


May 29th, 10:40 AM

Botnet Forensic Investigation Techniques and Cost Evaluation

Brian Cusack

Junewon Park Digital Forensic Research Laboratories, Auckland University of Technology,
brian.cusack@aut.ac.nz

Follow this and additional works at: <https://commons.erau.edu/adfsl>

 Part of the [Aviation Safety and Security Commons](#), [Computer Law Commons](#), [Defense and Security Studies Commons](#), [Forensic Science and Technology Commons](#), [Information Security Commons](#), [National Security Law Commons](#), [OS and Networks Commons](#), [Other Computer Sciences Commons](#), and the [Social Control, Law, Crime, and Deviance Commons](#)

Scholarly Commons Citation

Cusack, Brian, "Botnet Forensic Investigation Techniques and Cost Evaluation" (2014). *Annual ADFSL Conference on Digital Forensics, Security and Law*. 9.
<https://commons.erau.edu/adfsl/2014/thursday/9>

This Peer Reviewed Paper is brought to you for free and open access by the Conferences at Scholarly Commons. It has been accepted for inclusion in Annual ADFSL Conference on Digital Forensics, Security and Law by an authorized administrator of Scholarly Commons. For more information, please contact commons@erau.edu.

EMBRY-RIDDLE
Aeronautical University™
SCHOLARLY COMMONS

(c)ADFSL



BOTNET FORENSIC INVESTIGATION TECHNIQUES AND COST EVALUATION

Brian Cusack
Junewon Park Digital Forensic Research Laboratories
Auckland University of Technology
Auckland, New Zealand
brian.cusack@aut.ac.nz

ABSTRACT

Botnets are responsible for a large percentage of damages and criminal activity on the Internet. They have shifted attacks from push activities to pull techniques for the distribution of malwares and continue to provide economic advantages to the exploiters at the expense of other legitimate Internet service users. In our research we asked; what is the cost of the procedural steps for forensically investigating a Botnet attack? The research method applies investigation guidelines provided by other researchers and evaluates these guidelines in terms of the cost to a digital forensic investigator. We conclude that investigation of Botnet attacks is both possible and procedurally feasible for a forensic investigator; but that scope management is critical for controlling the cost of investigation. We recommend quantifying Botnet investigations into five levels of cost based on time, complexity and technical requirements.

Keywords: Botnets, Cybercrime, Investigating, Techniques, Costs, Research

1. INTRODUCTION

The economic driver for Botnet propagation is simple. Someone (the master or herder) sets up a network of control over many computers (Bots) and steals the computing and communication system resources. The stolen capabilities are then on-sold to willing buyers who make a living from spamming, theft of personal identities, extortion, DDOS attacks and so on. It is a simple economic formula that delivers the promise of high financial gains to the masters. The propagation method employed by Botnet masters has been moved from a push-based model where the malwares are commissioned to remotely intrude a system through security flaws, to a pull-based model where the unwitting host performs an action such as a download or a mouse click (Provos, Mavrommatis, Rajab, and Monroe, 2008). One of the propagation techniques in this new model is using various social engineering techniques. For example, attackers gather visitors of a website with phishing methods, and allow the visitors to accidentally download the malware. Another technique involves exploitation of various browser vulnerabilities. In this case, visitors come to automatically download malware and run it without their knowledge. Using the techniques, the number of victims can be easily increased without any traditional security barriers because conventional protection mechanisms cannot prevent the victim actions (Chiang, and Lloyd, 2007).

The evolution of Botnet attacks from push to pull has made defense and investigation more difficult. In the investigation of a Botnet using a traditional method such as a push-based model, investigators might locate the attack vector by finding vulnerabilities in the system with penetration testing or by reconstructing the event. However, to find the initial phase of an attack in push-based Botnet methods, investigators must evaluate the various possibilities of how the Botnet malwares were distributed (Schiller, Binkley, Evron, Willems, Bradley, and Harley, 2007). The aim of investigation is to locate the binaries that give the Botnet the capability to expand and create zombies of other systems. It is the forensic analyst's goal to capture and to unpack these binaries so that the type and the source of the Botnet may be found. However around 90% of malware binaries employ analysis-resistance

techniques (Semantic Security Response, 2010). The most prevalent of which are the run-time unpacking of compressed and encrypted code, run-time modifications to existing code, and obfuscations of control transfers in the code. Hence the work is challenging. Most often a static analysis is undertaken where the binary is unpacked to learn its structure. Then the code is re run in a secure test bed to dynamically understand the behaviors that may be expected of the binary and these behaviors are mapped onto other evidences from an event and the affected system (Bailey, Cooke, Jahanian, Xu, and Karir, 2009).

The objective of our research was to establish a way through which digital forensic investigators could systematically investigate Botnet attacks and to reconstruct the event. The procedural steps required simplification from current investigation guidelines so that evolutionary trends and the consideration of cost effective professional practice might be factored into a comprehensive report. We set up test conditions that focused on the effects of Botnets rather than trying to penetrate carefully protected Botnet communications, architectures and defenses. Consequently our interest was the evidence remaining on a victim's system, the malwares and the traces that show the actions. We also strove to have investigation techniques that would be functional in practice and to be relatively simple to follow. One of the key functionalities is risk management that protects the integrity of the evidence and also the security of the investigator information system when for example binaries are executed for analysis. The remainder of this paper is structured to review previous literature, report our findings and to discuss the possibilities for cost efficient digital forensic investigation of Botnets.

2. PREVIOUS LITERATURE

A Botnet is a collection of computers or a large network of compromised computers (Ullah, Khan, and Aboalsamh, 2013). A Bot refers to malicious software that runs on an infected computer and gives control to the attacker (Rajab, Zarfoss, Monroe, and Terzis, 2006). A Bot is also known as a virus of viruses (Schiller, et al., 2007). The attacker controls Bots by using a C&C command channel for the exchange of instructions for actions (Correia, Rocha, Nogueira, and Salvador, 2012). The attacker usually uses one or more servers in order to allow continuous communication and to off load stolen information (Zahid, Belmekki, and Mezrioui, 2012). The command received through the C&C channel is executed autonomously and automatically without the end user's consent. The Botnet is also known as zombies because the malicious intent is hidden until activated by an instruction (Choo, 2007). Also the attacker who controls the C&C server is called the Bot master or the master (Rajab et al., 2006).

A Bot is different than other types of malicious software that harm the computer or a network. A Bot acts as an agent where the Bot software can execute the commands without making any communication with its operator (Provataki, and Katos, 2013; Zahid, Belmekki, and Mezrioui 2012). A Botnet is a collection of Bots that connect to each other through a malicious network imposed by a master for economic gain. The terms "Bot" and "Botnet" can be used in both hardware and software applications according to the context and refer to a system and a group of systems (Grizzard, Sharma, Nunnery, Kang, and Dagon, 2007). The Bot clients can use the functionality of other malicious codes to propagate themselves in order to hide from detection and to attack the target. The primary difference between the Bot clients and viruses or worms is that Bot clients are able to take an action autonomously and execute the given commands in a coordinated manner (Schiller et al., 2007). Bot clients have the ability to perform their actions when attackers are not logged into the target machine. For this reason, a Botnet can be classified by the C&C which are usually IRC Internet, P2P or HTTP (Chiang, and Lloyd, 2007). Bots are usually modular, adaptive, and are programmed to target specific processes to achieve particular functionalities. In this way the one Bot army can have both push and pull capability or operate with either capability independently. When a Bot discovers a new opportunity on a victim system, it can automatically install a specific module to distribute the malware. It means that defeating one component of a Botnet is not enough to ensure that the entire system is cleaned up. Also the Bots utilize a number of techniques to increase continuity and stability

depending on the situation of a specific system targeted (Hoagland, Ramzan, and Satish, 2008). In cases where authorities disrupt a C&C server at a certain IP address, the Bot master can easily set up another C&C server instantly with the same name at a different IP address.

Botnet investigations usually start with the active collecting of samples or the passive detection of Bot behaviors (Mell, Kent, and Nusabaum). Honey-pots have been widely used as an information system resource whose value lies in unauthorized or illicit use of that resource (The Honey-pot Project, 2007). Baecher et al. (2006) argue that the collecting and analyzing of malware samples provides a better defense against the existing threats and also against potential events. In particular, statistical information generated from the large scale samples can be useful to learn about the patterns, trends, and types of attack. The honey-pot technologies have been recognized as good sample providers in several Botnet research studies (Cooke, Jahanian, and McPherson, 2005; Freiling, Holz, and Wicherski, 2005). Detecting Botnets is another approach using passive network traffic monitoring and analysis. These techniques have been useful to identify the existence of Botnets by detection of behaviors associated with groups of compromised machines within a monitored network. Gu et al. (2008) conducted research in which they assumed that Bots within the same Botnet could be characterized by their protocols such as network communication traffic and malicious activities. Based on this assumption, the researchers categorized Bots by using IRC protocol and executed a large number of Bot samples obtained by this categorizing. These efforts enabled them to identify the first level of IRC servers and then infiltrate the corresponding IRC channels to snoop on the Botnets (Feily, Shahrestani, and Ramadas, 2009).

Recent research shows the latest trend in Botnets moving away from plaintext IRC protocols to encrypt HTTP-based or P2P protocols (Baecher, Koetter, Holz, Dornseif, and Freiling, 2006; Ianelli, and Hackworth, 2007). Those new techniques make the malware detection using the approach that is described above difficult. The reasons include the changes in the structure of the Botnet and difficulty of understanding encrypted network protocols. For example, the structure of Botnets is shifting from a centralized one to a distributed one because of its use of P2P architecture (Grizzard, Sharma, Nummery, Kang, and Dagon, 2007; Wang, Sparks, and Zou, 2007). Furthermore, a Botnet can change its C&C server address frequently during its lifetime by using fast-flux service networks (Bacher, Holz, Koetter, and Wicherski, 2008; Holz, Gorecki, Rieck, and Freiling, 2008). Therefore, the Botnet detection system should be independent of the C&C protocol, structure, and infection model of Botnets, requiring further research to address these issues. Stealth and deception techniques have been changed continuously to avoid detection and analysis. The technique for detecting the existence of malware is based on the signatures of a binary file such as byte sequences and strings (Tabish, Shafiq, and Farooq, 2009). The signature based malware detection can be easily defeated by packer and binary code obfuscation techniques (Stepan, 2006).

Previous research has introduced several methods of conducting Botnet investigations. Ard (2007) described two different stages necessary in any Botnet investigation where one is the analysis of the malware itself, which includes examining the binary file. This investigation may also include a run-time analysis to identify network information. The other stage involves tracking sources, which entails identifying the DNS name registers, the IRC servers and the controllers. However, this research did not provide the investigator with adequate procedures to acquire digital evidence while maintaining the integrity of the evidence (Wang, and Kao, 2007). The investigation conducted in the second stage is divided into two different parts: (1) off-line examination of abnormal files, and (2) on-line analysis of sniffing packets. The off-line examination is guided by step-by-step instructions. The essential steps include checking the system time clock, examining running processes and examining the original settings (Daswani, and Stoppelman, 2007). After going through those steps, it was determined which traffic is relevant to the investigation so that the investigator could gain connectivity and learn what the network activity looks like. The second part of packet sniffing gives an effective way to analyze what data is stolen and where it is sent. Similarly memory forensics has received attention in live

digital forensics (Ligh, Adair, Hartstein, and Richard, 2010). A physical memory can contain critical evidence that may not be obtained while the system is not active. Memory forensics can assist the investigation by breaking down the techniques that the malware writers employed to avoid detection and make analysis difficult. For example, the binary code loaded on a physical memory is in an unpacked state (Adelstein, 2006; Hay, Bishop, and Nance, 2009).

3. TEST SET UP

The test set-up was informed by the literature reviewed. The set up was designed into two parts; one part to trap the Botnet malwares in a low interaction honey-pot and then to export for external analysis. Secondly, the captured Botnet malwares were released in a controlled environment to study the behaviors and the behaviors in relation to the controlled environment. The purpose was to forensically identify the attack vectors by studying the malware behaviors and pathways within the victim system. The attempt was to assess the reconstruction of the event and the cost of investigation. This is a victim investigation and the purpose was to write a report that would be useful in hardening the computing system, educating users and for improving the resilience of the systems to further attack. No attempt was made to trace the origin of the attacks beyond identifying the originating IP addresses which may or may not have been spoofed by dynamic domain name system (DDNS) utilization. Similarly communications beyond the victim system were not monitored. The physical target acted as a victim's system where the event was reconstructed. The static analysis system consists of installed analysis tools for memory analysis and the reverse engineering of the binaries (see Figure 1).

The data processing comprised of four key stages. The first stage collected malwares from the Internet to build a localized malware signature database. This stage focused on the information taken from the malware signature and the result of dynamic analysis conducted by external service providers. A sample set from the set of malwares collected then became the input for stage 2. The aim of the second stage was to identify and preserve the source of possible digital evidence by conducting live forensic investigation on the infected host. The precedence for this stage is to select the forensic tools and procedures by reviewing case studies of previous work. In this stage, the research simulates infection from the collected sample set and then is followed by a forensic investigation. The stage particularly focuses on acquisition and preservation of volatile and non-volatile digital evidence.

The third stage involves forensic analysis of the malicious binary and the demonstration of the behaviors. This stage aims to identify and extract the malicious binaries related to abnormal activities and to analyze using static and dynamic methods. The stage involves forensic analysis of previously captured memory images. Dynamic analysis helps to determine the digital evidence that is a direct or indirect result of malicious activities caused by malware execution; and also to compare this evidence with memory images. In addition to the memory analysis, static analysis is supplemented with the information which is produced during the Stage 2 investigation. It serves to forecast potential malicious functionalities that have not yet been performed.

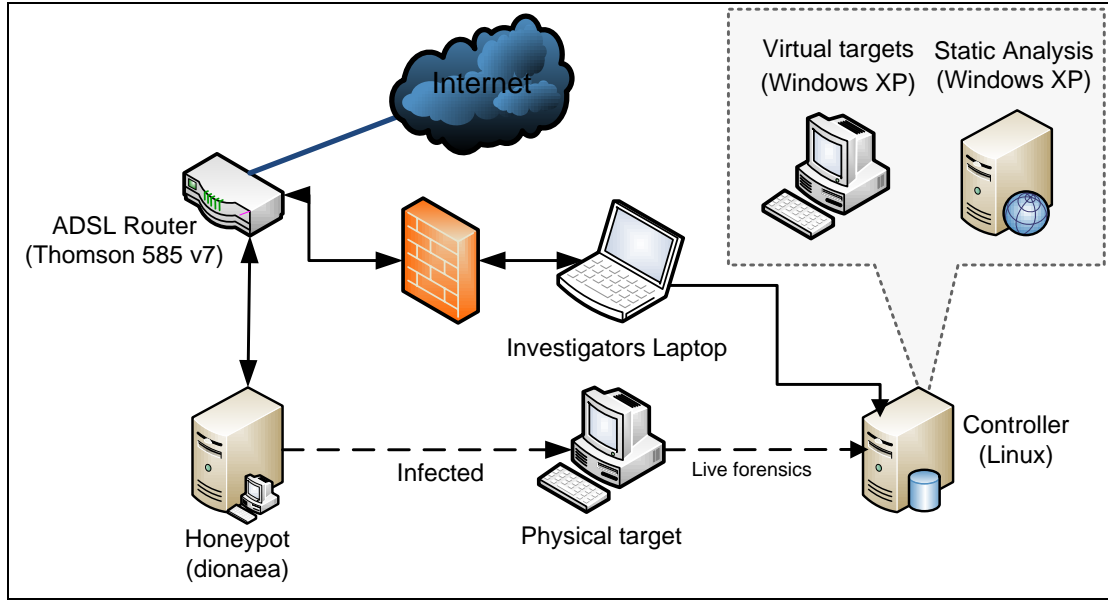


Figure 1 System Architecture for the Botnet Investigation

Finally, the aim of the fourth stage is to evaluate the processes of research and to assess the cost of achieving an effective Botnet investigation. Such information is helpful in setting the scope of an investigation.

4. THE RESULTS

The honey-pot reported after 11 days more than 140,000 exploitation attempts, the repelling of 3,227 attacks, 1,466 malware samples and 110 unique binaries. These events were exported for analysis and the analysis reports showed that 96% of the malicious malware were of Conflicker.B and Conflicker.C Bot. Virtualization software provided the most efficient and flexible method to catch Botnet malwares. When a researcher uses physical computers and completes their own analysis then the complexity and costs increase rapidly. Costs are not just financial and time driven but also include efficiencies and risk management. A hybrid of physical, virtual and outsourcing services optimizes the requirement for cost effectiveness. Table 1 lists a full scope of the software and services we used. The honey pot was hosted virtually on VMware and the analysis services outsourced to Anubis and CWSandbox. After virus scanning the binaries were further analyzed using unpacking, string extraction and reverse engineering techniques. The static evidence was then compiled and used to run a dynamic simulation in a secure machine. The following Tables and Figures report evidence of each procedural step with the exception of the port analysis Table that was too large to include. The intention is that another scientist or investigator may replicate this study and compare results in the interest of growing knowledge in this area of forensic investigation. A commentary is provided for each table or figure to interpret and explain the content of each exhibit.

Table 1 Tools for Data Collection and Analysis

Type	Name	Purpose
Malware collection	Dionaea	A low interaction honey-pot that collects a copy of the malware exploiting vulnerabilities
Virtualization	VMware workstation Virtual Box	Tools for visualizing the computer system.
Forensic Image	Helix Pro	A forensic tool that is specified for incident response.
Memory analysis	Volatility Framework	A forensic tool that can extract various types of information from a memory image.
Initial virus scan	Virus Total	A public service that analyzes suspicious files and URLs
Initial sandbox analysis	Anubis, CWSandbox	Public services that analyze the behavior of Windows PE-executables with special focus on the analysis of malware
Packer Detectors	PEiD v 0.94	A tool that detects packers, cryptors and compilers for Windows PE-executables
String extractor	BinText v3.03	A tool that finds ASCII, Unicode and Resource strings in a file.
Disassemblers and Debuggers	IDA Pro OllyDbg	Tools for reverse engineering.

We found that the purpose and the behavior of the Botnets could be established from the reports. For each process the malicious code is described by file, Registry, and network activities. Figure 2 shows the result of IRCBot analysis.

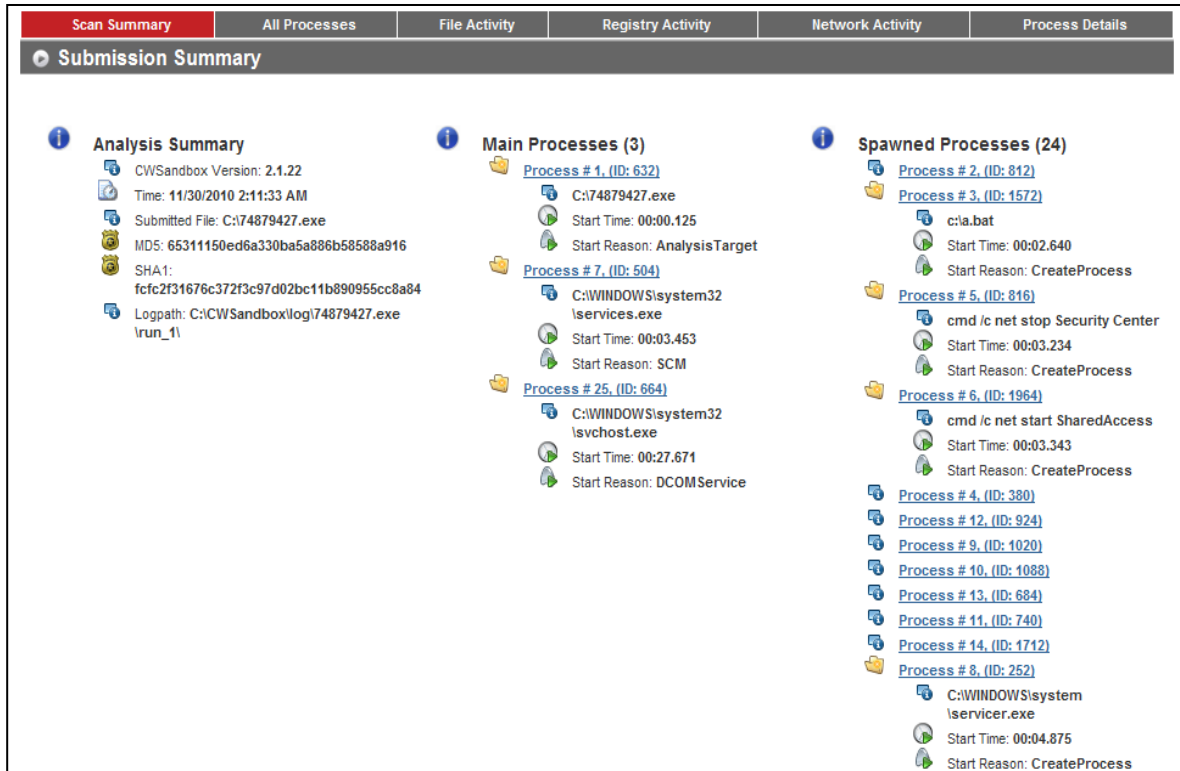


Figure 2 The Analysis Summary of IRC Bot Generated by CWSandBox

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

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

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

In the file activities section, the result shows evidence of the malicious code in the infected system. The a.bat file has been created by the Process #1 (ID: 632). At the same time, this process copies itself to the Windows System folder (C:\WINDOWS\system) as named 'servicer.exe'. Next, the created batch file creates a Registry file name l.reg at the administrator's temporary folder (C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\). This Registry file is loaded by the same process. After executing batch files and updating the Registry, the batch and Registry files are deleted by themselves to hide their activities. In addition to deletion of created files, the first infected file also has been deleted by the process which has launched the copied file. Table 2 shows the summary of file activities of IRCBot on the infected machine.

Table 2 Summary of File Activities of Ircbot on Infected Machine

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

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

Table 3 Registry Values Changed by IRCBot

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

Table 3 shows the Registry values that are changed by the IRCBot process. Those values are used to prevent Windows Security Center and Update Services from starting automatically. In addition, an attacker changed the TCP/IP service parameter as shown in the report. The main strength of CWSandBox is to provide information of the network activities. In the network section, the result shows the network communication through the IRC channel. The Process #8 (ID: 252) communicated with 60.10.179.100:8681 (the IP address of a remote host). The process used “SP2-501” as user name and “USA|XP|SP3|446911” as a nickname. The report of network activities is shown in Figure 3. According to the keywords on the communication message, the researcher can infer that this binary

has the capability for a DDOS attack. The Botnet that this Bot belongs to has at least two C&C servers: 58.240.104.57 is for update and 60.10.179.100 is for distribution.

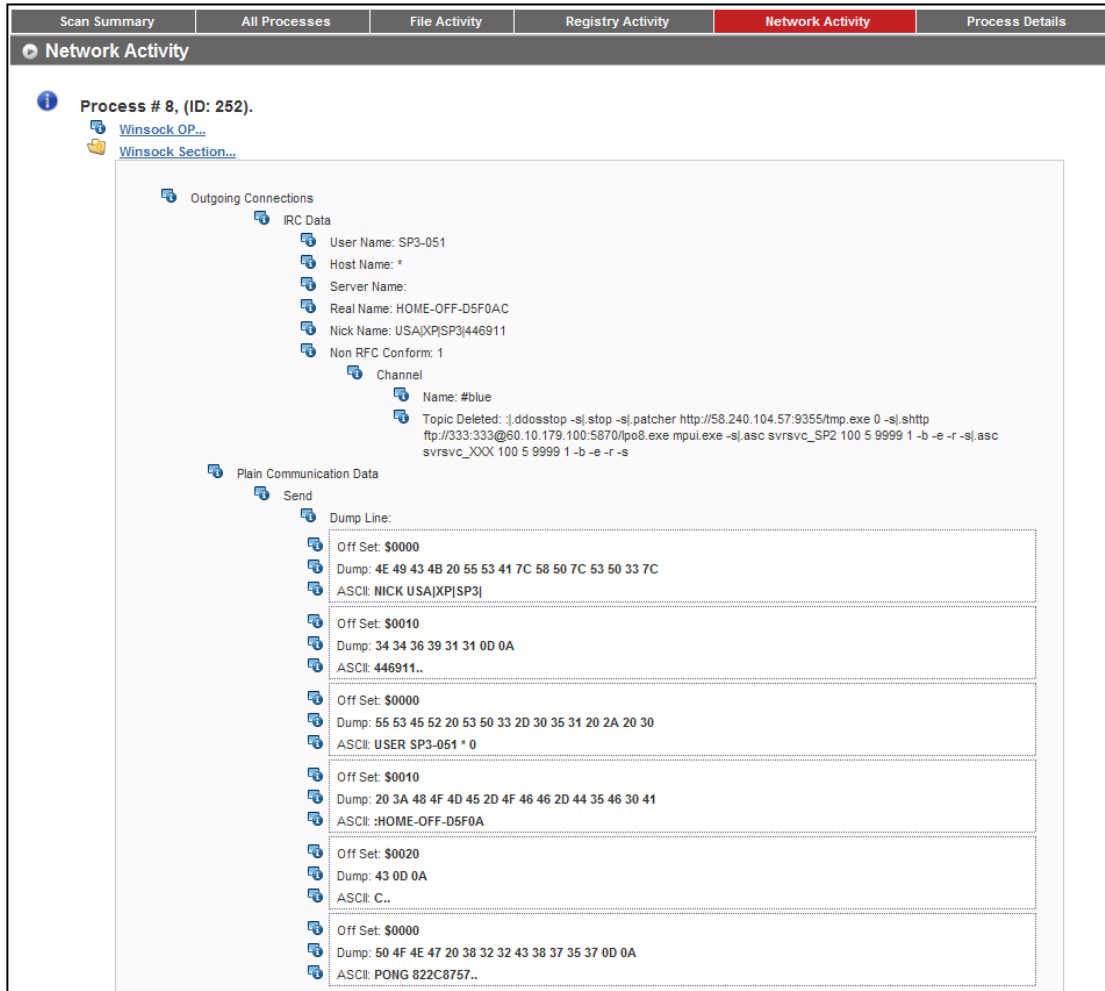


Figure 3 The Analysis of Network Activities on an Ircbot Infected Machine

There is a lot of similarity of the analysis reports generated by CWSandBox and Anubis. On the first page of the Anubis report, the risk level of analyzed malware is shown in different fields such as file modification and destruction, Registry activities, auto-start capabilities and so on. In this case, Anubis service gives a high level warning on permanent file modification and destruction. The Anubis report of the IRCBot shows two main processes: cmd.exe and services.exe. The process named cmd.exe performs several command line instructions as also shown on the CWSandBox result. While the structure and shape is different, the behavior of each process is similar.

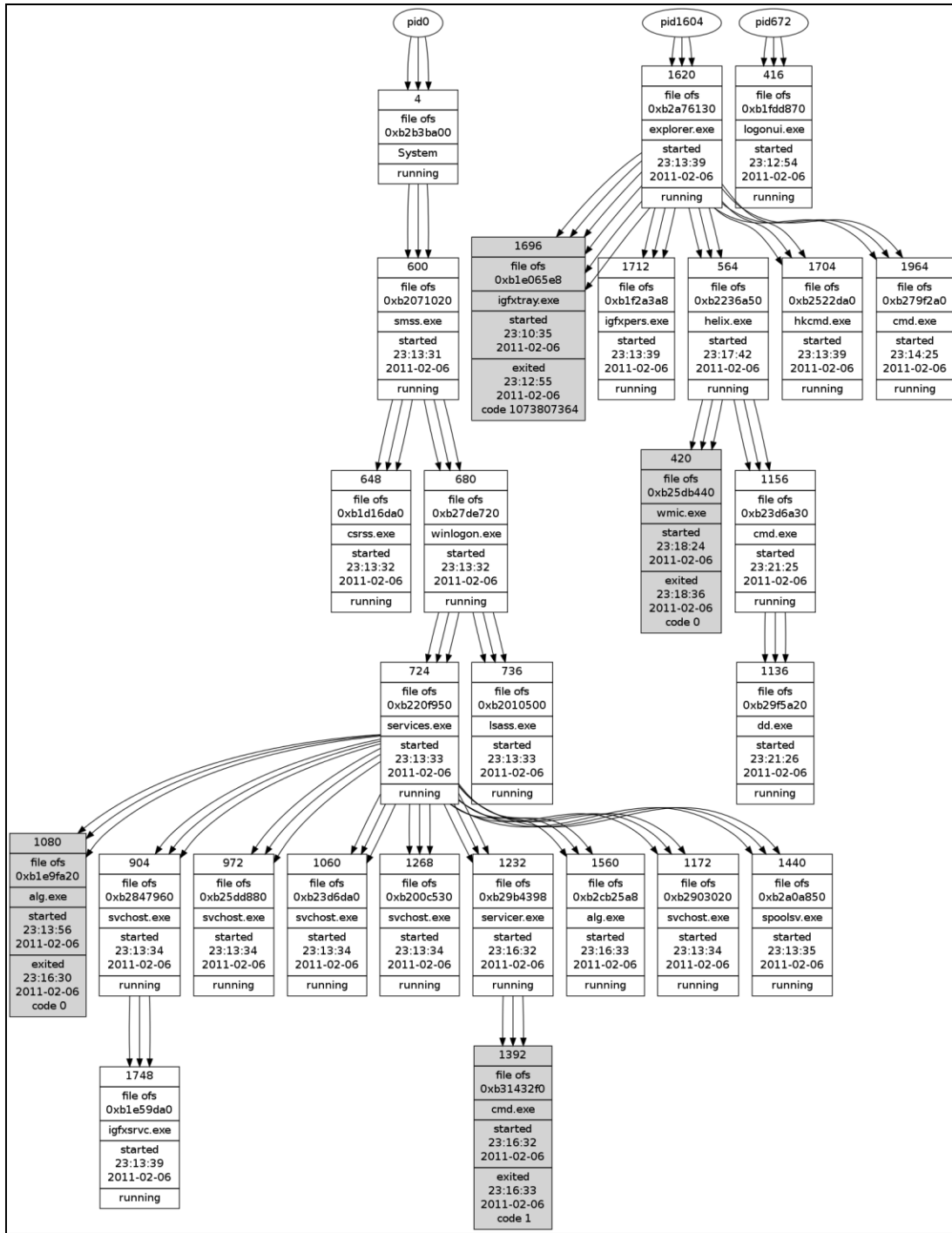


Figure 4 The Running Process Lists on a Victim's Physical Memory

The first step of investigation was to establish the existence of malware binaries and identify their location. In general, the malware is running as a single process or a part of legitimate process. To extract the process list from the forensic image of physical memory, the researcher used the Volatility Framework which is an open source memory forensic tool. Figure 4 shows the diagram of running processes on the physical memory acquired from the IRCBot infected victim machine. This information can be obtained by finding the _EPROCESS structures in a memory dump. The result of the Volatility Framework shows the relationship between parent and child processes. In the process

graph, Pid 0, the System Idle Process, does not have details because it is not a real process. The details of Pid 1636 are not available because the parent process of this has been finished and terminated at that moment. Based on the tree structure, it shows that a user logged onto the machine and ran helix.exe from explorer.exe. Using the cmd.exe shell on the helix CD, the user invoked dd.exe to dump the machine's memory. In the current state, the investigator cannot identify any malicious processes but the investigation step is important as evidence can be found in a process analysis. Also the connections were determined between the infected system and a remote location. Consequently a port analysis was conducted to identify open ports on the infected machine.

After the infection, a live forensic investigation to acquire an image of the hard disk and physical memory was performed. The physical memory showed hidden abnormal mapped files; injected DLL and memory segments (see Figure 5). The result of signature based analysis showed related processes and contained a memory offset, output file path and a dumped binary. Signature analysis on a collected memory dump can help to reduce the number of suspicious processor and related files. While the total number of process listed is 28, after signature analysis the suspicious files were reduced to 13. In addition to the binary information, the result provides an assembly code of related memory offsets. The process named servicer.exe is the most suspicious. As shown in Figure 5 the malware calls VirtualAllocEx function to perform a typical code injection.

Process	Pid	Start	End	Tag	Hits	Protection
servicer.exe	1308	0x320000	0x321fff	VadS	0	6MM_EXECUTE_READWRITE
0x00320000	08 00 00 00 00 00 00 00	56 57 53 55 8b 5c 24 1cvws...	\$.		
0x00320010	85 db 0f 84 ab 00 00 00	e8 0d 00 00 00 61 65 72ker			
0x00320020	6e 65 6c 33 32 2e 64 6c	6c 00 ff 13 85 c0 0f 84	nel32.dll.....			
0x00320030	8f 00 00 00 8b f0 e8 0c	00 00 56 69 72 74 75virtu			
0x00320040	61 6c 46 72 65 65 00 56	ff 53 04 85 c0 74 74 8b	alFree.v.s...tt.			
0x00320050	e8 e8 0d 00 00 00 56 69	72 74 75 61 6c 41 6c 6cVirtualAll			
0x00320060	6f 63 00 56 ff 53 04 85	c0 74 58 8b 74 24 14 8b	oc.v.s...tx..t\$..			
0x00320070	7c 24 18 6a 04 68 00 10	00 00 ff 36 6a 00 ff d0	\$.j.h....6j...			
00320000:	0800	OR [EAX], AL				
00320002:	0000	ADD [EAX], AL				
00320004:	0000	ADD [EAX], AL				
00320006:	0000	ADD [EAX], AL				
00320008:	56	PUSH ESI				
00320009:	57	PUSH EDI				
0032000a:	53	PUSH EBX				
0032000b:	55	PUSH EBP				
0032000c:	8b5c241c	MOV EBX, [ESP+0x1c]				
00320010:	85db	TEST EBX, EBX				

Figure 5 Suspicious Memory Ranges and Injected Code

The purpose of investigating Registry is to determine which Registry keys are accessed by suspicious processes and to figure out the values and data of those keys. In general, the attacker changes existing Registry values or creates new keys for various reasons. For instance, some malware store their command and control server information. In addition to the configuration purpose, Registry keys-related security policy is changed for accessing confidential information and bypassing the local firewall. The same Registry activities are found in the memory image. The analysis of file activities identifies changed files and examines the executable's Import Table. Identifying changed files is a key aspect of malware analysis. An effective way to detect the changes the malware causes to a victim system is by determining the changes that happen in normal situations. In this research, the memory image only contains the state of a certain moment when an investigator is conducting the acquiring procedure. For this reason, the researcher collected the list of files that were currently opened by the running processes. The files opened by IRCBot used three Index.dat files at different locations. Index.dat files are binary files that Internet Explorer uses to store the URLs of a user. They are

designed for IE's internal usage and are usually located under the user's document folder. However, according to the information extracted from the memory image, *serviser.exe* process used those files for malicious purposes. Furthermore, they are not stored in the current user's document folder.

Table 6 The File List that is Opened by the IRCBot.

```
C:\WINDOWS\system32
C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-
Controls_6595b64144ccf1df_6.0.2600.2180_x-ww_a84f1ff9
C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-
Controls_6595b64144ccf1df_6.0.2600.2180_x-ww_a84f1ff9
C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-
Controls_6595b64144ccf1df_6.0.2600.2180_x-ww_a84f1ff9
C:\Documents and Settings\LocalService\Local
Settings\Temporary Internet Files \Content.IE5\index.dat
C:\Documents and Settings\LocalService\Cookies\index.dat
C:\Documents and Settings\LocalService\Local
Settings\History\History.IE5\index.dat
C:\net\NtControlPipe10
C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-
Controls_6595b64144ccf1df_6.0.2600.2180_x-ww_a84f1ff9
```

In order to analyze the imported function tables, the researcher started with gathering the information currently loaded by external libraries. The author of the malicious executables is using external libraries to increase its functionality with static and dynamic linking. The static approach can make malicious software run in a standalone mode by embedding external libraries. However dynamic linking is more popular because it can decrease the size of executable binaries. Also this method improves its portability across the various versions of operating systems. Therefore determination of associate DLLs and imported functions can be useful to explain the behavior of malicious binaries. The loaded DLL of *servicer.exe* process is shown in Table 7.

Hence the Investigator is now in a position to generate an overall picture of the Botnet attack by putting all the evidence sources together. The propagation mechanism of the sample Botnet can be found in the log file of a malware collection system. At first, the infected machine (IP Address: 118.92.101.75) was exploited by the remote host (IP Address: 118.91.176.154). The remote host connected the victim host through port 445 and exploited the vulnerability of Microsoft Server Message Block (SMB) Protocol. In this case, the attack machine used a type of remote shell code to download a malicious Bot binary. Table 8 shows the instruction for the shell code downloaded from the remote host. The shell code downloaded a file named *lpo8.exe* from an ftp server (<ftp://123:123@60.10.179.100:3069/lpo8.exe>).

Table 7 The List of Loaded External Libraries

Base	Size	Path
servicer.exe pid: 1232		
Command line : "C:\WINDOWS\system\servicer.exe"		
Service Pack 2		
0x400000	0x78000	C:\WINDOWS\system\servicer.exe
0x7c900000	0xb0000	C:\WINDOWS\system32\ntdll.dll
0x7c800000	0xf4000	C:\WINDOWS\system32\kernel32.dll
0x77d40000	0x90000	C:\WINDOWS\system32\user32.dll
0x77f10000	0x46000	C:\WINDOWS\system32\GDI32.dll
0x77dd0000	0x9b000	C:\WINDOWS\system32\ADVAPI32.dll
0x77e70000	0x91000	C:\WINDOWS\system32\RPCRT4.dll
0x71b20000	0x12000	C:\WINDOWS\system32\MPR.dll
0x7c9c0000	0x814000	C:\WINDOWS\system32\SHELL32.dll
0x77c10000	0x58000	C:\WINDOWS\system32\msvcrt.dll
0x77f60000	0x76000	C:\WINDOWS\system32\SHLWAPI.dll
0x773d0000	0x102000	C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2600.2180_x-ww_a84f1ff9\comctl32.dll
0x5d090000	0x97000	C:\WINDOWS\system32\comctl32.dll
0x71ab0000	0x17000	C:\WINDOWS\system32\WS2_32.dll
0x71aa0000	0x8000	C:\WINDOWS\system32\WS2HELP.dll
0x76d60000	0x19000	C:\WINDOWS\system32\iphlpapi.dll
0x771b0000	0xa6000	C:\WINDOWS\system32\WININET.dll
0x77a80000	0x94000	C:\WINDOWS\system32\CRYPT32.dll
0x77b20000	0x12000	C:\WINDOWS\system32\MSASN1.dll
0x77120000	0x8c000	C:\WINDOWS\system32\OLEAUT32.dll
0x774e0000	0x13c000	C:\WINDOWS\system32\ole32.dll
0x5b860000	0x54000	C:\WINDOWS\system32\NETAPI32.dll
0x77260000	0x9c000	C:\WINDOWS\system32\urlmon.dll
0x77c00000	0x8000	C:\WINDOWS\system32\VERSION.dll
0x73dd0000	0xfe000	C:\WINDOWS\system32\MFC42.DLL
0x77fe0000	0x11000	C:\WINDOWS\system32\Secur32.dll
0x71ad0000	0x9000	C:\WINDOWS\system32\wsock32.dll
0x74290000	0x4000	C:\WINDOWS\system32\icmp.dll
0x76f20000	0x27000	C:\WINDOWS\system32\dnsapi.dll
0x74320000	0x3d000	C:\WINDOWS\system32\odbc32.dll
0x763b0000	0x49000	C:\WINDOWS\system32\comdlg32.dll
0x20000000	0x17000	C:\WINDOWS\system32\odbcint.dll
0x76bf0000	0xb000	C:\WINDOWS\system32\psapi.dll
0x77b40000	0x22000	C:\WINDOWS\system32\Apphelp.dll
0x71a50000	0x3f000	C:\WINDOWS\System32\mswsock.dll
0x76fb0000	0x8000	C:\WINDOWS\System32\winrnr.dll
0x76f60000	0x2c000	C:\WINDOWS\system32\WLDAP32.dll
0x76fc0000	0x6000	C:\WINDOWS\system32\rasadhlp.dll
0x76ee0000	0x3c000	C:\WINDOWS\system32\RASAPI32.DLL
0x76e90000	0x12000	C:\WINDOWS\system32\rasman.dll
0x76eb0000	0x2f000	C:\WINDOWS\system32\TAPI32.dll
0x76e80000	0xe000	C:\WINDOWS\system32\rtutils.dll
0x76b40000	0x2d000	C:\WINDOWS\system32\WINMM.dll
0x722b0000	0x5000	C:\WINDOWS\system32\sensapi.dll

Table 8 The Shell Code Decode by Dionaea

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

The activities caused by malicious binaries are explained according to the type of activities and explained in order of time. After downloading a binary, it self-executed. At first it stopped in the Windows Firewall and Security Centre Service to hide its existence. This process created a batch file name a.bat and the batch file was executed. Also it copied itself to the Windows system folder (C:\WINDOWS\system\) and changed its name as servicer.exe. The created batch file created a Registry file named 1.reg to change the Registry values of Windows Firewall, Security Centre Service and Automatic Update Service. Moreover, this process installed a copied file as a Windows service to start when the system is booted. Finally, the process started servicer.exe and alg.exe Windows service process. Servicer.exe process executed similar instructions to the downloaded binary because the two binaries have the same MD5 signature. However, the latter process worked in a slightly different way. According to the analysis report from CWSandbox, this process connected to the IRC server (IP Address: 60.10.179.100: 8681) and joined the IRC channel. Also the malicious process patched itself from another server (IP Address: 58.240.104.57:9355).

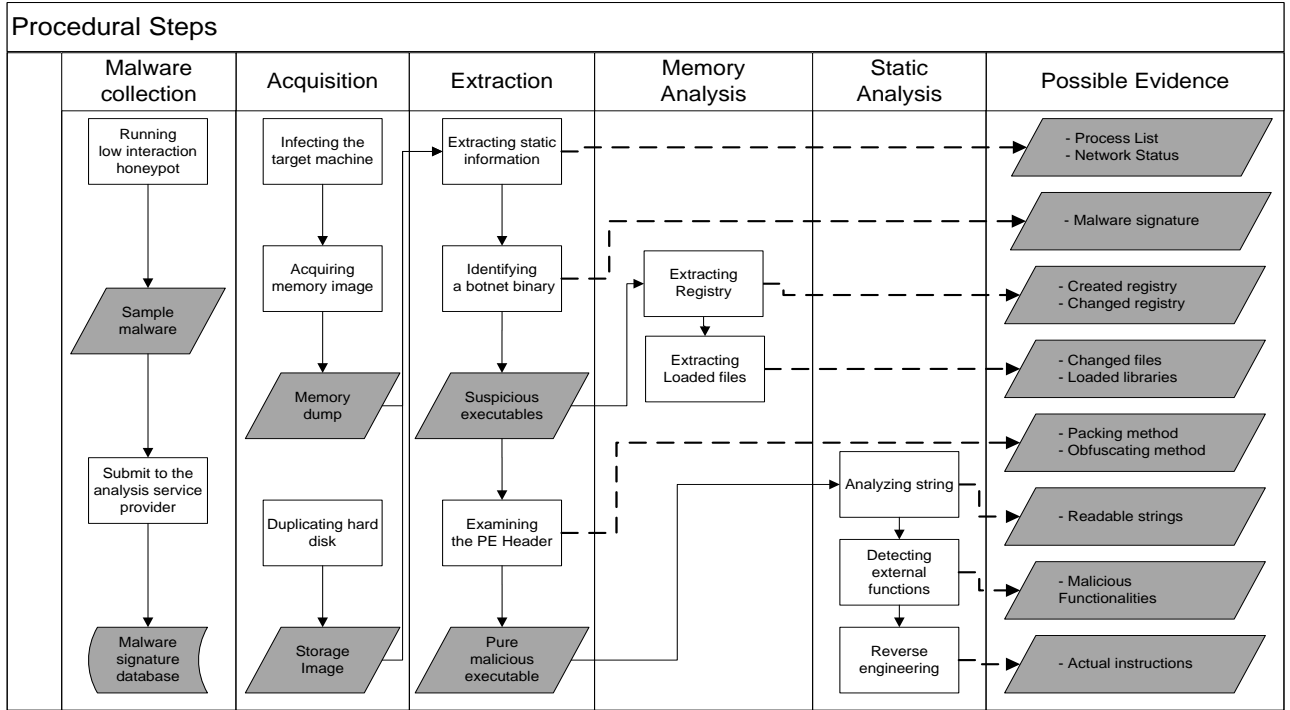


Figure 6 Summary of Investigation Procedural Steps

5. IMPLICATIONS FOR PRACTICE

The literature reviewed gave guidelines for investigating a Botnet attack and defined the complexities involved. We set further limitations by designing an experiment that was manageable and was targeted to explore a scenario a digital forensic investigator may encounter in a call out. Some of the bigger and more complex investigation possibilities were left out of the scope of a victim investigation. These matters included identifying zombie behaviors in a live network and attempting to defeat Botnet defensive mechanisms. Hence the research systematically investigated malware signatures and behaviors; and investigated a victim system in order to reconstruct the event. The findings have shown that each phase of investigation is possible and that independently the results report potential evidences. However when the independent evidences are triangulated an overall picture of the event can be established. In push attacks the attack vector can be readily located by such procedural steps as we have demonstrated. However pull attacks require a greater scope for evidence where a lot more is required to be known about the user of the system and their use behaviors. The requirement relates to social engineering and the related actions that may not be fully represented in the technical system analysis. Our research borders on explaining the human user use behavior but falls short when the user decisions and actions are required to clarify the vector origins. We can locate the process but not the user reasoning for opening the process or if the process was intentional, automated or unintended subversion. The matters show the scope of our investigation but also define requirements for a general Botnet investigation that can be discussed in terms of what we have demonstrated. For example the costs and risks of out of scope activities can be estimated based on the procedural steps of the research.

Digital forensic investigation is costed out against the expected return of evidence. However, the expectation may not be realized in many instances because the evidence is not there, the evidence is damaged, the technical challenge is too great and techniques are inadequate for effective evidence recovery. Regardless the expenditure of resources and the employment of technical skills create cost. Consequently in many instances and in particular in civil cases the cost of investigation sets the limits to which an investigation may go. In our experiment we were conscious of the cost of time and the benefits of risk mitigation. Consequently we outsourced phase 1 to service suppliers for signature

analysis and reports; and used virtual environments in isolation to observe the behaviors of captured malwares. These tactics reduced time, minimize technical requirements and managed the risk of damages within acceptable tolerance. To generalize these actions to practitioner requirements is not difficult. We recommend the setting of levels of investigation expectation based on the estimated cost of conducting the Botnet investigation procedures. At the first Level an investigator can expect to simply do signature analysis using an outsourcing agency with the benefit of hardening the system from further attack, assuring the anti-virus software is adequate and updating alert triggers. The second level of investigation was more complex and involved the static testing of malware binaries in a controlled virtual environment. The knowledge gained demonstrated the effects of the malware on a victim computing environment and assisted the reconstruction of an event. Once an investigator has set up a test bed and practiced several investigations the technical cost of testing binaries drops considerably but the time cost remains high. Hence at level 2 the initial technical cost is high and the time cost similar. At Level 3 consideration of detecting Botnets in a network from their behaviors ran beyond the scope of our investigation. Such activity can be accessed and costed as an outsourcing opportunity from Government and network agency providers. There are many agencies that provide network level reports of Internet traffic and the signature analysis for zombies. In parallel with Level 3 outsourcing, a Level 4 investigation of the interception of Botnet C&C communications can be attempted in an attempt to locate the IP addresses for the controller and also any addresses where stolen properties are deposited. At the fifth and highest level of expense the Botnet defenses can be breached to destroy the Botnet. This is a highly problematic activity with considerable technical and time costs.

A digital forensic investigator must decide the scope of a Botnet investigation based on the trade-off of costs and benefits, and in negotiation with the client. Botnet investigations have high complexity until the technical requirements are automated or outsourced but still absorb considerable time resources. Cost efficiencies can be maintained by limiting an investigation to a victim investigation. The procedural steps we have demonstrated can be outsourced, virtualized and automated whereby once a laboratory has setup, benchmarked and tested each procedure and tool set, any binary can be released for observation. The resulting reports are adequate for reconstructing a push event and explaining the technical processes leading to the event. Similarly a laboratory can set up a human computer interaction (HCI) test bed where the victim can show how they used a computer, what actions they take and explain how decisions are made. In this way the pull aspects of a Botnet attack can be mapped onto the push and technical aspects and a full picture of an event formed.

6. CONCLUSION

Botnets remain a challenge for the legitimate users of the Internet and the freedom from economic harm. We have recommended ways in which the problem can be quantified and costed against what may be expended to protect users against Botnet attacks. Victim investigation procedures provide the best cost efficiencies in investigation and open the victim and the system for better protection. The system can be harden and tuned for the best defenses and the victim themselves educated towards better ways to resist social engineering attacks and online trickery. The recommended levels of expenditure allow the problem to be manageable for each budget and to set expectations that may be realized by both investigator and the client. The scope of investigation and the quality of investigation need not be dwarfed by the size of the problem but rather scaled to fit affordable and effective means.

Level	Description	Cost	Benefit	Event Reconstruction
1	Signature analysis by outsourcing agent	Technical = Low Time = Low Complexity = Low	System hardening Anti-virus update Alerts update	Outsource Reports indicate attack signatures and vectors
2a	Level 1 + static test environment build + binary execution and observation	Technical = High Time = High Complexity = High	Level 1 + file, registry and network effects demonstrated	Port, process, and memory analysis plus external contacts (eg., libraries) and IP addresses. Attack Vector reconstruction.
2b	Level 1 + binary execution and observation	Technical = Low Time = Medium Complexity = Low	Many of the processes in Level 2a can be automated	Port, process, and memory analysis plus external contacts (eg., libraries) and IP addresses. Attack Vector reconstruction.
2c	Level 2a + 2b + Full analysis of human with computer interface and human explanation of actions	Technical = Medium Time = High Complexity = Medium	Technical process analysis can be mapped onto human interaction.	The event with the human social and behavioral evidences may be gained to understand the pull attack vector.
3	Network level observation is made that looks for zombie behaviors. Outsourcing recommended.	Technical = Medium Time = Medium Complexity = Low	Pre-emptive actions may be taken and alerts issued.	The Botnet strategy may be observed and countered.
4	Interception of C&C communications	Technical = High Time = High Complexity = High	The wider Botnet scope can be observed.	Counter intelligence activities can be initiated to disrupt the Botnet.
5	Breaching of Botnet defenses and destruction	Technical = High Time = High Complexity = High	Control can be returned to legitimate Internet users.	Full event deconstruction and secure defenses implemented.

Figure 7 Investigation Costs and Benefits

REFERENCES

- Adelstein, F. (2006). Live forensics: Diagnosing your system without killing it first. *Communications of the ACM*, 49(2), 63-66.
- Aquilina, J. M., Casey, E., & Malin, C. H. (2008). *Malware Forensics: Investigating and Analyzing Malicious Code*. Burlington, MA: Syngress.
- Ard, C. (2007). Botnet analysis. *The International Journal of Forensic Computer Science*, 2(1), 65-74.
- Baar, R., Alink, W., & Ballegooij, A. (2008). Forensic memory analysis: Files mapped in memory. *Digital Investigation*, 5(Supplement 1), S52-S57.
- Bächer, P., Holz, T., Kötter, M., & Wicherski, G. (2008). Know your enemy: Tracking botnets. Retrieved October 01, 2013 from <http://www.honeynet.org/papers/bots/>

- Baecher, P., Koetter, M., Holz, T., Dornseif, M., & Freiling, F. (2006). The Nepenthes platform: An efficient approach to collect malware. Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection (RAID 2006), Hamburg, Germany. doi:10.1007/11856214_9
- Bailey, M., Cooke, E., Jahanian, F., Xu, Y., & Karir, M. (2009). A survey of Botnet technology and defenses. Proceedings of the 2009 Cybersecurity Applications & Technology Conference for Homeland Security. Doi:10.1109/CATCH.2009.40
- Balas, E., & Viecco, C. (2005). Towards a third generation data capture architecture for honeynets. Retrieved 11 October 2013 from <http://ro.ecu.edu.au/cgi/viewcontent.cgi?>
- Barford, P., Yegneswaran, V. (2007). An inside look at Botnets. *Advances in Information Security*, 27, 171-191.
- Chiang, K., & Lloyd, L. (2007). A case study of the Rustock rootkit and spam bot. Proceedings of the First Workshop on Hot Topics in Understanding Botnets, Cambridge, MA. Retrieved from http://www.usenix.org/event/hotbots07/tech/full_papers/chiang/chiang.pdf
- Choo, K. (2007). Zombies and Botnets. Canberra: Australian Institute of Criminology. Retrieved from <http://www.aic.gov.au/en/publications/current%20series/tandi/321-340/tandi333.aspx>.
- Cooke, E., Jahanian, F., & McPherson, D. (2005). The Zombie roundup: understanding, detecting, and disrupting botnets. Proceedings of the Steps to Reducing Unwanted Traffic on the Internet (SRUTI '05), Cambridge, MA.
- Correia, P., Rocha, E., Nogueira, A., & Salvador, P. (2012). Statistical characterization of the Botnets C&C traffic. *Procedia Technology*, 1, 158-166.
- Daswani, N., & Stoppelman, M. (2007). The anatomy of Clickbot.A. Proceedings of the First Workshop on Hot Topics in Understanding Botnets, Cambridge, MA.
- Feily, M., Shahrestani, A., & Ramadass, S. (2009). A survey of Botnet and Botnet detection. Proceedings of the Emerging Security Information, Systems and Technologies Conference, 2009. SECURWARE '09.
- Freiling, F. C., Holz, T., & Wicherski, G. (2005). Botnet tracking: Exploring a root-cause methodology to prevent distributed denial-of-service attacks. *Computer Security-ESORICS 2005* 319-335. Retrieved from http://dx.doi.org/10.1007/11555827_19
- Grizzard, J. B., Sharma, V., Nunnery, C., Kang, B., & Dagon, D. (2007). Peer-to-peer Botnets: overview and case study. Proceedings of the First Workshop on Hot Topics in Understanding Botnets, Cambridge, MA.
- Gu, G., Perdisci, R., Zhang, J., & Lee, W. (2008). Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. Proceedings of the 17th USENIX Security Symposium, San Jose, CA.
- Hay, B., Bishop, M., & Nance, K. (2009). Live analysis: progress and challenges. *IEEE Transactions on Security & Privacy*, 7(2), 30-37.
- Hoagland, J., Ramzan, Z., & Satish, S. (2008). Bot networks. In M. Jakobsson & Z. Ramzan (Eds.), *Crimeware: Understanding New Attacks and Defenses*, 183-227. Addison-Wesley Professional.
- Holz, T., Gorecki, C., Rieck, K., Freiling, F. C. (2008). Measuring and detecting fast-flux service networks. Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS' 08), San Diego, CA.
- Ianelli, N. & Hackworth, A. (2007). Botnets as a vehicle for online crime. *The International Journal of Forensic Computer Science*, 2(1), 19-39.

- Ligh, M. H., Adair, S., Hartstein, B., & Richard, M. (2010). *Malware Analyst's Cookbook and DVD: Tools and Techniques for Fighting Malicious Code*. New York, NY: Wiley.
- Mell, P., Kent, K., & Nusabaum, J. NIST. Guide to malware incident prevention and handling. Special Publication 800-83. National Institute of Standards and Technology, Washington DC, USA.
- Provataki, A., & Katos, V. (2013). Differential malware forensics. *Digital Investigation*, 10, 311-322.
- Provos, N., Mavrommatis, P., Rajab, M. A., & Monrose, F. (2008). *All your iFRAMES point to Us*, San Jose, CA: Wiley.
- Rajab, M. A., Zarfoss, J., Monrose, F., & Terzis, A. (2006). A multifaceted approach to understanding the botnet phenomenon. Proceedings of the 6th ACM SIGCOMM conference on Internet measurement, Rio de Janeiro, Brazil.
- Rushi, J., Mokhtari, E., Ghorbani, A. (2011). Estimating botnet virulence within mathematical models of botnet propagation dynamics. *Computers & Security*, 30(8), 791-802.
- Schiller, C., Binkley, J., Evron, G., Willems, C., Bradley, T., & Harley, D. (2007). *Botnets: The Killer Web App*. Burlington, MA: Syngress.
- Stepan, A. (2006). Improving proactive detection of packed malware. Retrieved 28 September, 2012, from <http://www.virusbtn.com/virusbulletin/archive/2006/03/vb200603-packed>
- Symantec Security Response. (2010). Symantec global internet security threat report: Trends for 2009 (Technical Report): Symantec Corporation. Retrieved from http://eval.symantec.com/mktginfo/enterprise/white_papers/bwhitepaper_internet_security_threat_report_xv_04-2010.en-us.pdf
- Tabish, S., Shafiq, M., & Farooq, M. (2009). Malware detection using statistical analysis of byte-level file content. Retrieved October 2013 from <http://ro.ecu.edu.au/cgi/viewcontent.cgi>
- The Honeynet Project. (2007). Know your enemy: Fast-flux service networks. Retrieved 15 September, 2012, from <http://www.honeynet.org/papers/ff>
- Ullah, I., Khan, N., & Aboalsamh, H. (2013). Survey on BOTNET: Its architecture, detection, prevention and mitigation. *IEEE Transactions on Forensics and Security*, 660-665.
- Wang, P., Sparks, S., & Zou, C. (2007). An advanced hybrid peer-to-peer Botnet. Proceedings of the First Workshop on Hot Topics in Understanding Botnets, Cambridge, MA.
- Wang, S. & Kao, D. (2007). Internet forensics on the basis of evidence gathering with Peep attacks. *Computer Standards & Interfaces*, 29(4), 423-429.
- Zahid, M., Belmekki, A., & Mezrioui, A. (2012). A new architecture for detecting DDoS/Brute force attack and destroying the botnet behind. *IEEE Transactions in Forensics and Security*, 1-5.

