC servers were still active and 55% of them still showed scanning activities. Curious was that the longer living botnets were those that used static IP addresses instead of DNS names.

4.6 Result 6: Software Taxonomy

The following table shows the different thread types captured in the experiment:

[12] The churn rate refers to the number of peers leaving a system during a given period of time.



Utility Software Thread	Frequency (%)
AV/FW Killer	49
Identd Server	43
System Security Monitor	40
Registry Monitor	38

Table 2: The percentage of bots that launched the respective services on the victim machines.

50% of the bot threads[13] ran *AV/FW Killers*, which disable anti-viruses and firewalls to make the systems vulnerable for the botmaster's commands. Others ran *identd* utilities, which are used for collecting the user's TCP connection identity and therefore being able to join IRC channels that require it as authorization. Registry Monitor threads are used to alert bots of possible tries to disable them. System Security Monitor threads are used by the botmaster for strengthening of their armies.

Regarding to the operating systems affected, it can be said that even the last version of WIN XP was attacked. Knowing this it was an interesting thing to do to check the anti-virus responses against botnet activities. For this task the latest versions of *Norton*'s anti-virus and *ClamAV* were used. Fortunately these two protection systems were found to be quite effective, having detected 137 out of 192 (*ClamAV)* and 179 out of 192 (*Norton).*

4.7 Further Results

It was found in the experiments that many different botmasters existed. Not only novices but also really sophisticated. It was also seen that botmasters often share information of which prefixes they should not scan. They try to keep the chatter on IRC channels to a minimum and also make their own bots to be aware of possible misbehaviour and to search for victims with important resources. Also migration commands were detected in the study. Migration can be deduced from the following figure. The population of the first botnet suddenly decreases to a minimum and simultaneously the population of the second one increases to a maximum. This is due to a migration.

[13] Threads are understood here as different processes that run in the machines of the victims.

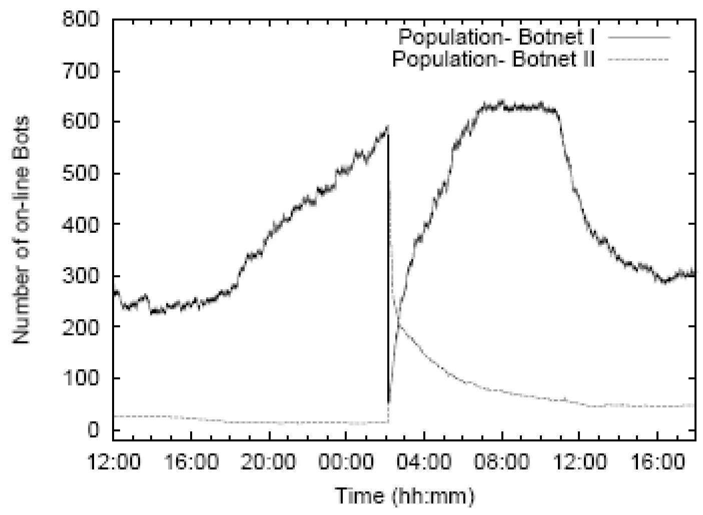


Figure 5: Migration of a botnet as observed by the IRC Tracker.

The tracked botnets were used for different tasks, shown on the next table:

Command Type	Frequency (%)
Control	33
Scanning	28
Cloning	15
Mining	7
Download	7
Attack	7
Other	3

Table 3: Relative Frequency of commands observed across all tracked botnets.

5. Author's Conclusion

Botnets represent one of the most severe threats to the Internet. Despite of this fact, our knowledge of botnet behaviour is incomplete. Results show that botnets are a major contribution to the overall unwanted traffic on the Internet. Botnet scanning activities are different from that seen by autonomous malware because of its manual handling. The dominant protocol used for botnet's activities is kept to be IRC because of its versatility. The effective sizes of the botnets studied go from a few hundreds to a few thousands of simultaneous online bots. Although the number of fingerprints is much larger. The discrepancy is due to high levels of churn rate; a bot's average channel occupancy is less than a half on an hour. It was also seen that the level of sophistication is becoming larger and larger and includes self-protection mechanisms.

6. References

[1] Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose, Andreas Terzis. Computer Science Department, Johns Hopkins University. *A Multifaceted Approach to Understanding the Botnet Phenomenon.*

[2] The Nephenthes Platform: An efficient Approach to Collect Malware. *http://www.mwcollect.org*

[3] Honeynet Project and Research Alliance, *http://www.honeynet.org/papers/bots*

[4] The UnrealIRC Team. *http://www.unrealircd.com*

[5] Graybox Software Testing in Real-Time in the Real World, *http://www.cleanscape.net/docs_lib/paper_graybox.pdf*

[6] DNS Cache Snooping, *http://www.sysvalue.com/papers/DNS-Cache-Snooping/files/DNS_Cache_Snooping_1.1.pdf*

[7] Understanding Localized-Scanning Worms, *http://users.ece.gatech.edu/~zchen/localized.pdf*

[8] English Wikipedia. http://www.en.wikipedia.org