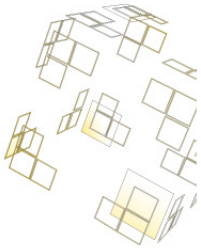
University of Twente

Master Thesis

Detection of APT Malware through External and Internal Network Traffic Correlation



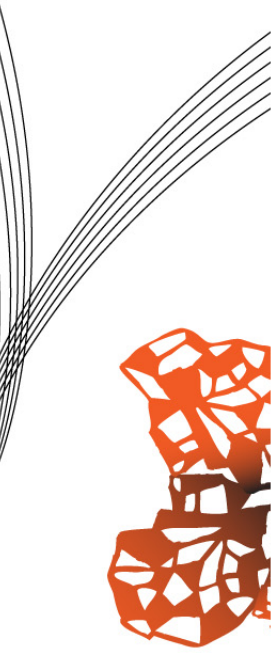
Author:
Terence Slot

Supervisor:
Dr. Frank Kargl

A thesis submitted in fulfilment of the requirements for the degree of Master Computer Science

in the

November 2014



UNIVERSITEIT TWENTE.

Declaration of Authorship

I, Terence Slot, declare that this thesis titled, 'Detection of APT Malware through External and Internal Network Traffic Correlation' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at the University of Twente and Madison Gurkha in Eindhoven.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Signed: _____

Date: _____

Acknowledgements

I would like to thank my professor Frank Kargl for guiding me during this research project.

Furthermore I am honored to have gain the internship experience at Madison Gurkha. The employees at Madison Gurkha were very skillful and have helped me a lot during this thesis. Therefor I would like to thank my supervisor Hans van de Looy at Madison Gurkha for keeping me on the right track whenever I had doubts about which direction to go. The same applies for my professor Frank Kargl.

Many special thanks go to Anna Sperotto for bringing the idea of creating a prototype malware.

I thank Loren Weith for giving me expert insight on designing a virtual network architecture.

Finally I would like to thank Mila Parkour for providing the additional APT malware samples.

Abstract

This master thesis presents overview on advanced persistent threat (APT) definition and explanation of it. One of the most dangerous APT named: "Snake" will be presented along with other similar APT's. Various virtual environments like e.g. VirtualBox will be investigated in order to understand how APT malware behaves in these environments. The central focus of this master thesis lies on detection of futuristic APT malware based on cross-referencing communication patterns in order to detect APT malware. A prototype detection tool will be created and tested in order to detect similar APT's like Snake. Additionally a prototype malware will be supplied as well, which contain similar stealth communication techniques as the Snake APT malware. This prototype malware will be tested with the current state of commercial firewall applications in order to prove its effectiveness. In the end challenges and solutions will be presented for future research work.

Introduction

We live in a world where computer technology is ever growing at an exponential rate every two years, according to Moore's law. Besides from computer technology ever growing, security is known to fall behind. The security experts at Mandiant[1] call this the "Security Gap", where criminals are always one step ahead of the security experts. The rise of term "Advanced Persistent Threat" (APT) is getting more and more important every day with the increase in complexity of computer technology. Where countries in the world would create sophisticated malware in order to spy on each other. Some of these cases are already known through surveillance programs like "Prism". This information was provided to general public through famous whistleblower Edward Snowden. In order to minimize the security gap, new innovative ideas need to be created.

The goal of this master thesis is to create prototype defend mechanism against the most recent APT malware available that has been discovered lately. The idea is to create a client-server model to cross-reference data transmissions in order to detect anomalies, through comparing regular data transmissions. When this prototype is established and sound, other researcher and security experts may benefit from the prototype through implementing its architectural design in their existing security products.

This master thesis starts with the investigation of how complex and dangerous APT are. The details can be found in Chapter 1, where APT will be explained and categorized in the end of the chapter. APT is categorized into attack and evasive methods performed by the perpetrators, whereas the opposite party is respectively defined in defend and tracing methods. In chapter 2 APT malware samples will be collected to determined what their behavior is. Furthermore the chapter provides a list of virtual environments like e.g. VMware or VirtualBox in order to assess what possible evasive counter measures exist for APT malware to detect these environments. Two control groups will be created in the form of modified virtual environments known as sandboxes in order to detect if any of the selected samples may discover its designated environment. This preliminary research step is performed to visualize, which malware samples communicate with their corresponding C&C server. In chapter 3 the cross-referencing idea will be explained how the prototype detection tool named "XCOM" came to existence. In chapter 4 the XCOM prototype detection tool structure will be explained in detail. Furthermore information will be provided on how the XCOM tool operates and the detection results on 21 APT malware samples. Chapter 5 provides a small prototype malware named "MOCX", which was designed accordingly to support the outcome on the XCOM detection tool. The MOCX malware will be tested with commercial firewall

applications in order to test its credibility. In chapter 6 challenges and solutions will be provided for more direction on future research with the current state of this project.

Contents

Declaration of Authorship i

Acknowledgements ii

Abstract iii

Introduction iv

Contents vi

List of Figures x

1 APT Forensics 1

1.1 What are Advanced Persistent Threats? . . . 1

1.1.1 The Anatomy of an APT . . . 2

1.1.2 The history of APT attacks . . . 3

1.1.2.1 The basic concept of creating a botnet . . . 4

1.1.2.2 The operations of a Botnet . . . 5

1.1.2.3 The most recent versions of APTs . . . 6

Stuxnet . . . 6

Duqu . . . 6

Flame . . . 7

Red October . . . 7

1.1.2.4 The malware characteristics of the recent APT attacks . . . 7

1.1.3 The future of APT attacks . . . 8

1.1.4 Success probability of APT . . . 8

1.1.5 Other APT variations of threats . . . 9

Insider Threat . . . 9

Industrialized Threat . . . 10

Bring-Your-Own-Device Threat (BYOD) . . . 10

1.2 Which APT attack vectors currently exists? . . . 10

1.2.1 Initial attack vector . . . 11

Spear-Phishing . . . 11

Drive-by-download . . . 11

Online Social Networking . . . 12

Watering Hole attack . . . 12
Longlining . . . 13
Search engine poisoning . . . 13
1.2.2 Exploitation through rootkits . . . 14
1.2.2.1 What is a rootkit? . . . 14
1.2.2.2 Infection and stealthy techniques . . . 15
MBR Infection . . . 15
Alternate Data Streams . . . 16
Slack Space . . . 16
Bad Sectors . . . 16
Hidden Partition . . . 16
Interrupt Hooks . . . 17
Message Hooks . . . 17
SSDT Hooks . . . 17
1.2.3 APT Evasive techniques for avoiding detection by authorities . . . 17
Zero-day exploit . . . 18
Encrypted Network Communication . . . 18
Exploitation of digital signatures . . . 18
Drive-by-download evasion . . . 19
Fast-Flux exfiltration . . . 19
Avoid detection through network packet manipulation . . . 20
Logic Bomb . . . 20
1.3 How can APT attacks be prevented? . . . 20
1.3.1 Detection method for APT attacks . . . 21
1.3.1.1 Intrusion detection systems . . . 21
Signature-Based Intrusion Detection Systems . . . 21
Anomaly Detection Systems . . . 22
Passive and reactive systems . . . 22
Host based intrusion detection systems . . . 23
1.3.1.2 Statistical correlation techniques . . . 23
Counter method for Drive-by-download . . . 23
Tracking down Trojans . . . 24
Multiple Sensors Scanners . . . 24
1.3.1.3 Defend techniques against APT . . . 25
Email inspection . . . 25
Protection by an DNS-based Black hole List (DNSBL) . . . 25
Sandboxing Applications . . . 26
1.3.2 Tracking Methods . . . 26
Fast-flux tracing . . . 26
Utilizing honeypots . . . 27
Reverse engineering malware samples . . . 28
1.4 Tackling the APT in the operation phase . . . 28

2 Forcing communication with C&C servers 31
2.1 APT Counter measures for sandboxed environments . . . 31
VMware . . . 31
VirtualPC . . . 32

VirtualBox . . . 32
2.2 APT malware samples . . . 34
Careto . . . 34
IXESHE . . . 34
Lurid . . . 34
Mirage . . . 35
Snake . . . 35
Sykipot . . . 35
Taidoor . . . 36
Winnti . . . 36
2.3 Triggering multiple APT malware samples to communicate to C&C servers through custom sandboxes . . . 37
2.4 APT malware samples that communicate to C&C servers . . . 38

3 Cross-referencing idea **40**

4 The XCOM experiment **43**
Structure . . . 43
OpenWRT . . . 44
Server . . . 44
Client . . . 45
Apache Thrift framework . . . 45
Negative Result . . . 45

5 Prototype Malware MOCX **48**
Structure . . . 48
Commercial Firewall Solutions . . . 49
Comodo Internet Security . . . 49
FortKnox Firewall . . . 49
Outpost Security Suite . . . 49
ZoneAlarm Security Suite . . . 50
XCOM prototype detection tool . . . 50
Circumventing Host-based Intrusion Prevention Systems (HIPS) . . . 50
Overview Firewall Result Detection . . . 50

6 Challenges and Solutions **53**
Malware Samples . . . 53
XCOM Detection Tool . . . 53
MOCX Prototype Malware . . . 54

7 Conclusion **55**

A Overview APT **57**

B Correlation Table Wireshark Experiment **59**

C Correlation Table XCOM Experiment **60**

D Detection Result XCOM Experiment Scenario 1 **62**

E Detection Result XCOM Experiment Scenario 2 **64**

Bibliography **66**

List of Figures

1.1 the history time table of the botnets taken from[2] 4
1.2 Matrix table taken from [3] 7
1.3 drive-by-download process taken from [4] 12
1.4 Kernel rings 15

2.1 Detection VMware 32
2.2 Detection VirtualPC 32
2.3 Detection VirtualBox 32
2.4 Network Architecture Experiment 38
2.5 Results Wireshark Experiment 39

3.1 XCOM Network Architecture Sketch 41
3.2 Windows 7 Network Subsystem Architecture 41

4.1 XCOM Communication Process 45

5.1 Screenshot Outpost 52

Chapter 1

APT Forensics

1.1 What are Advanced Persistent Threats?

Advanced Persistent Threat is an acronym developed by MANDIANT[1], but was mentioned earlier by Mike Cloppart[1], which describe a sophisticated threat which can be divided into the following parts accordingly to the following researchers[5, 6]:

- Advanced(A): This stands for that an adversary which is highly advanced in hacking skills. This adversary consist out of a team of specialized trained people often backed-up by an unknown entity which provides them with financial or technical support[6]. Their inventory consists of network intrusion technologies and they are able of creating their own exploits[5].
- Persistent(P): Persistent refers to an adversary which passively remains hidden illegally on corporate networks after infiltration until the specific target goal has been reached. The adversary prefers to stay for a prolonged time on the corporate network, shown in [1] the median number of days were 243 days active on the network. The perpetrators also create several back-doors for reentry in case the security environment changes as shown in case studies in [1].
- Threat(T): Threat stands for causing damage or stealing intellectual property and damaging legitimate (web) services.

Advanced persistent threat is real threat in the current business environment[7]. Not like other researchers and marketing managers claim that it is a threat in order to sell their security products and services[8]. The goal for these adversaries these days

[1] http://digital-forensics.sans.org/blog/2009/07/22/security-intelligence-introduction-pt-1

is focusing on business intellectual property and sensitive information[1]. Their target could be a high-valued individual such as a political representative [9] or company in general[10]. There is a in disagreement about APT whether it is fully automated or manual controlled by an adversary[11]. Although whether it is automated or manually performed both share the same structure of attack shown in the next chapter. A simple definition of APT would be either digital espionage[9] or damage to intellectual property see Section 1.1.2.3.

1.1.1 The Anatomy of an APT

Advanced Persistent Threats all share the same characteristics as they go through the attack process, for that it exhibits certain phases which the attack goes through before the final goal is reach from the adversary perspective. This fact applies to all APT attacks that currently exist[6]. The following phases describe how an APT attack is performed. Although other researchers present it in the form of kill chains, which ultimately result into the same abstract structure[12].

Reconnaissance:

This phase involves getting as much information as possible on the designated target at hand. Therefor besides the actual target also other information sources are commonly exploited, e.g., social networks [13], Internet services, or dust bins of employees. The perpetrators will try and find out as much as possible about the employees of a company and create profiles from them in order to establish a organizational topology of the company. Furthermore they tend to use common network scan techniques like port scans to detect potential vulnerable web services for infection later on.

Delivery:

The delivery phase tends to lure potential intermediate targets into the exploitation phase. This could be done through spear-phishing techniques[14] or through other business threats see Section 1.1.5. Which is sending a specific email to a designated target which sounds legitimate for the target user in order to open up an attachment or web link. In case of a HTTP link, then a technique called drive-by download is activated[15, 16]. see Section 1.2.1

Exploitation:

After the user clicked on the malicious link or opened the malicious attachment, the exploitation phase starts. The user's machine gets infected with malware, more specifically it gets infected by a rootkit. This is an example of an automated approach in order to exploit a system. Other manual methods exists such as SQL

Injection or Cross Site Scripting[17]. This malware is able to control the user's machine entirely. It can monitor screen output or log keystrokes. Furthermore it is able to propagate through scanning the local network for potential vulnerabilities for infecting them. All these actions are hidden from the user's machine, because the rootkit tends to hide itself. See Section 1.2.2 for more details on this. The Malware will try and set up a Command and Control connection to the attackers server in order to receive more specific commands.

Operation:

In this phase the attackers scan the internal network, looking for the targeted information they want to exfiltrate. Again they create profiles of how the internal network is structured. If they realize that the targeted information is not reachable, due to tightened security measures, then they escalate their privileges by sending out a new spear-phishing emails in order to gain higher credentials, until they have the correct security level.

Data Collection:

The data collection phase is all about retrieving the target information and setting up intermediate server often called staging servers. Examples of targeted information could be insider knowledge from political emails or a closed-source code from a software company. Here the sensitive data is being encrypted and compressed, so that in the exfiltration phase the data can be shipped out. Often the attackers create multiple C&C connection in order to prevent for the target network from changing the network infrastructure due to security policies changes.

Exfiltration:

The final stage is about ex-filtrating the target information to the drop servers. The attackers could be using certain evasive measure in order to avoid detection and tracking. One of these evasive measure is the fast-flux technique, more on this in Section 1.2.3 If this phase is successful then the attackers have succeeded in their attack and the target data is compromised and stolen. Often when they ex-filtrated the target information they hide their traces, which makes for forensic investigators extremely hard to track their tracks.

1.1.2 The history of APT attacks

The question we must ask ourselves is: "Where do APT attacks come from originally?" Before the term APT was used, there were some primitive versions of APT in the field. In today's setting an APT attack is based on fundamental platform called a botnet[18] which contained several Command and Control servers(C&C). These were

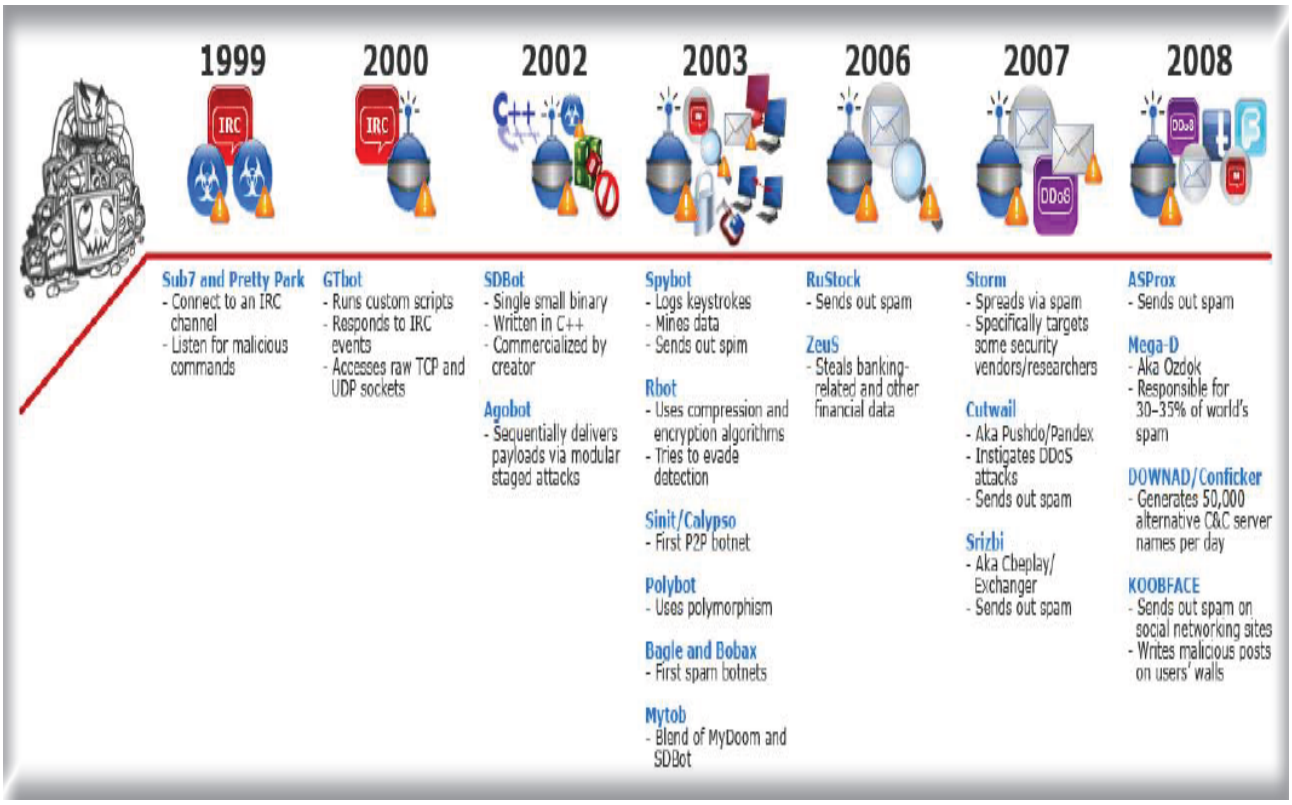


FIGURE 1.1: the history time table of the botnets taken from[2]

mostly automated by Bots (a.k.a. Zombies). Botnets exist already for a very long time since 1999[2]. Before APT attacks were carefully coordinated by a criminal organization directing an APT attack, Script kiddies were involved in managing a botnet in the beginning. Figure 1.1 show the history time table of the botnets.

1.1.2.1 The basic concept of creating a botnet

Script Kiddies are also known as botnet herders. In the beginning botnets were created with IRC (Internet Relay Chat) networks. What a script kiddie would do to initially setup a botnet is, first get their hands on a toolkit of a Bot. A Bot is an infected PC, which typically contains a Trojan. After getting the source code they would manually infect a server with some exploits in order to compromise the server. For example, this could be a web server, but any other web service would suffice. After the server has been compromised, the script kiddie would setup up C&C server and configure it so that the Bot can connect back to the compromised server. The bot itself would be compiled and loaded with exploits[19] in order to exploits other systems. If one system got infected with the Bot, then the Bot itself would propagate automatically using the exploits it has in his inventory. This is the basic concept of a botnet[18].

1.1.2.2 The operations of a Botnet

All the operations of a botnet aim to monetize the network. Even Script kiddies used botnets back in the old days to generate money. In the following section the script kiddie is referred to as the perpetrator. The operations are as follows[18]:

DDOS Attack:

Although this operation doesn't directly generate money, it can be extended with extortion or blackmail in order to gain revenue. A Distributed Denial Of Service attack is one of the most common attacks today. It consists out of C&C server that sends out a command to all Bots with a specific IP address as target command. Then all the Bots send IP packets to the designated IP address, which in turn overloads the legitimate web server or other web service. It can sometimes be so devastating that it crashes the target computer/server.

Spamming:

Spamming is still most of the common operations today. It consists out of several emails being transmitted from all the Bots in a botnet. Today they use compromised email relay server in order to look more legitimate for the receiving end.

Financial Fraud:

Because of key loggers that are within a Bot, the perpetrator is able to retrieve all keyboard inputs done on the victims computer. And so it is able to retrieve sensitive information like credit card information or social security numbers for impersonating the victim. Even username and password from bank sites are not uncommon. For the perpetrator it was then possible to steal someone money for their own gain.

Search Engine Optimization poisoning:

Through SEO poisoning the perpetrator is able to promote their own malicious site which could sell bogus products or it could be a legitimate site they own in order to improve the page ranking mechanism of Google.

Pay-per-Click (PPC) fraud:

This technique is seen in legitimate ways as in you install a freeware software and the installer includes some advertisements which are packed with the installer. The user is then able to accept or decline the advertisement software. But with PPC the perpetrator modifies the original advertisement installer package in order to let it execute without the users consent. When this is combined with the execution of all the Bots within a Botnet this can generate a lot of revenue.

Corporate and Industrial Espionage:

With the botnets today a perpetrator is able to hire a botnet for performing malicious actions in order to steal corporate information. Although the security community still debates over whether it is being used for digital espionage we know now that it has been done with APT attacks in todays setting[3]. The perpetrator would typically use a botnet in order to avoid detection, through the use of fast-flux techniques see Section 1.2.3.

Bitcoin Mining:

Bitcoin mining today is a legitimate business, although if a botnet is involved then each Bot will hold some Bitcoin mining software in order to mine this virtual currency. This is done by performing some complex hash calculation with the CPU processor, while the computer is idle in order to not raise any suspicion.

1.1.2.3 The most recent versions of APTs

The four most recent APT attacks that have been detected the last 5 years were: "Stuxnet, Duqu, Flame, and Red October; although it is still speculated that some of these APTs have been active for almost 10 years[3]. The following subsections describe the characteristics of these APT's. See Figure 1.2 for the different characteristic aspects amongst the APT's. There are also more APT variants, due to time constraints other APT's were omitted here.

Stuxnet Stuxnet got detected in June 2010. This is believed the first mayor digital sabotage APT attack in history. It was designed to be fully autonomous as in "launch and forget" APT. The target was an Iranian nuclear enrichment facility. It's goal was to sabotage programmable logic controllers (PLC) within the facility. They APT attack eventually succeeded and caused massive damage which effectively slowed down the operations by four years[20]. It is said that the initial infection is unknown of Stuxnet[20]. Although some assumptions can be made because it contained malicious code which infected removable drives. It propagated through a remote network exploit. Eventually the infection reached the PLC systems, which in turn were destroyed by feeding the monitoring controls false readings. This resulted in overloading the physical structural components within the facility[10].

Duqu Duqu was detected in September 2011. Duqu had similar characteristics like Stuxnet, only it had an different goal then sabotage. Duqu goal was mainly espionage and did not infect more than 50 targets worldwide[21]. It remained 30 days active after

APT's name	Stuxnet	Duqu	Flame	Red October
Active since	June 2009 (2005)	November 2010	May 2012 (2006)	May 2007
Detected	June 2010	September 2011	May 2012	October 2012
PE executable	DLL		OCX	EXE
Initial infection	Unknown	MS Word	Unknown	MS Excel/Word, Java
Self-replication	Removable drives, Over the network	Manual replication only		
Rootkit functionality	Yes		No - Not yet known	
Key logging module	No	Yes		
Targets sec. products	Yes			No
Encryption	XOR	XOR, AES-CBC	XOR, Substitution, RC4	XOR
Target	Sabotage	Information gathering		

FIGURE 1.2: Matrix table taken from [3]

initial infection. After that period it self-destructed, but the perpetrators were able to extend the deadline if needed. Like any other malware it contains malicious tools for monitoring and logging sensitive information.

Flame Flame got detected in May 2012. But apparently people are still speculating for how long it has been active[21]. An interesting fact is that Flame has a malware size of 20MB, including all the modules it contains. Between Flame and (Stuxnet or Duqu) are no similarities so we can assume that it has been developed by a different party, probably state funded since it has a high complexity.

Red October It targeted mainly the Russian Federation, thus its name. It has been discovered in October 2012[22]. It targeted mainly governmental and scientific institutions. It's main malware component is small, but it had a plugin architecture which allows it to grow larger. It had a sophisticated operation, which could undelete files from removable drives[22].

1.1.2.4 The malware characteristics of the recent APT attacks

What is interesting is that most of the recent APT attacks use 32-bit Malware. This is probably due to the complexity of 64-bit architectures. Let alone the signature based protection system in windows 7 systems. But probably in the future this won't hold anymore, since the SANS institute already discovered that Malware exists, which can bypass the signature verification mechanism of windows 7 and above[23]. Typically the initial attack vectors from all the four APT attacks were using techniques such Spear-Phishing [14] or through office documents infection in email attachments[9] or through infected PDF files[24]. In [3], Nikos Virvillis et al. suggest a method for preventing infection through office documents by using the latest version of Microsoft Office, because

it includes a sand-boxing principle. The only problem is that Malware developers will work around this in the future as it is described in [23]. As claimed in [3] that all the recent APT attacks made use of zero-day exploits the threat continues to rise, although they claim that Microsoft systems should be properly patched it is impossible to prevent a zero-day from exploiting, when no patch exists yet. The network communication the recent APT attacks used were mostly encrypted with an XOR encryption, which in turn effectively circumvents the network intrusion detection systems, because they rely on signature based matching upon packet inspection. What is interesting is that Stuxnet and Duqu used digitally signed binaries in order to avoid detection for Host Intrusion Protection Systems (HIPS), because most of them skip binary signed files for scanning out of performance reasons.

1.1.3 The future of APT attacks

Over the future of APT attacks can only be speculated and debated, since we don't have capabilities of looking into the future, but several interesting fields have appeared so far. Since the development of smart-phones has dramatically increased over the past years[25]. It is possible to expect the next APT attack to be happening on mobile devices. Not only on mobile devices has the threat been increased,[26] but also recent new anonymity network protocols have appeared such as: "Aqua,Tor, Tarzan, Dissent, P5, MixMinion" over the past years. If these two concepts are being combined for APT attack in the future it can have severe damage capabilities. Although some anonymity network already have been comprised in the sense that traffic analysis is possible on anonymity network protocols such as: "Tor" [27, 28]. But if several new techniques are combined with today's technology, then we get prototype systems such as mobile controlled P2P sms botnet[26] or ASP2P social network botnet [29]. The rise of these dangerous new botnet concepts, must warn us about a potential new APT attack is waiting to occur in today's technology environment. Furthermore an APT attack consists out of advanced components in order to be successful. If for example a BluePill like rootkit[30] is being used in an APT attack combined with these new concepts of a botnet, an dangerous APT attack is waiting to be launched.

1.1.4 Success probability of APT

In order for an APT attack to be successful it needs one of the following criteria's:

- The initial infection method needs to make use of zero-day exploits for spreading the malware as ingredient for launching preparations. Because zero-day exploits

are vulnerabilities that have been discovered, but have not been patched[19]. Other exploits might also be successful, but the success probability would decrease, because of current detection measures.

- Then the malicious party needs to setup several initial attack vectors See Section 1.2.1 in order to propagate the initial infection for a higher success probability.
- After the exploit shell code has infiltrated the victims computer. The victim machine downloads automatically a sophisticated rootkit preferably a virtual-based rootkit, for which the detection methods are currently still unknown.
- The rootkit installs a sophisticated Trojan, which has a large supplies of tools in order to steal corporate or sensitive data.
- This Trojan needs to communicate then with a C&C botnet, which uses techniques like anonymity network protocols in order to avoid trace and prosecution.

These success probability suggestions can create a possible APT attack, that cannot be detected within the current technology era. More information to backup these criteria's can be viewed in Section 1.2 and Section 1.3. In order to be ahead of these new potential APT attack methods it is recommended to have multiple security defense layers within a company[31]. In the future we could expect a more advanced Blue pill[30] rootkit, that would operate like a dual boot operating system. This would imply that the victim machine host a dual operating system. One is the legitimate one and the other is the malicious operating system hidden in slack space see Section 1.2.2.2. This would totally avoid tampering with the original operating system, which would render current detection methods useless.

1.1.5 Other APT variations of threats

These are the threats that closely resemble APT.

Insider Threat There are some other threats that can make an APT attack easier to perform under the right circumstances. Insider threats is an example of such. The Insider threat implies that there is an malicious employee inside an organization that is motivated by financial reasons in order to perform malicious acts for a third party. This is considered a great risk for corporations [32]. From the attackers perspective the benefit of the Insider threat during an APT attack is that an attacker can skip several APT phases in the APT process. Since the attacker already has gained indirectly entry to the designated organization. This could effectively result in skipping the first 4 phases

of an APT process, which are the reconnaissance, delivery, exploitation and operation. This only applies if an attacker performs an attack manually to steal sensitive data. For persistence exploitation is required for reentry or in some cases the compromised employee is part of the perpetrators criminal organization[11]. According to Verizon[33] 48% of data breaches in 2009 were caused by insiders, where 90% of this percentage had malicious intent.

Industrialized Threat The industrialized threat beholds monetizing as fast as possible from the attackers perspective. The attackers deploy on a large scale, malware which focusses on stealing someone's online bank credentials or credit card information. These can then be used for example malicious purposes such as purchasing illegal goods or any other form of products or services. An example of this malware is SpyEye[18], which is example of stealing online bank credentials. This threat is often backed up by a large botnet and it would target on a large range but less effective payload for infiltrating target computers.

Bring-Your-Own-Device Threat (BYOD) Organizations today face challenges of protecting their corporate network against employees that prefer bringing their own devices in order to use them inside the organization. Most companies up until now are not prepared for BYOD threat. [2] or don't consider it a huge threat according to the global security survey. The problem here lies in the fact that if employees inside a company get annoyed by to tight coupled security measures that will slow down their work, they just deploy mifi- connection, which implies setting up an external internet connection in order to perform their work further without any delays. These procedures done by employees will render a corporate intrusion detection system useless, since it only guards the network perimeter see Section 1.3.1.1.

1.2 Which APT attack vectors currently exists?

This scope of this chapter is that it focuses on several aspects of an advanced persistent threat phase previously explained in Section 1.1.1. Here we go into detail how the initial infection method occurs and how in detail rootkits will operate within an APT attack if used for infection. This section will conclude with some APT evasive maneuvers in order to avoid detection.

[2] http://www.infosecurity-magazine.com/view/37051/

1.2.1 Initial attack vector

The initial attack vector can be described as the first attempt to exploit a target machine. This happens after the reconnaissance phase and inside the delivery phase. The following initial attack vectors will be explained more in detail. In order to perform an APT attack initial delivery or exploitation must occur. This is done in several known ways. First of all the term "phishing" started originally in 1995 by Jason Shannon of AST Computer[3] but the company is now out of business. Phishing consists of sending out massive amount of emails in hope that someone will fall for the trap. They often tend to masquerade themselves as legitimate bank emails or other trusted entities[9]. Other variants of phishing are: "Spear-phishing, Clone-phishing and Whaling". Clone-phishing consists of cloning a legitimate email that contains a download link and it will change the download link. Whaling is a term used for targeting high-valued targets within a company.

Spear-Phishing Spear-phishing on the other hand is a more sophisticated attempt in deceiving the target user[14]. The difference between a normal phishing email and a spear-phishing email is that it's specifically crafted for target person in order to present itself more legitimate. They often tend to use legitimate looking trusted sender. Sometimes the term "Whaling" is a synonym used for spear-phishing. According to Trend Micro[34] 94% of all the spear-phishing email that they have detected use malicious email attachment. The other 6% consist of email without attachments, this implies that the email contain malicious links in order to lure the target victim into clicking on it. A different term can be associated with the last phrase, which is called drive-by-download. See the next section for more details.

Drive-by-download A Drive-by-download[4] describe a process which the computer user is lured into visiting a website which he might have accidentally clicked on or without the users consent. This could be through various ways, such as clicking on a malicious web link inside an email or by mistake visiting the wrong site which hosts a malicious advertisement. In today setting it is not required anymore clicking on advertisement in order to get a redirect to a malicious site. A simplified explanation would be loading a genuine webpage, with some malicious code inside which causes to redirect through injection to a landing server where the user is furthermore being exploited. See Figure[drive-by-download]. An Drive-by-download is often combined with a phishing email, where the malicious link is provided within the phishing email. The reason for multiple servers you see in Figure[drive-by-download] is in order to avoid detection and

[3] http://www.computerhope.com/comp/ast.htm

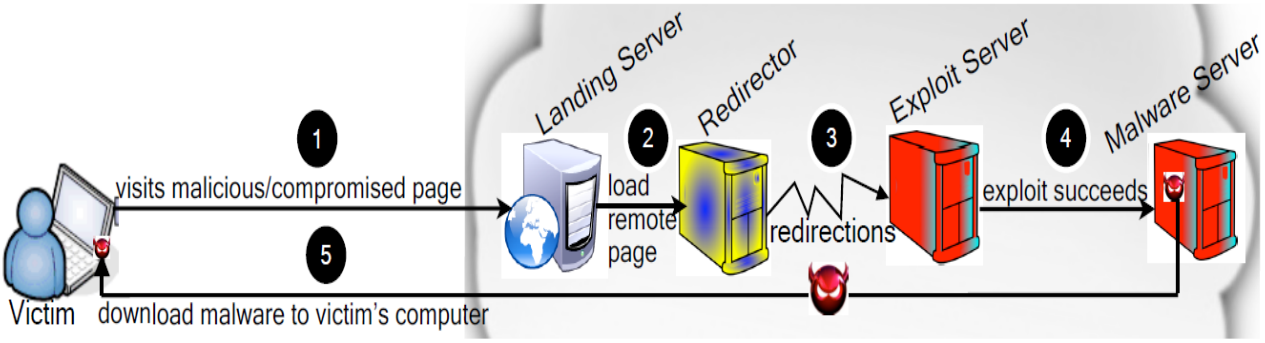


FIGURE 1.3: drive-by-download process taken from [4]

traceability. For example if the landing server domain is reported for a known malicious phishing website, then the adversary can switch the intermediate servers in the process in order to retain some components of the drive-by-download. Although there has been sophisticated methods in order to detect these Drive-by-downloads[4], they exists still at large because of extensive evasive counter measures see Section 1.2.3.

Online Social Networking Today's online social networking has advanced significantly in the terms of technology. This also creates possibilities for malware developers to create malicious new ways to use online social networking. Some researchers[16] even noticed that botnets behave with similar characteristics as those of online social networks. We can anticipate that drive-by-downloads will be used in combination with social networks. With the development tools we have today in social networks we can make use of several API's (Application Programmable Interface) in order design very nice programs, but here also lies the catch that malware developers can do the same. Imagine a scenario where an adversary is able to make use of these API's and creates a piece of malware which automatically sends malicious messages towards other users within the network. Therefor this can contain a drive-by-download link in order to infect the target machine. For the reconnaissance phase this is a perfect opportunity to gather as much as possible about a company or its members in order to fuel their APT campaign[13, 35].

Watering Hole attack The Waterhole attack consists out of attack method, which targets a company or high-value target in order to infects affiliate websites as much as possible. This implies that an attackers tracks the social behavior patterns of users (possibly through social media) what kind of websites they prefer visiting. This attack is often combined with the Drive-by-download method[36]. As soon as the an adversary has established profiles of which websites the target prefers, he will then infect those websites with various other attack tools in order to create a exploit loaded website. When the

target visits the website he will then get infected and an adversary can enter through a backdoor.(Malware,Trojan) There have been reports [37] of sophisticated waterhole attack which infiltrated open source software code, which is largely available. It infected the source code with a backdoor and the infection evolved through software engineers using the third party code. This should raise awareness that software engineers should not blindly trust third-party code. A software company should train their staff in order to prevent such a sophisticated attack.

Longlining Longlining [38] is one of the latest advanced phishing methods that are available in today's market. It operate on a high-overall volume amount of emails and a low volume on emails for each targeted company. It operates under the condition that it is able to hide itself under enormous amount of fast switching IP address web links, similar to the fast-flux technique see Section 1.2.3. It also includes randomization techniques in order to spoof the senders address. The longline operation runs within an extremely short time period(3 hours), which sends initially a probe email in order to check whether conventional email filtering systems can detect it's prototype sample. If the target can be infected with the initial probe email, then the content is efficient. If it is not able to penetrate the target system, then it uses advanced obfuscation techniques in order to dynamically change the emails content. This is supported with nice visual markup html style email messages, not like the original phishing email with bad grammar and plain text markup.

Search engine poisoning Search engine poisoning(SEP) is a technique in order to lure as much possible visitors to a malicious website[39]. This is done by abusing normal legitimate SEO (Search Engine Optimization) techniques. Normally when a website is being new created or a website is being updated an search engine crawler often called as robot is being used in order to analyze these websites. They have a special User-Agent value in order to distinguish them from normal web browsers. They read meta-data from websites in order for the page ranking process to process them. The website developer is able to define the keywords used in the meta-data and would normally enter relevant search key terms targeting his website. But malicious web developers have begun entering popular search key terms into their meta-data tags often retrieved from Google Trends[4]. This ultimately results in poisoning the search results in the designated search engine. These malicious websites host often malware, which is similar to Drive-by-download see Section 1.2.1. Not to be mistaken with search inflating technique. This implies that a malicious website tends to copy-cat a genuine popular website with a high page ranking in order to retrieve similar amount of visitors. The

[4] http://www.google.com/trends/

difference between these two techniques is that SEP contains random popular search key words, while search inflating only focusses on the scope popular website. The SEP technique can be combined with several other initial attack vectors previously explained in this chapter.

1.2.2 Exploitation through rootkits

If an APT attack occurs within a targeted company it is usually done with the help of a rootkit. For example the "Stuxnet" (Section 1.1.2.3) APT attack used a kernel-based rootkit in order to hide its presence. From the security community[5] it is considered an extremely large threat if a rootkit is operating on an OS.

1.2.2.1 What is a rootkit?

A rootkit is a subcomponent of the category Malware, which provides the functionality to hide itself within an OS. It is not only limited for hiding itself, but also other types of Trojans. Usually a rootkit and a Trojan work together. The Trojan contains a set of tools to monitor the computer in various ways, such could be monitoring the screen by creating screenshots of the desktop computer of the victim. It can also provide more functionalities such as downloading files or uploading them. The more recent Trojans can log keystroke inputs and can perform Man-in-the-browser (MiTB) attacks[40], which are very dangerous when performing online banking activities. One of the most dangerous tasks of a rootkit would be hiding inside the operating system and hiding the Trojan which contain all the tools. Because rootkits are small they often do not have functionalities like the Trojans do. This is due to the reason that they reside in places where they have memory constrains or limited API capabilities, because of the environment they are in. There are in general 4 protection domains (often called rings) a rootkit can reside in. See Figure 1.4. The Figure 1.4 typically displays an privileged environment layer. Although the modern Windows operating system uses only 2 rings, which are Ring 0 and Ring 3. This was due to original hardware implementations constraints of Windows. Rootkits in today's world vary from Ring 0 to Ring 3. Mostly Rootkits that are in Ring 3, which is the user-mode level can be detected by ordinary security applications scanners, but rootkits that reside in Ring 0 are considered to by kernel-based rootkits. These are considered dangerous, because of their privilege level they are difficult to remove. There is also a different kind of rootkit which has an exceptional location to rings. This is often called as a hypervisor rootkit, such as the blue pill [30]. This is a rootkit which resides below ring 0, which is often refer to as ring -1. It

[5] https://www.owasp.org

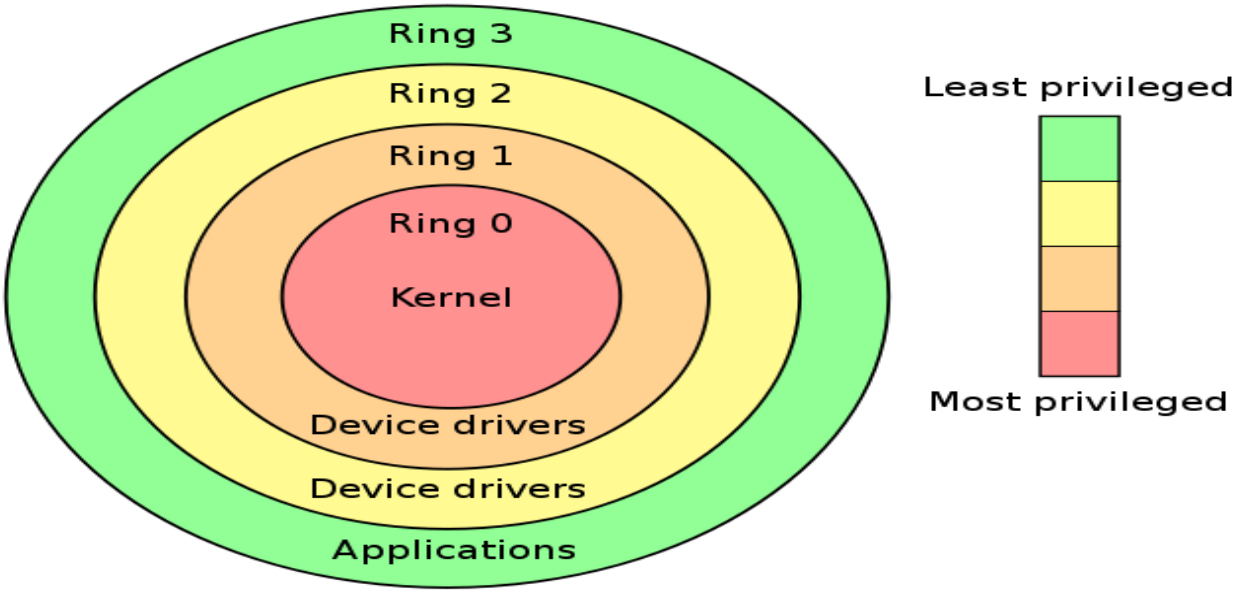


FIGURE 1.4: Kernel rings

is said[30] that these hypervisor rootkits create their own virtual based privilege layer, which is commonly used for genuine purposes in virtual based appliances, like VMware based applications. This rootkit can defeat any kernel-based rootkit detection methods, through the fact that it is operating in a higher privilege environment. Although some people claim it can be detected by some time-based methods.[6] These types of rootkits are rather new and can be expected to evolve in newer APT attacks.

1.2.2.2 Infection and stealthy techniques

This subsection will describe some of the techniques in order for a rootkit to infect and remain hidden.

MBR Infection This technique common for rootkits to do, but in the past recent years many tools already have capabilities for detecting these infections. Still some rootkits in the wild apply this technique, because of computers that are not properly patched or no anti-virus has been installed. The Master Boot Record (MBR) contains the startup sequence of an operating system. When a rootkit tries to modifies the MBR it loads itself during the startup of the computer from a hidden location on the system, these locations could be for example: "hard drive, bios firmware, PCI hardware memory". After the rootkit is loaded into memory then the original boot sequence is executed and the OS is started up. The following subsections describe infection methods for rootkits to stay hidden or to store itself.

[6] http://theinvisiblethings.blogspot.nl/2006/06/introducing-blue-pill.html

Alternate Data Streams Alternate data streams(ADS) is developed by Microsoft and only currently resides in NTFS partition format. With alternate data streams it is possible to store files hidden without the operating system to be able to see it through conventional ways such as the windows explorer application. This was originally designed, because Microsoft wanted to provide some backward compatibility with Macintosh Hierarchical File System (HFS). For example if a user would create the following command:
(C:\echo "test" >test.txt:hidden.txt), then the ADS is able to store the text "test" in the hidden file "hidden.txt". Thus for rootkits it is also possible to store executable files with this same technique, but this technique is only being used by some older rootkits, because todays anti-virus applications are able to scan these locations.

Slack Space Normally when files are stored on a hard drive it is stored in "chunks". These are fixed blocks of for example 1024 bytes. So if you would have a file that has a size of 512 bytes, then the other half is remained unused on the hard drive. These technique is done out of performance reasons when reading data from a hard drive. Of course rootkits are also capable of using these unused spaces in order store themselves hidden. Thus it is difficult to determine for an anti-virus or anti-rootkit to detect these spaces, considering these unused space might also be just some random data.

Bad Sectors Bad sectors were used whenever a hard drive succumbed to some physical hardware problems of reading the disk drives within a hard drive. The operating system would mark several sectors as "BAD" so that the operating system would skip reading them. This also would create interesting locations for rootkit to hide themselves in and mark itself as "BAD" in order to avoid scanning by anti-virus applications. The benefit would not only by that a rootkit could hide itself, but also be ensured that no other program would overwrite its location, thus guaranteeing survival.

Hidden Partition A hidden partition is sometimes created by rootkit in order to store their malicious payload. This hidden partition could be place at the end of a physical drive or stored in a fake file. It is risky for the perpetrator to allocate at the end of the drive, but some other hooks need to be implemented in order to avoid the operating system to overwrite the particular section of the drive. A fake file would be better and easier to manage, which could be similar to the TrueCrypt[7] container used in their system. Modern rootkits also encrypt this hidden partition in order to avoid detection.

[7] www.truecrypt.org

Interrupt Hooks Interrupt hooks are use normally to intercept system calls of a computer. Interrupt hooks are commonly used together with MBR infection, since it involves modifying the MBR in such a way that it hooks on a specific interrupt INT 13h. This is usually used for accessing the hard drive. Hooking this particular machine instruction allows a rootkit to control the hard drive on a very low level, which will in turn allow it to avoid detection from the OS it runs on.

Message Hooks Message hooks are used to intercept communication between programs or the OS. Since the OS manages message ques. Usually a user-mode rootkit or Trojan hooks into message hooks such as "WH_KEYBOARD" or "WH_MOUSE", which ultimately results in listening all physical inputs done on the keyboard or mouse. This is how key loggers usually operate.

SSDT Hooks The System Service Descriptor Table (SSDT) is used by the Windows OS. The equivalents of this table in Linux OS is System Call table. The benefit for a rootkit to manifest itself in the SSDT is that it is required for the OS to use this or else the OS cannot operate. A rootkit will settle itself in between the SSDT and other programs. So if any other program tries to read from the SSDT or tries to implement a hooking procedure, then the rootkit will operate in between. The reason why it has been so successful is that system calls that access for example the hard drive can be circumvented in a way that no program is able to read the location of the rootkit. Often anti-rootkit or anti-virus also use the same method in order to detect rootkits, but if a rootkit has already manifested itself on the SSDT, then it is very difficult for an anti-virus to detect this. Their exist some anti-rootkit programs, which scan the SSDT for any modifications, but these are very specific for certain rootkits. Although due to the new implementation of 64-bit architecture, Windows OS developed several countermeasures for hooking or modifying the SSDT. The method they use is the signing critical system files in order to protect the SSDT. See Section 1.2.3 for exploitation of this new signature protection mechanism of the Windows OS.

1.2.3 APT Evasive techniques for avoiding detection by authorities

Advanced persistent threat use several sophisticated evasive techniques in order to avoid detection. This section is devoted for listening these techniques. All these techniques have their individual capabilities in order to avoid detection in a certain APT Phase see Section 1.1.1.

Zero-day exploit A Zero-day exploit[19] is a well-known technique for avoiding detection in the Delivery and Exploitation phase of an APT attack. It uses a vulnerability in software, which is often also called as a software bug. As soon as it is discovered on for example OWASP[8] the perpetrator can make use of this vulnerability. Since it is described in detail how to trigger the software bug. If such a software bug is triggered, then an adversary can write a custom exploit which abuses the vulnerability and so its malicious payload for infecting the designated computer can be activated. This allows an attacker to advance to the next phase. Sometimes an adversary has a private exploit, which is similar to a zero-day exploit, except the vulnerability is not known yet to the security community. Often drive-by-download make use of this technique see Section 1.2.1. For example when an email contains a pdf or office document exploit[9], then it's loaded with a zero-day exploit in order to avoid detection by email attachment scanner. Since these scanners use signature based techniques a zero-day exploit is not known and thus it would allow it to pass through.

Encrypted Network Communication The recent APT attacks: "Stuxnet, Flame, Duqu" were designed to detect and evade common anti-virus applications. Not only limited to avoiding applications, they could also encrypt their network traffic to the C&C server. Stuxnet used a simple XOR encryption, while Duqu already used AES-CBC encryption. Flame used multiple encryption techniques like: "XOR, Substitution, RC4"[3]. Together with these encryption schemes they used a normal HTTP protocol or sometimes even HTTPS. This resulted in failure to recognize when conventional intrusion detection systems (IDS) tried to analyze the network traffic, because these conventional IDS use signature based detection methods in order to detect malicious traffic. This evasive technique allows an adversary to secretly exfiltrate sensitive data in the APT Exfiltration phase. To furthermore increase their stealth capability, they also tend to hide their network traffic on the local victim's machine, through infecting the network socket API.

Exploitation of digital signatures APT attacks are becoming more dangerous as they tend to subvert digital signatures in such a way that they are able to create legitimate certificates and thus applying them malicious files to appear legitimate. This is done by stealing a root certificate, which was performed in the DigiNotar incident[9] With a stolen root certificate it is possible to create new certificates, which can be used to sign malicious files. This would effectively disable the 64-bit kernel mode signing technique used in Windows 7 and above[3]. Furthermore anti-virus applications would

[8] https://www.owasp.org/

[9] http://www.onderzoeksraad.nl/uploads/items-docs/1833/Press_release_DigiNotar_280612_EN_opgemaakt.pdf

skip signed files in order to gain more performance when scanning the computer for vulnerabilities. This evasive technique would effectively grant the Operation and Data Collection phase of an APT attack a free pass.

Drive-by-download evasion The evasive techniques used by a drive-by-download[41] are consist mainly out of 3 characteristics which are redirection, fingerprinting and obfuscation. These techniques are not necessary in order to perform a drive-by-download attack, but it makes it for forensic experts difficult to analyze where the attack originated from. Redirection is an evasive technique, which allows an adversary to redirect the targeted victim to a large number of different websites. If for example a redirection is performed it will be redirected to 10 servers, whereas the victim reaches the 5th server it will enter a DNS pool of addresses, which would be randomly switched, similar to the fast-flux technique Section 1.2.3. The fingerprinting technique allows an attacker to profile the victim machine by analyzing, which plugins the browser uses or which user-agent the victim uses. This is done in order to send specific exploits for targeted vulnerabilities in these plugins. The same counts for the user-agent, but then the target would be the browser itself. Since these are specific exploits for specific target environments it would retrieve its exploits from different servers, which in turn makes it difficult to track down all the exploits hosted in web servers. This is furthermore complicated by the obfuscation technique, which allows for an attacker to obfuscate JavaScript code by a certain degree that it is able to hide the malicious code from web scanners. The most advanced evasive technique is Domain Generation Algorithm-based (DGA) [42]. This allows an attacker to use a algorithm for rapidly changing DNS names, making it also more difficult to track. This grants a sophisticated APT attack successful pass through the Delivery and Exploitation phase.

Fast-Flux exfiltration As the name implies this technique is used for the exfiltration phase of an APT attack. The goal of Fast-Flux [43] is for a DNS name for example: "test.hackorz.com" to have multiple IP addresses connected to it. This is done by switching IP addresses fast in and out of the flux. Due to the nature of DNS protocol only 1 IP address is able to connect at a certain time. By entering a low TTL(time-to-live) value it will allow an attacker to being more difficult to track. Usually the time is set to at least 3 minutes[16], before it switches to a different address. If authorities track this IP address, they will come to know that it is just a redirector for forwarding traffic to the next node. Accordingly to [43] there exist single and double flux networks. Whereas with a single flux network the victim machine does a DNS request for "test.hackorz.com" to a compromised name-server (ns.hackorz.com) which has a static IP address, which then in turn returns a temporary flux IP address. With each fast-flux network there

exists a mother ship, which is similar to a C&C server in the context of traditional botnet architecture. This mother ship then communicates with the compromised name-server for request and response packets. A double fast-flux network is very similar to a single flux network, except it also has flux addresses for the compromised name-servers. This implies that the name-server also switches frequently IP addresses. This ultimately results in difficult tracking procedures for authorities. Fast-flux networks are also used for deploying large amount of spam emails see Section 1.2.1. In the context of an APT attack it is difficult to track, where the exfiltrated sensitive data is being uploaded to, thus making it an effective method for exfiltrating data.

Avoid detection through network packet manipulation One of the latest discovered APT campaign is called "Snake" [44]. Snake is an APT that can cleverly hide its network communication through implementing a kernel mode NDIS protocol driver. It allows itself to sit close next to the network adapter as soon as the network communication enters the computer. Inbound connections are performed as follows: Before it reaches the destination user-mode program it already filtered the network packets from reaching its destination. For Outbound connections the user-mode Trojan inject itself through use of dynamic link library into a legitimate program, such as an internet browser, which already got approved by the local firewall application. This results in bypassing all HIDS and firewall applications.

Logic Bomb A logic bomb is defined as a execution date for a designated malware when it is being executed. This implies that after infection the malware will check the current date periodically, which could be once a day or every certain hour in order to avoid suspicion. Most online automated malware scanners are being circumvented this way. The malware usually stays dormant during this time. The more advanced malware may apply this tactic in order to avoid detection by authorities.

1.3 How can APT attacks be prevented?

It is difficult to prevent an APT attack, since there is no silver-bullet solution to mitigate the threat. Many researchers believe that an APT attack can only be partially prevented if responded in time[7, 9, 31, 45]. In 2010 MTrends report [46] stated that only 24% of all anti-virus products could detect APT malware out of 76% that remained undetected. Prevention and detection is really a necessity according to 2013 MTrends report[1], where perpetrators were able to remain 243 days median active on an infiltrated network. So the need for defense measures against APT attacks is large at hand.

1.3.1 Detection method for APT attacks

In this section we are going described traditional intrusion detection systems against more fine grained detection methods in order to track down an APT attack. For a company it's difficult to defend against the initial attack vector called "Spear-Phishing" see Section 1.2.1 as claimed by many researchers[8, 14, 31, 32, 34]. The best solution for detecting an APT attack would be through network- and host-based monitoring, accordingly to Mandiant[46].

1.3.1.1 Intrusion detection systems

Intrusion detection systems (IDS) are used for partially automatic overviewing a corporate network. IDS focus on a company's internal network. To be more specific there are mainly two types of IDS. These are Network IDS (NIDS) and Host-based IDS (HIDS). They well known for detecting common attacks such as port scans or registering large data transfers. Common known IDS are Snort or Dragon. These contain several techniques for detecting anomalies in a corporate network. These countermeasures are useful for detecting APT reconnaissance attempts, such as port scans. SQL injections are also detected with these systems. Although more advanced APT attacks do out-of-box reconnaissance tactics, such as scanning social networks for gathering information for social engineering attacks. The benefit of having HIDS installed on each client in a corporate network is that provide more fine-grained solution for detecting anomalies such as modification of critical system files. But one of these IDS are not enough to prevent an APT attack from occurring. But IDS would effectively detect an internal network port scan done by APT attack in the APT operation phase. "Stuxnet" see Section 1.1.2.3 is an APT attack, which uses internal port scan techniques in order to propagate itself over the network. Many researchers believe in order to protect against an APT attack multiple defense layer technique are required in order to protect against such threat[5, 6, 31, 45, 47]. A good start would be to combine a NIDS with a HIDS for creating a network defense layer. A down-side about NIDS is that it is signature based, which is similar how anti-virus applications are used to detect viruses in files. There have been some proposals of simplifying the false positives that a IDS generates[48]. The next section describes more in detail what types of IDS currently exist. The following two IDS are network based IDS.

Signature-Based Intrusion Detection Systems The attacker often leaves signatures or traces in the malware which can be detected through the network communication protocol. Analysis can be performed by signature-based IDS (SIDS), but only if the

malware is already known. A SIDS will be as effective as the network security manager manages his signatures database. Sometimes it's possible to detect slightly new versions of malware, but only because they use patterns of some known signatures. If a threat is detected an alert will be generated towards the network security manager. In case of an APT attack this conventional IDS will render itself useless against it, because APT attacks often use new types malware, which circumvent these signature-based IDS[5].

Anomaly Detection Systems Anomaly detection systems (ADS) are different from SIDS, because they don't apply signatures but instead use abnormal state patterns in network traffic. This implies that a network security manager creates a network state template, which shows stores normal legitimate network traffic patterns. With ADS they compare network characteristics, such as traffic size or packet size or which protocol used. The benefit of using anomaly detection systems is that new malware can be detected if they use large data transfers which were not originally in the template. MySQL Injection attempts are also easy to discover with ADS. The downside about it is that if a new malware uses covert channels [49], then it can hide in normal traffic without triggering the any anomaly. With an APT attack it is presumable that it will use the most advanced technique available in order to avoid detection, so covert channels are not excluded in an APT attack. But some new techniques in this area [50] have already been proposed to detect a potential APT attack based on network characteristics, which could improve anomaly based detection systems in the future.

Passive and reactive systems Passive systems would be systems like SIDS or ADS systems. They log any malicious activity defined in their templates. Usually when a log is generated it is send to the local network security manager of company for further inspection. Although this is effective for known attacks as previously explained, but if these systems are combined with reactive systems also known as intrusion prevention systems (IPS), then it has a more effective rate of stopping an attack. But the downside about an IPS is that if a template has not be carefully created then legitimate traffic also gets blocked and that creates performance issues and business continuity problems for the company. Often these two system are combined in a company to perform better security evaluation. These are called hybrid systems. Hybrid systems can also be combined with NIDS and HIDS. If an APT attack occurs the best suitable option would be not to use IPS, but hybrid passive systems[5]. So that under no circumstances legitimate traffic gets blocked, but passively monitored. A good example of a hybrid system is showed in [51].

Host based intrusion detection systems With host based intrusion detection systems (HIDS), a monitoring software get installed on a client computer in a corporate network. It will monitor critical systems files for any malicious changes and logs it locally on the client system. Not only will a HIDS monitor critical system files, but also regular log files used by regular applications such as http server applications or ftp applications will be monitored. OSSEC is such an applications which performs these actions[10] They tend to work together with a NIDS by sending the log files to NIDS server for a more controlled monitoring environment. Dragon[11] is such a system that makes used of NIDS and HIDS systems. Some commercial firewalls such as McAfee or Norton claim to have HIDS. But they make use of a broader system, which collects on a large scale network behavior patterns[12] called GTI-technology.

1.3.1.2 Statistical correlation techniques

Some researchers believe that through statistical correlation techniques APT attack can be predicted[15, 50] and in doing so early awareness could result in effective prevention capabilities. A better approach would be if statistical correlation techniques were used in combination with several different IDS systems, such presented in [5]. The following subsections describe different methods for tracking or anticipating APT attack phases.

Counter method for Drive-by-download Marco Balduzzi et al from Trend Micro research team[15] have discovered a novel system for detecting similar malicious URL's. Through using extensive threat data set which consists out of detected malicious URL's from client and server machines. They have collected these samples from all their customers worldwide. "Spunge" as they call their novel system uses a clustering algorithm that allows to identify groups of malicious URL's that share similar hostnames or requests. By calculating the distance in the hostname variance. For example: cr5aiglist.com and craigsli8st.com. If a new domain name would appear has character in a different location, then their prototype system would detect it. This is done by clustering techniques such as k-means[13] and x-means[14], which is used in order to detect Domain Generation Algorithm-based (DGA) Malware [42]. This prediction technique could be used to anticipate further malicious URLS which can then be combined with an internal DNS sinkhole[52] server to avoid visiting the malicious URLS. A DNS Sinkhole server can be used for employees in a designated company for avoiding infection.

[10] http://www.ossec.net/
[11] http://www.intrusion-detection-system-group.co.uk/dragon.htm
[12] http://www.mcafee.com/us/threat-center/technology/global-threat-intelligence-technology.aspx
[13] http://www.cs.uvm.edu/~{}xwu/kdd/Slides/Kmeans-ICDM06.pdf
[14] http://www.cs.cmu.edu/~{}dpelleg/download/xmeans.pdf

Combined with the free publicly available Blacklists (see Section 1.3.1.3) it provides an extra security barrier against perpetrators.

Tracking down Trojans The researchers from [50] designed a prototype tool that can detect Trojans based on the software network behavior on client machines on Windows. They modified the service provider interface in User-Mode in order to retrieve network communication information in windows. They used a fine-grained classifier based on Decision Tree and Naïve Bayes model. Their detection model is based on the following characteristics:

- Ratio of send and received traffic size
- Number of connections
- Proportion of upload connection
- Proportion of concurrent connection
- Number of distinct IP

They also claim that an APT attack consists out of very new versions of Trojans, which are being used to infect target machines. This detection method is useful during the APT operation phase. The only downside about this detection method is that it uses user-mode level monitoring on a client computer. Which would not detect a kernel-based rootkit, because it would hide its network connections from user-mode applications, like "Stuxnet" does.

Multiple Sensors Scanners With multiple sensors scanners it is possible to detect an APT attack in the early stages. Examples of these methods are explained in [5, 45]. Multiple sensors scanners implies that it uses multiple tools that target specific areas of a network security layer in order to provide protection, such tools could be Snort or Dragon for protecting the network layer. But the small scripts presented in [5] can help in order to detect fast-flux traffic. Another method that utilize multiple sensors scanners is a Process Query System described in [45]. As explained in [6] that an APT attack can only be overcome by implementing multiple security layers in a corporate network. This would only provide partial protection against the first and second APT phase see Section 1.1.1 .

1.3.1.3 Defend techniques against APT

For an APT attack there exists several defense techniques, but as explained previously (see Section 1.3) there exists is no single solution to prevent all phases of an APT attack from happening. In order to mitigate the APT several proposals have been done by other researchers [24, 53, 54] to prevail at least in some of the APT phases.

Email inspection One way to address the APT delivery phase is through email attachment inspection. According to Trend Micro[34] (see Section 1.2.1) 94% use a malicious email attachments for propagating into further APT phases. This indicates that providing protection mechanisms at this level would be efficient. Some researchers have already proposed a detection method for identifying malicious pdf attachment, where malicious JavaScript resides in[24]. Their approach consists of applying a PDF fingerprint for grouping similar PDF files that contain malicious code. They extract the JavaScript code and apply a tokenize feature on it in order to disable the perpetrator obfuscation techniques. The downside about this approach is that it is only applicable to PDF files. Accordingly to [14] only 8% of PDF type attachments are used for spear-phishing, hence since most of the APT attacks that have occurred recently all used spear-phishing techniques to propagate. Therefor an email filtering system proposed in [55] is a more efficient approach for inspecting email. It focusses on email attachment characteristics such as sender and receiver email address and also includes a timestamp for creating statistical framework. They apply would apply correlation techniques like if multiple recipients receive the same email or when the same sender is detected within an impossible time constrain for humans to process emails.

Protection by an DNS-based Black hole List (DNSBL) Originally a DNS Blacklist (DNSBL, aka DNS Black hole list) was created for blocking spam email. But recently it gotten more attention, because it is also useful for blocking bots which are part of a botnet. If bot is discovered by sending excessive amount spam emails or is captured by a honeypot (see Section 1.3.2) it is added to a DNSBL for preventing further malicious activities. If this is combined with an internal DNS Sinkhole server it can provide an extra defense barrier against APT attacks. But the benefits of using a DNSBL also have a downside, because perpetrators can also use these DNSBL in order to replace defective bots inside a botnet. The researchers in [53] have proposed query graph based method, which allows to return false information whenever a botnet herder tries to access these DNSBLs. They claim to provide fake query responses from these DNSBLs back to botnet herder. Whenever a botnet herder request an DNSBL update on one of their bots to check if they have been blacklisted it will return the opposite

answer. If the DNSBL doesn't contain a record of the designated bot it will return a valid response that it is indeed listed on the DNSBL, which results in letting the botnet herder think it's on the DNSBL and will replace the bot inside the botnet. One other scenario would be that the botnet herder will issue an request of an bot that is in fact listed on the DNSBL, but he will receive a response that it's not listed on the DNSBL, which ultimately results in fooling the botnet herder in believing that his bot has not been blacklisted. These researchers have only tested it so far on a test dataset. It is very probable that the perpetrator will eventually get a hold of this illusion, but it will still effectively confuse the perpetrator. A DNSBL combined with a DNS Sinkhole can provide effective protection against APT initial attack vectors, such as Long lining and Drive-by-downloads. This technique would reduce chances of propagating an APT delivery phase.

Sandboxing Applications A Sandbox is a concept that is familiar in the security community. It is being used for reverse engineering purposes and to catch malware in honeypot environments (see Section 1.3.2). A Sandbox is an environment[54] where in theory no harm can be done by any kind of malware. This is done by providing restrictions on network layer and execution environment. In the execution environment the application is limited to a specific boundary inside the memory. This way no buffer overflow can occur in order to escalate privileges. Still there is no 100% guarantee that malware can escape these environments. Often their weakness lies in how the sandbox is implemented on several platforms. An effective method for preventing initial infection through Drive-by-downloads from propagating is by sandboxing the web browser. The challenge lies in the sandbox application not to get infected by newer types of malware. In the context of an APT attack a sandbox can prevent the Delivery phase from further escalating.

1.3.2 Tracking Methods

Tracking methods are sometimes needed to remediate the damage done by an APT attack in order to perform some prosecutions. Most of the time it tends to be difficult, since a perpetrator uses sophisticated methods in order to avoid prosecutions. The following subsections describe some tracking methods for authorities in order to track down the perpetrator or preventing similar infections in the future.

Fast-flux tracing Fast-flux tracing is possible due to collecting large amount of DNS and IP records and mapping then in one big architectural overview. Some researchers have implemented a prototype which collected these records on a large scale[16]. They

claim to have an effective approach on fast-flux networks. In their research they showed that their prototype tool could distinguish several different botnet types, such as phishing botnets or malware botnets. From their results they noticed that an average botnet has a alive rate of approximately 2 months. They presented a website[15], where a user can submit IP address or DNS in order to determine if it belongs to a fast-flux botnet. Due to unknown reasons the website permanently offline, but an old web cache version can still be visited at[16]. Some other researchers provided a real-time detection method for detecting fast-flux webservers[56]. Their approach consists of measuring 3 types of delays in HTTP Get request in order to determine if the webserver is using a malicious fast-flux host. These are Network, Processing and document fetch delays.

Utilizing honeypots Honeypots are known to the security community for capturing malicious traffic or real-time hackers on duty. A honeypot is an environment which is similar to a Sandbox, but with additional capabilities. Besides the regular security characteristics shown in Section 1.3.1.3 it also employs mimicking a real environment. There are in general 2 types of honeypots, which are low and high interaction honeypots. A low-interaction honeypot can be seen as mimicking a single service, which could be for example an SSH service. One real example is "Kippo"[17]. It creates a real SSH shell environment, where a perpetrator can be caught if he tries to log into the honeypot. Its functionalities are logging all shell commands. This can be useful for whenever the perpetrator tries to download a malicious file for infection the target machine. In turn a security consultant can take this malicious sample for further investigation, such as reverse engineering explained in the next chapter. A high-interaction honeypot would be virtual operating system software, such as VMware or VirtualBox. It can clone an entire operating system in an virtual environment. This could be useful for whenever an malware has been infected on the virtual host. Therefor all aspects of the malware can be analyzed by forensics, such as network communication or file system changes. The researchers in [57] have developed an closed-source prototype which is capable of dynamically creating low and high interaction honeypots for forensics. The only large dependency is that it relies on corporate Security Information Event Manager (SIEM), which would be necessary to be installed on the network. A SIEM is also known as IDS and IPS systems. If this prototype would be effectively combined with a sophisticated IDS system, then it can be of high value for authorities to reverse engineering incoming malicious malware, which would be caught by the honeypot. For an APT attack it is important to trace its origin for prosecutions.

[15] http://www.fastfluxmonitor.com
[16] https://web.archive.org/web/20121031130129/http://www.fastfluxmonitor.com/
[17] https://code.google.com/p/kippo/

Reverse engineering malware samples Reverse engineering proves to be quite difficult in practice, especially because most of the time it is done manually. In case of an APT attack, the perpetrator will use the latest malware samples on the market in order to perform their malicious actions. As shown in [9] the research that was conducted, included some reverse engineering samples of an APT attack that was performed in a Hong Kong, that targeted some high-raking political persons. With reverse engineering of the malware samples they discovered how the APT attack was performed. Reverse engineering is a process that is most of the time combined with honeypots mentioned in the previous section, in order to determine the overall picture how an APT attack was structured. Another research was conducted on automated malware analysis in [58]. It uses a prototype called FARM, which is a modification of Cuckoo box[18]. It was designed for reverse engineers to simplify the reverse engineering process in order to save time and effort for authorities to track down APT attacks and other malicious malware.

1.4 Tackling the APT in the operation phase

The problem of APT is that we have seen that stealthiness is a main goal of APTs. In this paper we focus on is the operation phase after exploitation has been performed. The history of APT attacks show us that (see Section 1.1.2 and 1.1.1) most of the time it has been a core feature to deploy a Rootkit and Trojan for persistence, which will enable the APT attack to become invisible and active for a prolonged time in a corporate network. The following hypothesizes can be derived from this context:

a) We have seen that stealth capabability is a main goal of APTs.

b) APT needs to communicate in order to fulfill its purpose.

c) To satisfy a) and b), APTs will try to hide the network traffic it generates from inspection by the infected computer and other computers within the network or network tracing devices which is by definition not possible to 100%. Encryption or steganographic hiding of data in legitimate traffic may be an option.

For our main research question we ask ourselves: "Can this difference between internal and external view be used for detection of APTs and how can this be accomplished?"

In order to achieve this, we need to investigate more on the network traffic hiding capabilities of APT. More details on this in Section 1.2.3. Next, we need to identify how

[18] http://cuckoosandbox.org/

we can measure the difference between internal view and external view reliably. Finally, we need to investigate how reliable this is and whether there can be false positives due to regular activities in an OS.

Therefor the following research questions can be defined:

1. How can a APT malware be detected when communicating to the C&C server in the operation phase?
2. Is it possible to identify a recent APT malware on an infected system?
3. What possible evasive countermeasures can be performed by an future APT malware sample in order to avoid detection in a virtual network environment?

The following architecture is proposed to address these research questions consists out of 2 computers and 1 router, whereas the "OpenWRT" will be the router in this scheme and the "Openflow Controller" will operate as linux based controller. One client will be a windows system which hosts an APT malware sample. The idea is that through correlating network traffic and fingerprinting it, we could detect newer APT malware samples which contain similar characteristics of previous APT malware, when hiding their network communication. This idea could be a possible solution to detect newer versions of APT malware.

This solution can tackle two APT attack problems. The first attack would be if the an APT malware sample tries to avoid the genuine network socket API of windows in order to hide itself. The Second attack would be if an APT malware sample tries to modify legitimate network packets in order to communicate through legitimate channels.

Chapter 2

Forcing communication with C&C servers

In order to evaluate APT malware communication it is required that it executes under the right circumstances. To do this we must avoid detection traps set up by the malware developers. These detection traps can vary from anti-debugging tricks or checking system parameters. The following sub chapters will explain these techniques.

2.1 APT Counter measures for sandboxed environments

Many countermeasures for detecting sand-boxed environments exist today. Here are a couple of examples that are most common for detecting a sandboxed environment in popular virtual environments.

VMware VMware uses the magic number 'VMXh' which allows to communicate with the host machine, for interactive handling, such as copy-paste data or accessing the VMware tools. The trick here is that the instruction: "in eax, dx" is a privileged execution. If VMware is installed then is allowed to execute this instruction, but if VMware is not present then the system will crash since it is not authorized by the system to execute this instruction in user mode.

```
mov eax, 'VMXh'   ← magic number
mov ebx, 0
mov ecx, 0Ah      ← set function number / 0Ah = get VMware version
mov edx, 5658h    ← port number number used to communicate with VMware
in eax, dx        ← read a dword from that port
cmp eax, ebx      ← if VMware is present EAX == EBX
je __VMware_detected
```

FIGURE 2.1: Detection VMware

VirtualPC For VirtualPC there exists a special set of call instructions that allows communications with the host system. On a normal PC without the virtual environment the instruction "db 0Fh, 3Fh, 7, 0Bh" will result in an illegal instruction exception.

```
mov ebx, 0
mov eax, 1
db 0Fh, 3Fh, 7, 0Bh   // VPC Call
cmp ebx, 0
je __VPC_detected
```

FIGURE 2.2: Detection VirtualPC

VirtualBox For VirtualBox there exists a slightly different way then the previous ones, through finding a window handle. Since in VirtualBox there exists an tray icon, it is possible to detect this through the windows api call 'FindWindowA'. If it would receive a valid handle then it was able to detect VirtualBox.

```
push 0
push 'VBoxTrayToolWndclass'
call FindWindowA
test eax,eax
jnz _VboxDetected ; if we managed to obtain a valid window handle VBox was
detected
```

FIGURE 2.3: Detection VirtualBox

These "flaws" are called virtual anomalies that exist when working with virtual hardware. According to the T. Garfinkel et al[59]. they can be classified in several different categories. The "VMware" and "VirtualBox" figures shown in this chapter are examples of logical discrepancies of the CPU. Besides that APT malware can check on these CPU discrepancies, Off-chip discrepancies are also possible. Here we can think of the static hardware descriptions labels of the designated virtual environment provider(VEP) drivers. Of course a virtual environment provider like e.g. VMware can implement a richer set of emulated devices, but this would only increase the amount of work, while it has no benefit to the VEP customers. The research performed in [60] displayed in

form of an presentation shows that there are 4 categories of methods in order to detect a virtual machine environment (VME). These methods are shown as follows:

1. Look for VME artifacts in processes, file system, and/or registry.
2. Look for VME artifacts in memory.
3. Look for VME-specific virtual hardware.
4. Look for VME-specific processor instructions and capabilities.

Figure 2.3 example is coherent with the first category in the method list. This explains that finding handles to known VEP support programs in the virtual environment. For the second method the research in [60] shows examples of VME artifacts that can be found in memory by performing a simple string search looking for e.g. "VMware" , where they found more than 1500 string references. The third method is similar to the Off-chip discrepancies shown in [59]. Likewise the fourth method is finding anomalies in CPU instruction set similar to CPU discrepancies also found in [59]. Another anomaly would be the timing discrepancies[59]. Virtual and physical environments have a difference in timing. These can be classified into two categories:

- Local time sources: These timings can be measured on the device itself. Where it is possible to deploy race conditions in order to detect differences in how threads execute by measuring their time variance of execution.
- Remote time sources: These timings imply that it is possible to measure the transfer of a network packet from the outside world towards the virtual environment. Here a similar technique can be applied to calculate the variant time of transfer in order to detect the virtual environment.

In case of an sophisticated APT malware sample only one of these virtual anomalies would suffice in order to detection of an virtual environment.

According to the research in [59] it is possible to prevent local time sources variant calculations through applying "time dilation". The effect will that the overall execution of a virtual system will be slowed down in CPU cycles in order to match the physical system. The downside about this approach is that it would increase performance significantly in some cases[59]. This does not prevent remote time sources calculations. Hopefully as mentioned in the research paper [59] the possible trend for malware (including APT malware) would be that these possible detection methods would be rendered useless, because more and more enterprises are deploying virtual environments for their business servers.

2.2 APT malware samples

This chapter provides a small introduction on APT campaigns. The following APT malware samples will be used in the upcoming experiment. These APT malware were chosen for further investigation. There description will be as follows:

Careto According to Kaspersky [61] this APT malware targets multiple high-valued targets such as: Government institutions, Diplomatic / embassies, Energy, oil and gas, Private companies, Research institutions, Private equity firms and Activists. The APT malware has got its name from string reference inside the malware itself. Kaspersky presumes that this APT campaign is sponsored by a nation state, although they concluded that the sample was written in the Spanish language, this does not necessary mean that it was funded by a Spanish state. Often malware developers design their malware samples in such a way that they leave traces behind in order to confuse or falsify their origin. Furthermore they conclude that this malware sample is very sophisticated in a way that it contains a bootkit, rootkit, 32/64 bit versions, MAC/OS and Linux versions. The md5 hash used in the malware experiment is as follows: 5cfd31b1573461a381f5bffa49ea1ed6

IXESHE The IXESHE APT campaign primarily targets east Asian companies. The initial attack vector they used emails containing pdf or xls document exploits in order to gain access to their corporate network. TrendMicro [62] discovered that the attackers used over 60 C&C servers where most of them were compromised machines. Out of these C&C servers most of them were US and China servers. The attackers tactic was compromising a corporate network and install a C&C server in the network. From where all the victims within the network communicated internally with the C&C server in order to avoid suspicious activity on the external side of the corporate network firewall. Although TrendMicro does not mention if the APT malware hides its network activity, the upcoming experiment will reveal whether it performs hiding attempts or not. The md5 hash used in the malware experiment is as follows: 39822adc9bc7747dadd212e0338948cb

Lurid The Lurid APT campaign targets primarily Russian based countries, such as: Russia, Kazakhstan and Ukraine. The initial attack vector is similar to the IXESHE campaign and also utilizes a pdf exploits in order to gain foothold inside a target corporate network. TrendMicro was able to identify a C&C network that consisted out of 15 domain names and 10 IP addresses [63]. They were also able to identify certain campaign codes through reverse engineering C&C servers and communication

protocols. This analysis showed 301 campaign codes were discovered implying 301 attack were performed. The md5 hash used in the malware experiment is as follows: e4683f7480bb985f39ca72a7bc43c921

Mirage The Mirage campaign has targeted various high-value companies in various countries in the world. High-value companies such as oil-industries or military organizations. The different aspect from other APT campaigns is that it targets countries that do not exists in close proximity from each other as in the same continent. Although the majority of infection came from Taiwan and the Philippines. Dell SecureWorks [64] have identified the Mirage campaign and decrypted its C&C communication protocols. One relevant aspect of note worth mentioning is that a Mirage APT malware sample will try to communicate with the C&C server indefinitely if it cannot establish a successful connection. This aspect can be useful in the upcoming experiment. Furthermore it uses similar attack vectors to other APT campaigns, such as pdf exploit for initial infection. The md5 hash used in the malware experiment is as follows: abac650ab39c0dd074310710081d715d

Snake The Snake campaign is according to GData [65] one of the sophisticated APT campaign up until now. Snake APT is similar to Careto, when it comes down to 32-bit 64-bit versions of the APT malware and the complexity of the framework. Apparently GData discovered that this APT campaign has been undetected for the past 3 years, which makes this APT campaign very successful over the past years. GData suspects that the Snake APT has Russian roots, because of the Russian language embedded within. The initial infection method is still unknown occordingly to GData and BAE Systems[65, 66]. A key feature of Snake is that it is able to bypass the 64-bit patch guard protection system from Windows 7 system shown by the researchers in [67]. Snake uses an encrypted virtual file system (VFS) in order to hide from anti-virus and anti-rootkit applications. In the VFS third-party tools are being used in order to exploit the targeted network further. Examples of such tools are information gathering tools and dump tools for NTLM hashes in order to perform "pass-the-hash" exploits. The VFS is also hidden through the use of Alternate Data Streams (see Section 1.2.2.2) from within the NTFS file system. Another interesting fact is that Snake can reach out to other infected computers that are not connected to the internet through named-pipes. BAE Systems [66] and Gdata both believe due to the nature of this complex framework a government sponsored entity is behind this APT campaign.

Sykipot Sykipot is an APT campaign that has been around for 2006 according to a report from SANS Institute [68]. It has also similar exploitation techniques for initial

infection as other APT campaigns, through email attachments. In 2012 SANS Institute discovered that there was a smartcard variant that allowed spreading through smartcards. It has several anti-reverse engineering techniques such as covertly masking itself as an Microsoft Corporation file or a delayed dial-in for the login towards the C&C server. What is interesting with this specific piece of APT malware is that it includes a command in the dropper exe. If the command "-removekys" is executed on the dropper exe, then it will remove itself from an infected system. It is capable of injecting a hook dll into firefox application. Furthermore it uses a custom DES encryption scheme in order to encrypt the C&C server commands. The communication ports that are used for communicating towards the C&C server are default ports for secure and unsecure browsing, which are 80 and 443.

Taidoor The Taidoor APT campaign has been around 2008 according to Symantec [69], but Trend Micro [70] claims that it saw the earliest sample in October 2010. Although the attackers behind the Taidoor campaign do not used zero-day exploits in the APT malware, instead they rely on old target systems, which are not patched. Accordingly to Symantec [69] the attackers primarily target companies in East Asian countries. Pdf exploits were mostly used for initial email infection. The emails where send mostly from compromised Taiwan email servers. In order to hide from authorities, the attackers would hack third-party servers in order to relay their C&C communications to their real C&C server. The attackers utilize a custom RC4 encryption in order to encrypt their C&C communication and the encryption would also encrypt the data inside the dropper executable.

Winnti Winnti APT campaign was named after Symantec and Kaspersky took the same name for this campaign [71]. Similar campaign existed as well, but under a slightly different name called: "ETSO"[72]. The Winnti campaign is different from other APT campaigns mentioned in the previous sections. Winnti focused primarily on the online gaming industry. Since in today's setting more and more gamers are willing to pay for online upgrade where they need to pay for. The actors behind Winnti were able to infiltrate several online gaming companies and infected them. By accident the APT malware infected the game update server and the users were infected as well. This is how Winnti got discovered, because the users complained about infected update files and became suspicious about the intentions of the designated online gaming company, but the truth was that the online gaming company network got infected. Surprisingly Winnti used a stolen legitimate certificate in order to remain undetected. 32-bit and 64-bit were also discovered. According to Kaspersky's research Winnti tries to hide its

network connections from an infected system. Some tactics were also deployed by the attackers to confuse anti-malware researchers if they were only looking at functions calls.

2.3 Triggering multiple APT malware samples to communicate to C&C servers through custom sandboxes

In this chapter we try and provoke the APT malware sample to communicate to its C&C server. This is done by creating two control groups in the form of sandboxes. Once the designated APT malware sample is being executed the external side of the network will be closely monitored and compared to a list of known C&C servers IP addresses en DNS names listed in [61–72]. A final summary of this C&C Address list can be found in Appendix B and C. By utilizing multiple sandbox environments we try to discover how the malware behaves. Several sandboxes already exist in the wild and here we will apply some of them in order to gain more insight into the APT malware samples. The most common known high-interaction honeypots also known as sandboxes, which are e.g. VMware and Virtual Box. Before we can trigger the APT malware samples to communicate with the C&C server, sandbox environments must first be set up. Primarily two sandbox environments will be set up. One VMware version, which contains several anti-sandbox detection tricks as described in [60]. The second version will be in Virtual Box which does not contain any prevention measures for detecting an sandbox environment. The following configuration file for VMware will be used in shown below to avoid detection by APT malware, these were taken from [60]. Then the external view of the network will be monitored by Wireshark (see section 2.4).

VMware Configuration Options – used in the guest's .vmx file

- isolation.tools.getPtrLocation.disable = "TRUE"
- isolation.tools.setPtrLocation.disable = "TRUE"
- isolation.tools.setVersion.disable = "TRUE"
- isolation.tools.getVersion.disable = "TRUE"
- monitor_control.disable_directexec = "TRUE"
- monitor_control.disable_chksimd = "TRUE"
- monitor_control.disable_ntreloc = "TRUE"
- monitor_control.disable_selfmod = "TRUE"
- monitor_control.disable_reloc = "TRUE"
- monitor_control.disable_btinout = "TRUE"
- monitor_control.disable_btmemspace = "TRUE"
- monitor_control.disable_btpriv = "TRUE"

- monitor_control.disable_btseg = "TRUE"

2.4 APT malware samples that communicate to C&C servers

In order to discover APT malware traffic, the sandbox environments in the previous section will be used to detect traffic through the use of the application "Wireshark"[1]. The "Wireshark" experiment consists out of sniffing network packets from the external and internal side of the designated infected client system. Internal in this context means that Wireshark is running inside the infected client system. Where the external side is referred in this context through deploying a Wireshark application on a Ubuntu Linux system, which acts as a server. The network architecture is visualized in Figure 2.4. Wireshark utilizes WinPcap[2] in order to operate the packet filtering procedure. This is done through the kernel mode driver which is called the Netgroup Packet Filter(NPF). [3]

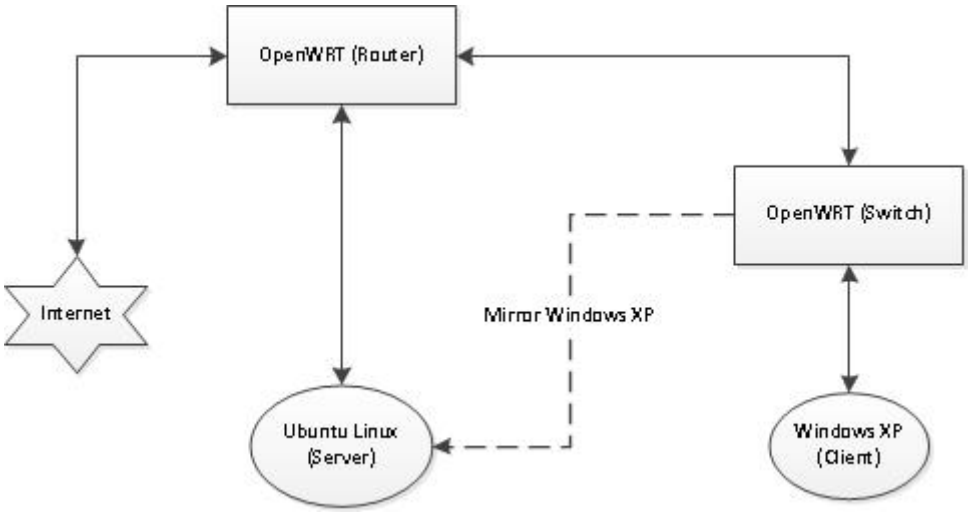


FIGURE 2.4: Network Architecture Experiment

After performing the "Wireshark" experiment the following results came forth. The VMware results were omitted, since it yielded no results considering that the results were distinct when it was compared to VirtualBox.

[1] https://www.wireshark.org
[2] http://www.winpcap.org
[3] http://www.winpcap.org/docs/docs_412/html/group__NPF.html

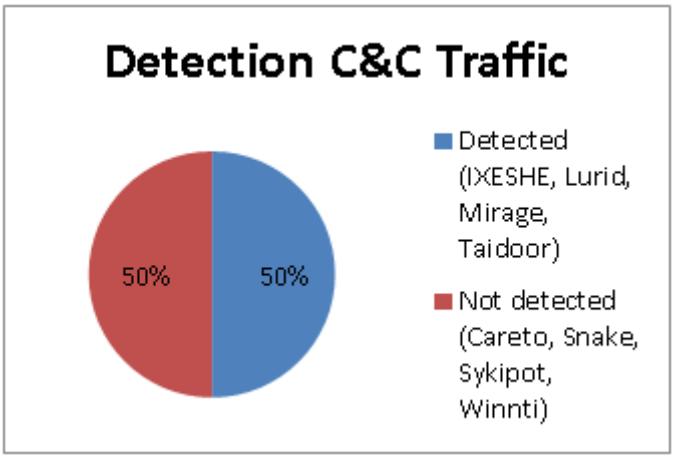


FIGURE 2.5: Results Wireshark Experiment

Multiple reasons may be the possible cause for not interacting with a C&C server, for example a logic bomb (see section 1.2.3) which only executes at a specific date and time or when other specific conditions have met, such as environmental values which refer to specific system characteristics like e.g. driver names or registry values. Due to time constrain it was not possible to reverse engineer the malware samples in order to analyze the behavior and to force the samples to communicate with the C&C servers.

Chapter 3

Cross-referencing idea

The idea of cross-referencing communication data came originally during the preliminary research effort in Chapter 1. Kaspersky discovered in the past that Winnti [71] malware performed hiding efforts in order to prevent itself from being discovered through the command "netstat" in windows, which displays current active network connections in Microsoft windows.

During the investigation of the more recent APT malware variant of Turla called "Snake" [65, 66] came to light that this particular malware created its own network drivers in windows in order to hide its network traffic. The Snake malware was able to filter and inject network traffic in the OSI model, before it reached either the application or physical layer. This behavior created the following hypothesis:

"Can the difference between internal and external view be used for detection of APTs and how can this be accomplished?".

To further explain this hypothesis, internal view is considered monitoring the network traffic from inside the infected machine. The outside view is realized through deploying a monitoring host within the same network. The concept is visualized in the XCOM Network Architecture Sketch model shown in Figure 3.1.

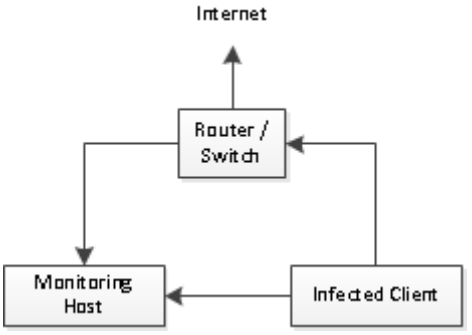


FIGURE 3.1: XCOM Network Architecture Sketch

By comparing the two outgoing network streams we can verify if any modification had taken place or if packets were filtered out. This includes packet injection as well. The following diagram displays this concept more clearly.

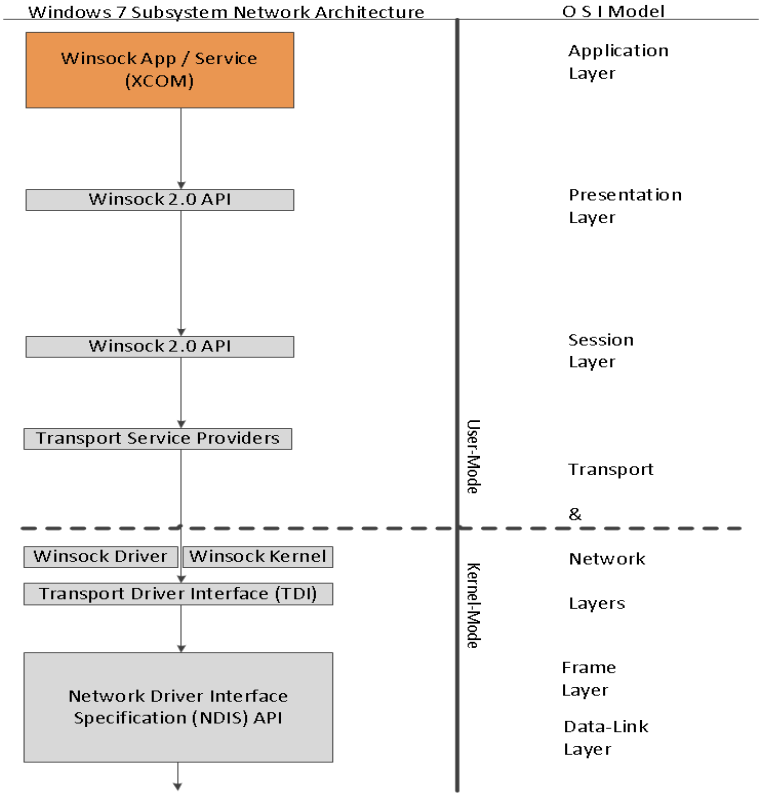


FIGURE 3.2: Windows 7 Network Subsystem Architecture

The idea behind the XCOM experiment is that through communicating from user-mode level it is possible reveal APT malware through comparing the traffic patterns with the monitoring host. Through this method we can detect packet modification attempts by APT malware. If the following condition is met, then XCOM can detect the APT malware.

- If any APT malware tries to modify, filter or inject network traffic data between Presentation layer and Data-Link Layer of OSI model outside the original Windows network stack.

The hypothesis will be proven in the XCOM experiment. The goal eventually is to detect APT C&C network traffic in a reliable way. If C&C traffic can be detected, then other third-party utilities can provide a more specific direction of where the network communication is originating from.

Chapter 4

The XCOM experiment

The XCOM experiment was created in order to prove the hypothesis in chapter 3. XCOM stands for cross-referencing communications. In the Wireshark experiment we have seen that 50% out of the 8 samples try to communicate with the C&C servers in chapter 2. The remaining 4 APT malware samples will used for the XCOM experiment. This is due to the fact that the XCOM experiment tries to detect C&C communication traffic. In the event that these 4 APT malware samples do not deem enough, then more samples will be provided in order to achieve a more reliable outcome. The XCOM experiment uses the same network architecture as the Wireshark experiment. The pre-condition for the XCOM experiment is that APT malware must communicate towards the C&C server in order for XCOM detection tool to detect it. The following paragraph will explain the structure of this experiment.

Structure The XCOM prototype is divided in two applications. One for the server side and one for the client side. The server and client side both use a packet sniffer implementation. The XCOM server application will be used in order to compare network traffic in the role of the monitoring host. The XCOM client application will be deployed on the machine that has been infected by the APT malware. The XCOM experiment was deployed using a virtual environment designed in VirtualBox. This was possible through emulating some custom router firmware called: "OpenWRT" more information can be found in the next paragraph. One router and one switch was emulated this way in order to create the target network architecture required for the XCOM project. Although it is not necessary to deploy this virtual network environment in a real situation like a corporate network, in order to execute the XCOM experiment. If the XCOM experiment were to be deployed in a corporate network for network surveillance purposes then the requirement for the XCOM experiment would be to obtain a switch with a port mirroring

feature with a capable TCP dump filter for performance reasons. These are explained further in chapter 6.

OpenWRT OpenWRT is a custom firmware for routers. Most today's off-the-shelve routers can be modified with OpenWRT. The compatibility of OpenWRT on these routers can be viewed on their website[1]. OpenWRT was designed to create a single static firmware for Linux embedded devices. Therefor it is easy to emulate routers and switches in Virtual Environments like e.g. VirtualBox etc. through this custom firmware. OpenWRT offers a vast amount of software packages that can be installed after the custom firmware has been deployed. Including various user interfaces, which allows to perform interesting functionalities like e.g. "bandwidth limiting for clients" even on router devices that did not originally had this capability. In case of the XCOM structure it was required to perform port mirroring, which was included in one of the many software packages that came with OpenWRT.

Server The server application is written in Java en the underlying operating system is Ubuntu 32-bit. JPcap[2] will be used for the packet sniffer implementation on the server side. It was designed by Patrick Charles for capturing network packets in Java. It is capable of visualizing captured network packets in a target network. Developers can utilize this API to design several different forms of packet capture applications. The current state of the XCOM server application is through manual observation in the form window, it is possible to detect anomalies through comparing the two outgoing data streams. The detection is performed by comparing MD5 hash of each outgoing packet that originates from the infected client machine. These were filtered automatically through the program. In future development it can be easily automated towards writing the anomalies away in a log file. For more details on the comparison of the two outgoing data streams please look at Figure 4.1. Initially the traffic path starts at the infected system. The blue line represents the XCOM client application communication path, which contains the original outbound traffic of the infected machine. The red line represents the normal communication path which is being mirrored at the switch. Normal traffic which goes towards the internet is being omitted in the XCOM communication diagram. At the switch there is a TCP dump filter which is configured to pass on all outbound communicate to the monitoring host and neglecting all local network and inbound traffic in order to avoid exponential growth in network traffic.

[1] http://wiki.openwrt.org/toh/start
[2] http://jpcap.sourceforge.net/

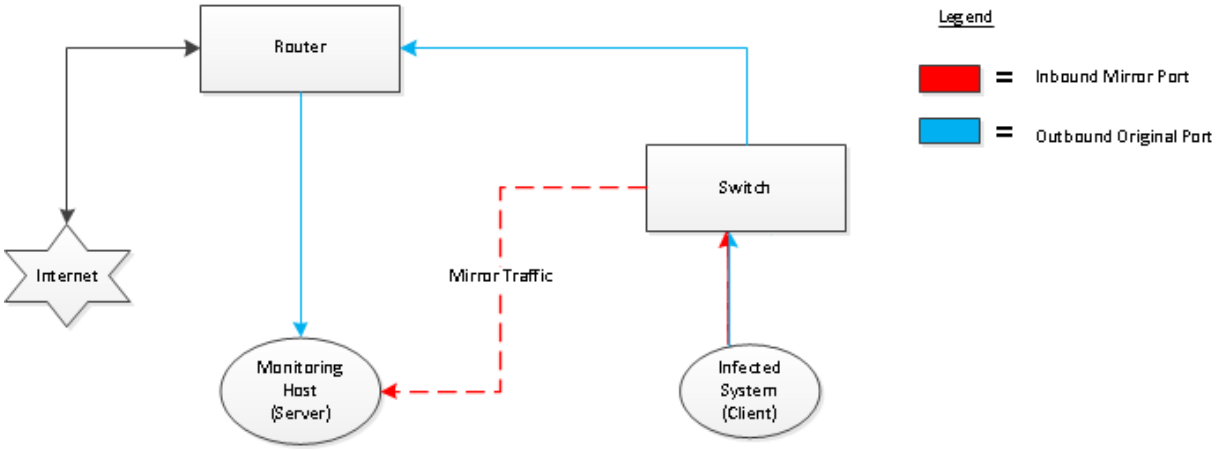


FIGURE 4.1: XCOM Communication Process

Client The client side however will not use any third-party tools, instead it will rely on the windows socket 2 API, which comes default with windows. Any application build with the windows socket 2 API resides in the application/presentation layer of the OSI model shown in Figure 3.2. By collecting network traffic at the application layer from the OSI model we can intercept all modification performed through the lower levels, hence since we are able to verify the data-link layer as well. The current operation of the XCOM client prototype is to record all outbound network packets. All packets are being hashed by the current implemented MD5 hashing algorithm. Once a configurable buffer size has been reached by capturing network packets the whole content of the buffer is being transmitted towards the XCOM server application. This is performed by Apache thrift framework.

Apache Thrift framework Apache's Thrift framework [72] allows developers to establish a communication channel between multiple programming languages e.g. C++, C#, Python, Ruby, etc. through an intermediate cross referencing language in the form of interface file. Apache's thrift framework will automatically generate the corresponding code for the target program language defined in the thrift file. An interesting fact is that it has been developed by the Facebook company in order to create more compatibility between different services for their back-end servers.

Negative Result The results from the XCOM experiment where inconclusive. Therefor the experiment was unable to detect any malware that attempted to hide C&C network traffic. Here we can deduct the following possible reasons:

- The malware was able to detect the virtual environment

- The malware had a logic bomb embedded for communicating with the C&C
- The malware did not attempt to hide its network traffic on the infected machine.

21 APT malware samples were used in the XCOM experiment. From 6 out of 21 samples did not communicate with their C&C servers. The remaining 15 samples were detected using the following method. The initial 8 APT samples that were selected had an background literature reference. These can be reviewed back in Section 2.2. The other samples came from: "Mila Parkour". She is malware researcher from DeepEnd Research. She was kind enough to deliver the remaining APT samples. These did not came with their corresponding whitepaper, but instead a pcap file for referencing IP addresses from C&C servers which worked quite well, combined with looking up the destination IP addresses. The APT samples that communicated with the C&C servers were analyzed in order to check if these we hiding their network connections on the infected machine. This was not the case, none of the remaining APT samples were hiding their network connections on the infected machine. The XCOM prototype was designed for APT malware with a specific behavior, which was similar to the Snake malware behavior. In order to still prove that the XCOM prototype detection tool will be useful, a prototype malware will be shown in the next chapter. The XCOM experiment results can be seen in Appendix C. In order to clarify the verification and result procedure the following table will be established to create a more clearer overview. The scope of data scheme are the captured packets that reside into a data stream which is categorized by the destination IP address.

A = Inbound packets from the mirror port.
B = Outbound packets encapsulated through the XCOM program.
C = Missed inbound packets.
D = Missed outbound packets.

The table corresponds with the screen-shot taken of the XCOM detection tool result in Appendix D.

A	C
B	D

The following scenarios are possible outcomes for the XCOM experiment:

1. If all packets from A are equal to D, then it is a true positive.
2. If some packets from A are equal to D, then it is a false positive, due to packet loss.
3. If any packets arrive in C, then the TCP dump filter is incorrectly configured at the mirror switch, which is considered a false negative.
4. If C and D remain both equal to 0, then no match has occurred thus no correlation was detected. This is considered a true negative.

Only one packet correlation was detected through the XCOM experiment during the execution of the malware samples. This result was a false positive, considering that an entire TCP or UDP data stream of packets need to be detected in order to achieve a reliable result. The APT sample "Letsgo" had 1 packet loss from the entire data stream. Let assume the data stream was 100 packets long, using the schema we can determine if it was a false positive or not. In this case it would be scenario 2, because partial packets were matched.

A = 100 packets
B = 99 packets
C = 0 packets
D = 1 packet

The detection result for scenario 2 can be seen in Appendix E. The next chapter will demonstrate how we can achieve the ideal scenario 1.

Chapter 5

Prototype Malware MOCX

In order to prove that the XCOM detection prototype is sound from design perspective, a prototype malware named: "MOCX" will be created based on the characteristics of the recent Snake APT malware. The name came from mocking cross-referencing communication. The MOCX malware will be made publicly available alongside with the XCOM detection tool to prove that the MOCX malware has not been specifically been made to adapt to a positive outcome of this research. The goal of the prototype malware is to create an application which will not get easily detected by any conventional firewall solutions in order to prove the stealth capability. The prototype will be tested with 4 commercial firewall solutions on the market for Windows. These were selected randomly through voting sites of the most popular firewall solutions in the selection category of 2014[1][2]. Finally the prototype malware must be detected by the XCOM detection tool. The following paragraph will explain the structure of the prototype.

Structure The prototype malware will send a simple tcp/udp packet towards a fixed IP address location every 4 seconds. The content of the malware packet is irrelevant. If outgoing packets can be detected at the external border of the network without detection by a commercial host based firewall then the MOCX malware will be a success. Two versions of the MOCX malware will be created. The first variant malware will use a its own NDIS network driver simulated by WinPcap in Section (3.5). This is implemented in order to avoid detection by any user-mode host based firewall solutions, which typically reside in the transport layer and up in the OSI layer model.The second variant will utilize a API called WinDivert[3], which manipulates the the network stack of windows. Furthermore it is installed in to replace the existing windows indexer service, which

[1] http://soft4sharing.com/top-8-free-firewall-2014.html
[2] http://www.thewindowsclub.com/best-free-firewalls-windows
[3] http://reqrypt.org/windivert.html

supplies the functionality of indexing and searching of files inside windows operating system. The MOCX malware prototype contains similar network stealth characteristics, which is performed by deploying the NDIS network driver.

Commercial Firewall Solutions This section will provide a description on the free commercial firewall solutions that will be tested on the MOCX malware, furthermore the XCOM prototype detection tool result will be added with the list of the security applications along the way. The MOCX malware should ideally not be triggered by any firewall solutions products at the moment.

Comodo Internet Security Comodo internet security (CIS) provides a lot of features which does not exist in the other commercial firewall solutions presented in this list . CIS contains an anti-virus for malware pattern match scanning, which is rendered useless against new malware that is deployed on the market. Only after initial detection of the malware will their cloud based behavior analysis kick in. Furthermore they have a host based firewall solution and an auto sandbox technology which is disabled by default installation. The most significant mayor feature of CIS is their Host-based Intrusion Prevention System (HIPS). The HIPS installs a kernel–mode driver in order monitor any access to critical system files. The HIPS is able to detect the XCOM prototype, but the firewall feature is unable to detect the malware. A small adjustment can be made in order to circumvent the HIPS of CIS, see subsection ("Circumventing most Host-based Intrusion Prevention Systems") for more information. Because MOCX malware is able to replace an existing windows service it can effectively circumvent CIS, without even showing an alert message from either the firewall or the HIPS.

FortKnox Firewall The developers of FortKnox Firewall claim to have integrated intrusion prevention system (IPS), which was not activated or seen as the XCOM prototype malware was installed and activated. No firewall log was recorded, when the data transmission from the malware was sent towards the internet. No kernel-mode driver was installed for the IPS. Although considering the feature page on their website[4], they claim to have a lot of focus on defending browser functionalities, such as ActiveX, Cookies, Ad Blocking.

Outpost Security Suite Outpost Security Suite was able to block the MOCX malware, but not able to detect it. Probable due to their system guard protection[5]. The

[4] http://www.fortknox-firewall.com/
[5] http://www.agnitum.com/lp/outpost-7-secures-whitepaper.php

system guard protection mechanism shielded critical system files from modification attempts by scanning their digital signatures. Thus Outpost Security Suite was able to block MOCX malware which had not a valid digital signature. If the MOCX malware had a valid Microsoft digital signature on the binary of the malware file, then the likelihood of detecting the MOCX malware would decrease significantly.

ZoneAlarm Security Suite ZoneAlarm Security Suite was capable of blocking the communication of MOCX malware, but was not capable of detecting the MOCX malware. This could be due to their OSFirewall implementation [6], which states that it verifies critical system files, but it does not log any attempt made by the MOCX malware. The same probability of the digital signature scheme applies to the previous paragraph.

XCOM prototype detection tool The XCOM prototype detection tool was able to detect the WinPcap variant of MOCX malware even though it injected packets on link-layer level. This was due to that WinPcap driver uses its own TCP stack, when injecting packets. The XCOM application discovered a discrepancy between the inbound and outbound packets of the target client system, through comparing the two network streams. The detection result can be viewed in Appendix D. Although the WinDivert variant of the MOCX malware was not detected, because it used the same TCP stack as Windows Operating system.

Circumventing Host-based Intrusion Prevention Systems (HIPS) Through exploiting digital signatures most HIPS can be circumvented. This trick is performed by spoofing a digitally signed binary through hash collision or stealing a root certificate. Either method is capable of replicating a genuine Windows Service. A malicious proxy service can be created in order to redirect requests to the legitimate Windows Services in order to avoid detection. Since most anti-virus and commercial security suites do not scan for binary files, which were digitally signed by the Microsoft Corporation. In case of the MOCX malware an attempt was made for replacing a critical system file e.g. the Windows Indexer Service. But this was limited due to not being able to replicate the digital signature which was used to signed the binary file, neither was a collision hash successful on the MOCX malware.

Overview Firewall Result Detection In the end two out of four were not able to detect the MOCX malware. The explanation lies in that these firewall solutions do

[6] http://www.zonealarm.com/security/en-us/zonealarm-pc-security-free-firewall.htm

not have an effective Host-based Intrusion Prevention System. (HIPS) Either they only monitor on the user-mode level or their kernel-mode driver those not detect other kernel-mode drivers that operate on the same level. Initially only WinPcap driver was implemented in the MOCX malware for injecting network packets, but after it got silently blocked by the remaining firewall solutions WinDivert[7] was used in order to inject network packet. The motivation for this attempt was done by the assumption that these remaining firewall solutions utilized a blacklist function in order to stay protected, since WinPcap library is well-known packet sniffing/injecting tool in the security community. Therefor a less-known packet sniffing/injection library was used in order to see if any change occurred. The following table displays an overview on the results for both packet sniffing/injection library implementations in the MOCX malware.

Result	Comodo Internet Security Suite	FortKnox Firewall	Outpost Security Suite	ZoneAlarm Firewall
WinPcap	U & A	U & A	U & B	U & B
WinDivert	U & A	U & A	D & (A or B)	U & B

Legend:
A = Activated
B = Blocked
U = Undetected
D = Detected
& = And

A change was detected when Outpost Security Suite was used as firewall implementation. This change could likely imply that WinDivert kernel-mode network driver was not included in blacklist feature of Outpost Security Suite. Therefor it asked the end-user what choice should be made. The following screenshot figure displays this question.

[7] http://reqrypt.org/windivert.html

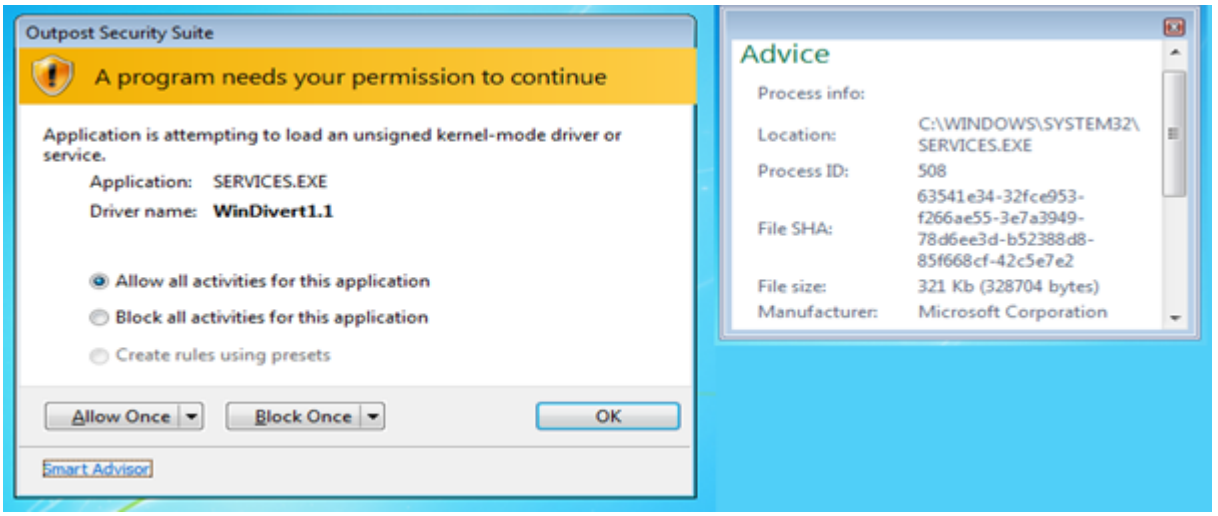


FIGURE 5.1: Screenshot Outpost

Interesting is that a legitimate windows service is loading kernel-mode driver on behalf of the MOCX malware, but the MOCX malware process named "WinDivertMalware.exe" is not seen anywhere in the request. But still this would raise suspicion by any security officer in working in the IT department of a security oriented company. The blacklisting feature which Outpost and ZoneAlarm utilizes is connected to a cloud-based environment, where in this case the MOCX malware sample gets submitted to. An interesting behavior was noticed during the submission of the MOCX Malware sample in both environment settings. During the first submission, the MOCX malware was allowed to operate within a limited time period until the submission results came back in from the cloud-based environment. After the submission results came back, the MOCX malware was not allowed to communicate anymore, but still no notification message came towards the end-user.

Chapter 6

Challenges and Solutions

There are still some unresolved challenges which might be interesting for fellow researchers who would like to continue on the XCOM research project. The following challenges will be listed here, included with their suggested solutions. These will be divided accordingly to their designated categories.

Malware Samples The challenge here lies in acquiring the right malware sample that will activate in the right environment. The malware samples that were used in this research project were carefully selected, but still better malware samples may yet be acquired for testing the XCOM Detection Tool. As mentioned before in Section 2.4 it is vital to acquire the correct malware samples, since some malware samples are specifically created for a very specific target corporate environment. If the target corporate network can be replicated under the right circumstances, then more profitable results may be gained from this research. If by any chance the right malware sample was selected and it was able to perform a communication attempt with its corresponding C&C server, then the XCOM detection tool will detect this attempt.

XCOM Detection Tool The XCOM Prototype detection tool can be improved in several ways. First of the XCOM detection tool was designed to be very flexible. Initially it was designed to block packet modification attempts and missing packets. In the end this proved to be heavy performance related. Since each packet needed to be inspected for any spoofing or stenographic attempts. The XCOM detection tool could be improved to only inspect the destination IP addresses for outgoing data transmission. This would effectively increase the performance, and still at the minimal cost of security. Since no stenographic of spoofing attempts were detected during the XCOM experiment. Unless the malware modifies the first initial packet as well. From the malware

samples in Appendix C that were scanned by the XCOM detection tool it was possible to manually inspect the network packets through the Wireshark application in order to detect if any spoofing or stenographic attempts occurred. With the manual scanning operation it turned out that not any of the selected malware samples performed any spoofing or stenographic attempts. Therefor the XCOM detection tool could be improved by only inspecting the first initial packet of each outgoing data transmission. During the development of the XCOM detection tool an attempt was made for deploying the XCOM detection tool on Openflow[1] with the floodlight controller that was available in Java[2], but finally the decision was made to develop the prototype as a stand-alone tool in Java only. The next challenge would be to deploy the XCOM detection tool as an Intrusion Detection System (IDS). This could be effectively done with Openflow. Openflow is an technology that allows for software defined networks to perform complex network switching, based on the first initial packet of an data transmission. With some additional research the XCOM detection tool could be ported to OpenFlow, since the Monitoring part is already written in Java and the communication bridge which is used for communicating with the clients is designed in a cross-platform environment, which makes porting more flexible. In order to be more resilient for future attacks by malware, the XCOM detection tool can be adequately be protected through deploying two-way authentication scheme, considering that the monitoring component does not get compromised along the way. Only then can the XCOM detection tool be secured properly. Another solution would be to deploy file integrity management system like OSSEC[3] to verify if the target process/file has been modified. But for the time being this is considered out of scope.

MOCX Prototype Malware The challenge for MOCX malware lies in the part where it is able acquires a digital signature from the Microsoft Corporation or create a hash collision with an existing binary from the Microsoft Corporation. If any other research is focused on either these two subjects, then these can be combined with the MOCX malware. In case of the WinDivert variant of the MOCX malware, this would result in a signed kernel-mode driver, which would completely avoid the cloud-based blacklist features of most anti-virus security solutions. So in the end the MOCX malware would be undetectable. Another challenge would be to expand the MOCX malware to a real virus, which contains multiple tools similar to a zombie or bot that would operate in a botnet.

[1] https://www.opennetworking.org/
[2] http://www.projectfloodlight.org/floodlight/
[3] http://www.ossec.net/

Chapter 7

Conclusion

This master thesis focused initially on defining what APT is. After creating a solid understanding on APT in chapter 1, we tended to gain more insight on APT malware samples communicate that with C&C servers. Here we discovered that the Snake malware was the most sophisticated malware at the current time. Two control groups were created in order to trap the APT malware samples in chapter 2, but without any conclusive result on how these malware samples avoid their respective virtual environments. The APT malware samples that did communicated were used for the XCOM experiment. The XCOM experiment came to existence in chapter 3 from the idea of cross-referencing network communications patterns in order to detect APT network traffic. The initial samples that were selected did not deemed enough for a conclusive result. More samples were selected for a more accurate result during the XCOM experiment, but still the APT malware samples did not reveal themselves. To be more precise, those that hide their communication towards their designated C&C servers, including the Snake APT malware. Therefor an prototype malware called 'MOCX' was created in order to replicate the stealthy communication techniques applied by the Snake malware. After the malware was created, it was installed multiple times utilizing various commercial firewalls in order to check if any firewalls could detect it. This was partial successful, only 1 out of 4 commercial firewall would automatically block it, but this was due to a blacklisting system that was discovered in chapter 5. Here we could conclude that if the recommendations mentioned in chapter 6 were applied, then the MOCX malware would be undetectable, but due to time constrains more research was needed in order to fake the digital signature on the malware binary to be successful. But regardless the XCOM detection tool could detect the WinPcap variant MOCX malware, because it used a seperate TCP network stack, which is similar to that of the Snake APT Malware. Any hiding attempt performed by utilizing a seperate TCP network stack would be detected by the XCOM detection tool. If any other researchers are interested in continuing this

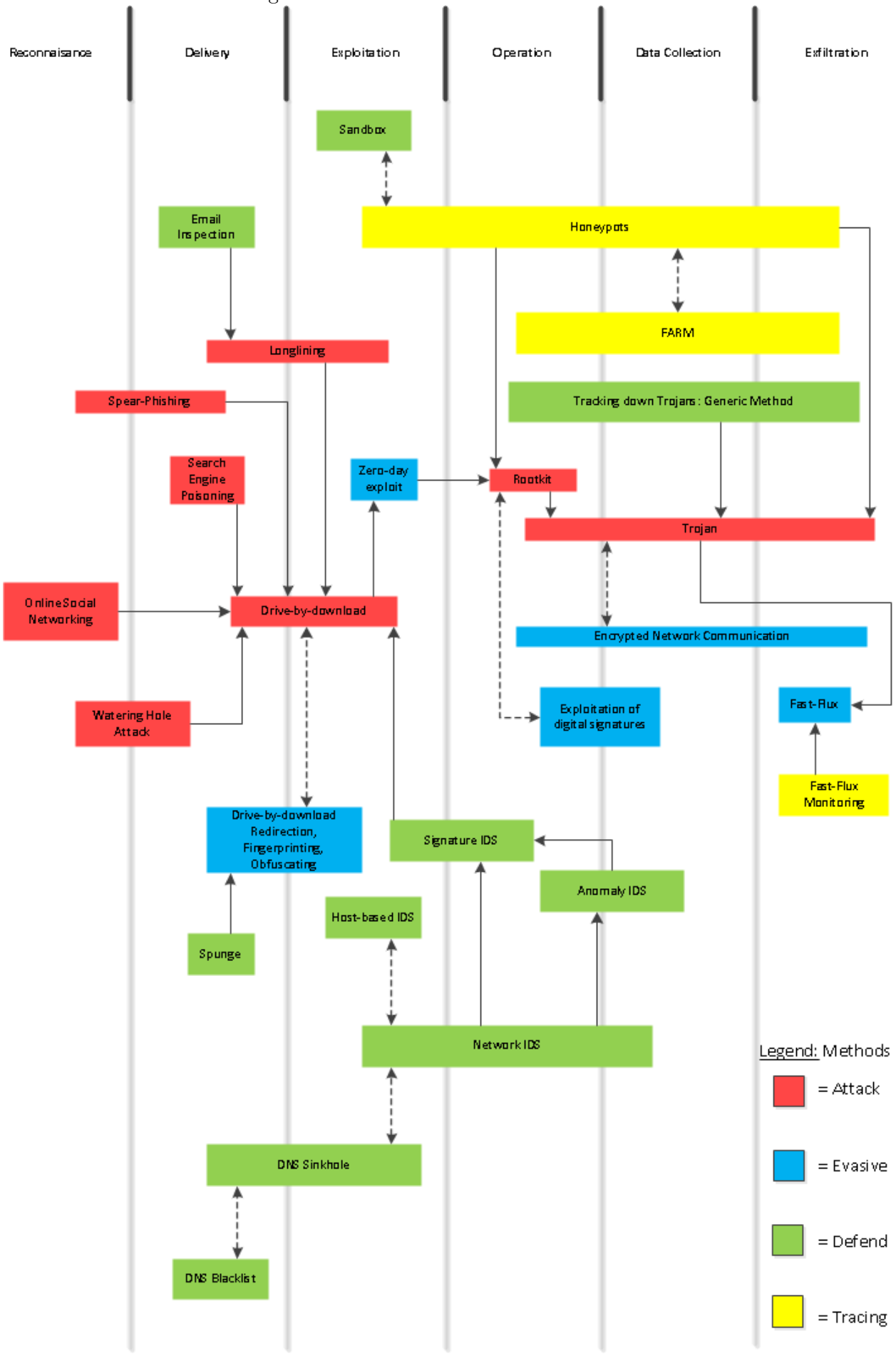
research, then the source code of both the XCOM detection tool and MOCX malware will be published on the following Github account [1].

[1] https://github.com/Tryan18/XCOM

Appendix A

Overview APT

Legend:
Double Arrow = Both elements make use of each other.
Single Arrow = Uses next element.



Appendix B

Correlation Table Wireshark Experiment

APT malware name	Correlation Detected	C&C IP	Virtual environment used	Notes
Careto	X	N/A	VirtualBox & VMware (Win XP)	
IXESHE	X	68.16.99.165	VirtualBox & VMware (Win XP)	
Lurid	X	winhelp.winxplenovo.com winhelp.winxplenovo.com.lan	VirtualBox & VMware (Win XP)	
Mirage	X	123.120.101.129 (youneedme.8866.org)	VirtualBox & VMware (Win XP)	
Snake	X	N/A	VirtualBox & VMware (Win XP)	
Sykipot	X	N/A	VirtualBox & VMware (Win XP)	Did not execute
Taidoor	X	61.7.158.11	VirtualBox & VMware (Win XP)	
Winnti	X	N/A	VirtualBox & VMware (Win XP)	Not possible due to DLL format

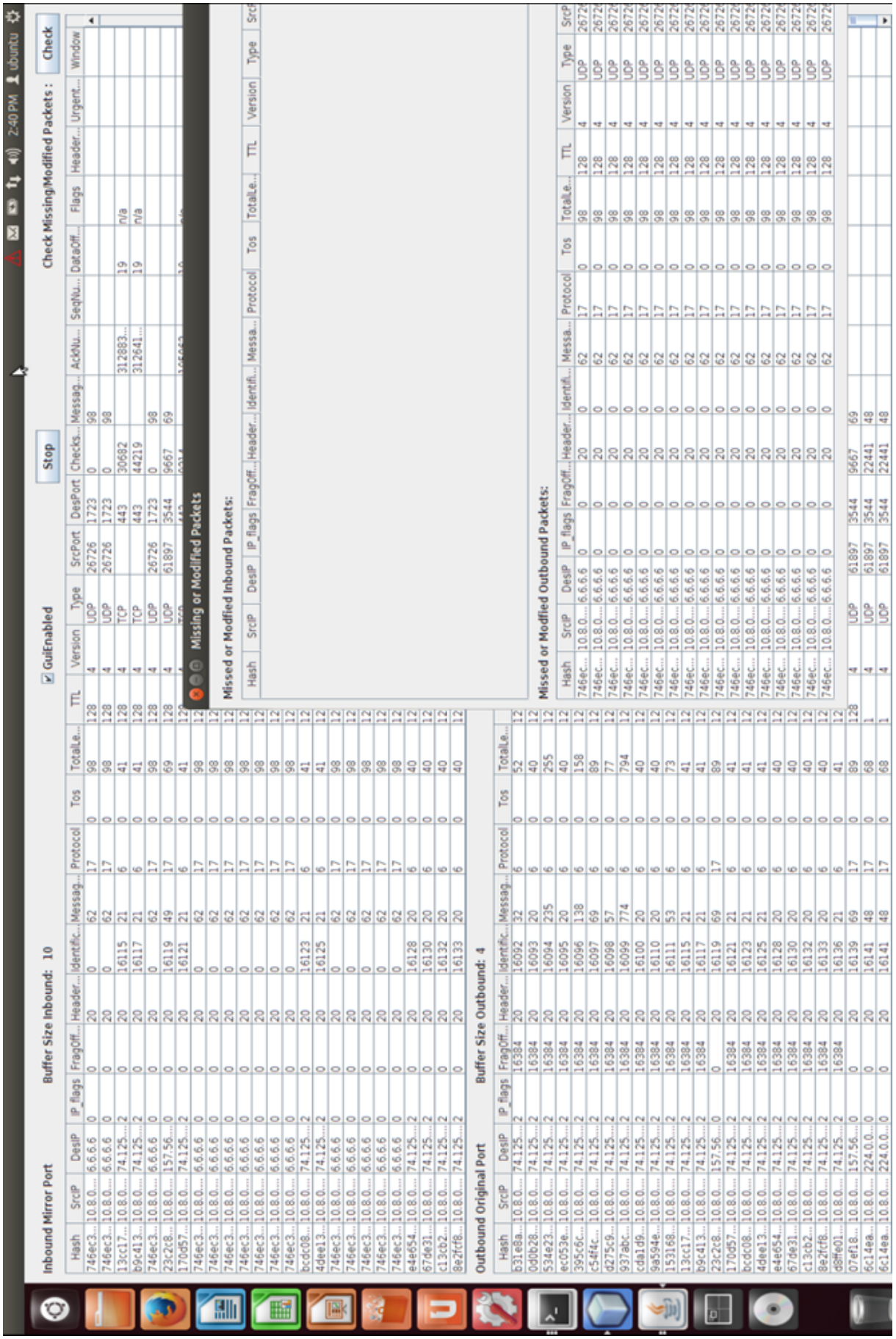
Appendix C

Correlation Table XCOM Experiment

APT malware name	Correlation Detected	C&C IP	Virtual environment used	Notes
IXESHE	☒	68.16.99.165	VirtualBox (Win 7)	
Lurid Downloader	☒	94.245.121.251 (winhelp.winxplenovo.com)	VirtualBox (Win 7)	
Mirage	☒	123.120.101.129 (youneedme.8866.org)	VirtualBox (Win 7)	
Taidoor	☒	61.7.158.11	VirtualBox (Win 7) VirtualBox (WinXP)	
9002	☒	199.36.76.113	VirtualBox (Win 7)	
DarkComet	☒	64.235.43.131, ?178.239.178.10	VirtualBox (Win 7)	
DestroyRAT	☒	N/A	VirtualBox (Win 7)	
Shamoon	☒	N/A	VirtualBox (Win 7)	
DNSWatch	☒	168.95.1.1, 139.175.55.244	VirtualBox (Win 7)	
Hupigon	☒	125.77.199.30	VirtualBox (Win 7)	
KRbanker	☒	174.139.68.18, 95.101.0.96, 92.122.189.32, 191.237.208.126, 157.56.144.215	VirtualBox (Win 7)	
Letsgo	☑	191.237.208.126	VirtualBox (Win 7)	Detected 1 packet miss
Mediana	☒	95.211.172.143	VirtualBox (Win 7)	performs dns 8.8.8.8 retry infinite
MSWab	☒	?74.125.136.138	VirtualBox (Win 7)	
njRat	☒	217.66.231.245	VirtualBox (Win 7)	
Wumins	☒	191.237.208.126	VirtualBox (Win 7)	
PlugX	☒	N/A	VirtualBox (Win 7)	
Surtr	☒	N/A	VirtualBox (Win 7)	invalid processor architecture
Taleret	☒	N/A	VirtualBox (Win 7)	no valid win32 application
Tapaoux	☒	N/A	VirtualBox (Win 7)	performs some sort of system check. Possible detection method for detecting virtual environment
Vidgrab	☒	113.10.246.46	VirtualBox (Win 7)	
MOCX	☑	6.6.6.6	VirtualBox (Win 7)	WinPcap variant

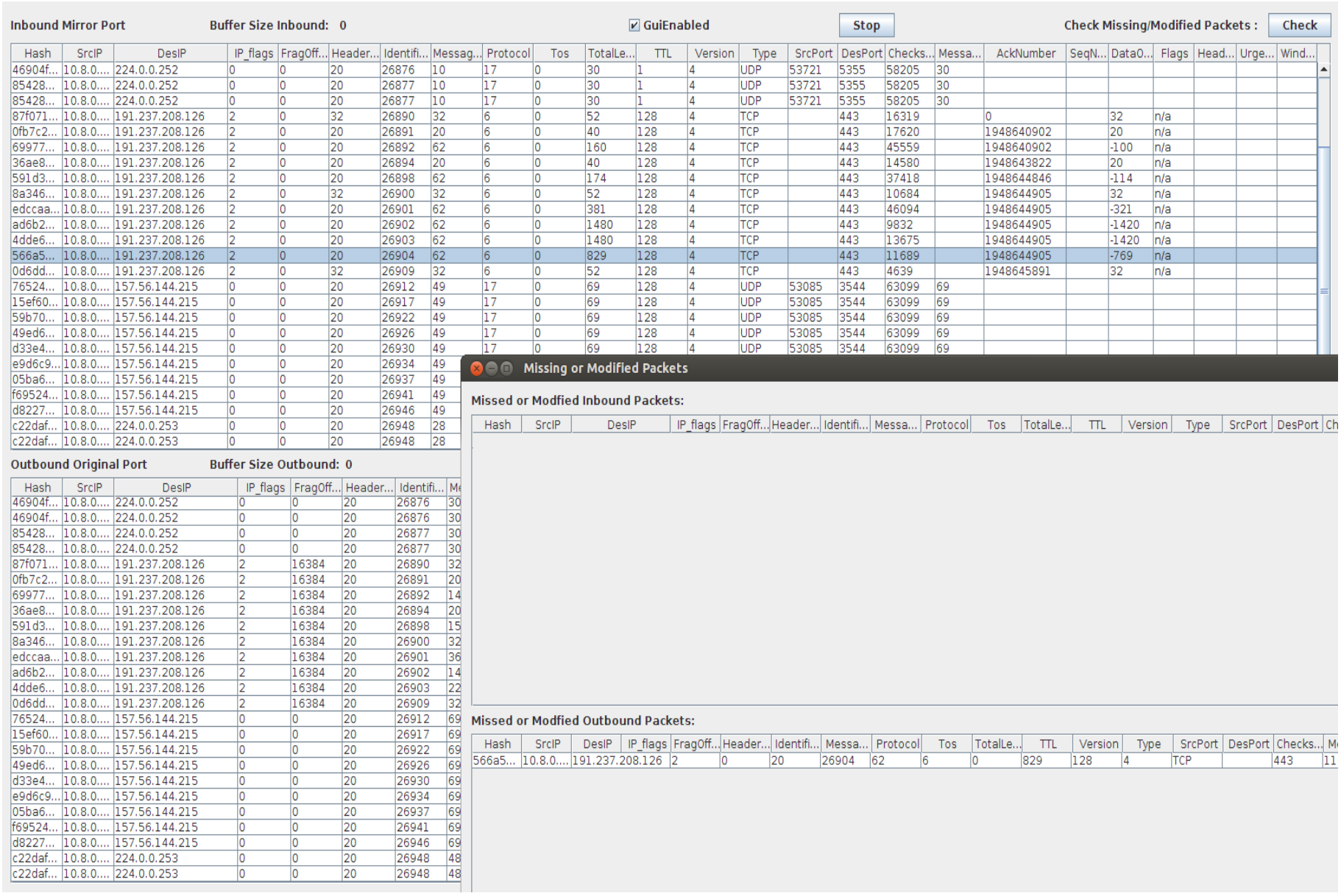
Appendix D

Detection Result XCOM Experiment Scenario 1


ubuntu
2:40 PM
Inbound Mirror Port
Buffer Size Inbound: 10
GuiEnabled
Stop
Check Missing/Modified Packets :
Check
Missing or Modified Packets
Missed or Modified Inbound Packets:
Missed or Modified Outbound Packets:
Outbound Original Port
Buffer Size Outbound: 4

Appendix E

Detection Result XCOM Experiment Scenario 2



Inbound Mirror Port Buffer Size Inbound: 0 GuiEnabled Stop Check Missing/Modified Packets : Check

Hash	SrcIP	DesIP	IP_flags	FragOff...	Header...	Identifi...	Messag...	Protocol	Tos	TotalLe...	TTL	Version	Type	SrcPort	DesPort	Checks...	Messa...	AckNumber	SeqN...	DataO...	Flags	Head...	Urge...	Wind...
46904f...	10.8.0....	224.0.0.252	0	0	20	26876	10	17	0	30	1	4	UDP	53721	5355	58205	30							
85428...	10.8.0....	224.0.0.252	0	0	20	26877	10	17	0	30	1	4	UDP	53721	5355	58205	30							
85428...	10.8.0....	224.0.0.252	0	0	20	26877	10	17	0	30	1	4	UDP	53721	5355	58205	30							
87f071...	10.8.0....	191.237.208.126	2	0	32	26890	32	6	0	52	128	4	TCP		443	16319		0		32	n/a			
0fb7c2...	10.8.0....	191.237.208.126	2	0	20	26891	20	6	0	40	128	4	TCP		443	17620		1948640902		20	n/a			
69977...	10.8.0....	191.237.208.126	2	0	20	26892	62	6	0	160	128	4	TCP		443	45559		1948640902		-100	n/a			
36ae8...	10.8.0....	191.237.208.126	2	0	20	26894	20	6	0	40	128	4	TCP		443	14580		1948643822		20	n/a			
591d3...	10.8.0....	191.237.208.126	2	0	20	26898	62	6	0	174	128	4	TCP		443	37418		1948644846		-114	n/a			
8a346...	10.8.0....	191.237.208.126	2	0	32	26900	32	6	0	52	128	4	TCP		443	10684		1948644905		32	n/a			
edccaa...	10.8.0....	191.237.208.126	2	0	20	26901	62	6	0	381	128	4	TCP		443	46094		1948644905		-321	n/a			
ad6b2...	10.8.0....	191.237.208.126	2	0	20	26902	62	6	0	1480	128	4	TCP		443	9832		1948644905		-1420	n/a			
4dde6...	10.8.0....	191.237.208.126	2	0	20	26903	62	6	0	1480	128	4	TCP		443	13675		1948644905		-1420	n/a			
566a5...	10.8.0....	191.237.208.126	2	0	20	26904	62	6	0	829	128	4	TCP		443	11689		1948644905		-769	n/a			
0d6dd...	10.8.0....	191.237.208.126	2	0	32	26909	32	6	0	52	128	4	TCP		443	4639		1948645891		32	n/a			
76524...	10.8.0....	157.56.144.215	0	0	20	26912	49	17	0	69	128	4	UDP	53085	3544	63099	69							
15ef60...	10.8.0....	157.56.144.215	0	0	20	26917	49	17	0	69	128	4	UDP	53085	3544	63099	69							
59b70...	10.8.0....	157.56.144.215	0	0	20	26922	49	17	0	69	128	4	UDP	53085	3544	63099	69							
49ed6...	10.8.0....	157.56.144.215	0	0	20	26926	49	17	0	69	128	4	UDP	53085	3544	63099	69							
d33e4...	10.8.0....	157.56.144.215	0	0	20	26930	49	17	0	69	128	4	UDP	53085	3544	63099	69							
e9d6c9...	10.8.0....	157.56.144.215	0	0	20	26934	49																	
05ba6...	10.8.0....	157.56.144.215	0	0	20	26937	49																	
f69524...	10.8.0....	157.56.144.215	0	0	20	26941	49																	
d8227...	10.8.0....	157.56.144.215	0	0	20	26946	49																	
c22daf...	10.8.0....	224.0.0.253	0	0	20	26948	28																	
c22daf...	10.8.0....	224.0.0.253	0	0	20	26948	28																	

Outbound Original Port Buffer Size Outbound: 0

Hash	SrcIP	DesIP	IP_flags	FragOff...	Header...	Identifi...	M
46904f...	10.8.0....	224.0.0.252	0	0	20	26876	30
46904f...	10.8.0....	224.0.0.252	0	0	20	26876	30
85428...	10.8.0....	224.0.0.252	0	0	20	26877	30
85428...	10.8.0....	224.0.0.252	0	0	20	26877	30
87f071...	10.8.0....	191.237.208.126	2	16384	20	26890	32
0fb7c2...	10.8.0....	191.237.208.126	2	16384	20	26891	20
69977...	10.8.0....	191.237.208.126	2	16384	20	26892	14
36ae8...	10.8.0....	191.237.208.126	2	16384	20	26894	20
591d3...	10.8.0....	191.237.208.126	2	16384	20	26898	15
8a346...	10.8.0....	191.237.208.126	2	16384	20	26900	32
edccaa...	10.8.0....	191.237.208.126	2	16384	20	26901	36
ad6b2...	10.8.0....	191.237.208.126	2	16384	20	26902	14
4dde6...	10.8.0....	191.237.208.126	2	16384	20	26903	22
0d6dd...	10.8.0....	191.237.208.126	2	16384	20	26909	32
76524...	10.8.0....	157.56.144.215	0	0	20	26912	69
15ef60...	10.8.0....	157.56.144.215	0	0	20	26917	69
59b70...	10.8.0....	157.56.144.215	0	0	20	26922	69
49ed6...	10.8.0....	157.56.144.215	0	0	20	26926	69
d33e4...	10.8.0....	157.56.144.215	0	0	20	26930	69
e9d6c9...	10.8.0....	157.56.144.215	0	0	20	26934	69
05ba6...	10.8.0....	157.56.144.215	0	0	20	26937	69
f69524...	10.8.0....	157.56.144.215	0	0	20	26941	69
d8227...	10.8.0....	157.56.144.215	0	0	20	26946	69
c22daf...	10.8.0....	224.0.0.253	0	0	20	26948	48
c22daf...	10.8.0....	224.0.0.253	0	0	20	26948	48

Missing or Modified Packets

Missed or Modfied Inbound Packets:

Hash	SrcIP	DesIP	IP_flags	FragOff...	Header...	Identifi...	Messa...	Protocol	Tos	TotalLe...	TTL	Version	Type	SrcPort	DesPort	Ch

Missed or Modfied Outbound Packets:

Hash	SrcIP	DesIP	IP_flags	FragOff...	Header...	Identifi...	Messa...	Protocol	Tos	TotalLe...	TTL	Version	Type	SrcPort	DesPort	Checks...	M
566a5...	10.8.0....	191.237.208.126	2	0	20	26904	62	6	0	829	128	4	TCP		443	11	

Bibliography

[1] Mandiant. M-trends: Attack the security gap. January 2013.

[2] R. Trend Micro: Ferguson. The botnet chronicles a journey to infamy. November 2010.

[3] N. Virvilis and D. Gritzalis. The big four - what we did wrong in advanced persistent threat detection? pages 248–254, 2013. cited By (since 1996)1.

[4] Birhanu Eshete. Effective analysis, characterization, and detection of malicious web pages. In *Proceedings of the 22nd international conference on World Wide Web companion*, pages 355–360. International World Wide Web Conferences Steering Committee, 2013.

[5] *Assessing Outbound Traffic to Uncover Advanced Persistent Threat*, may 2011. SANS Technology Institute.

[6] P. Giura and W. Wang. A context-based detection framework for advanced persistent threats. pages 69–74, 2013. cited By (since 1996)0.

[7] *Rsa Security Brief: Mobilizing Intelligent Security Operations for Advanced Persistent Threats*, 2011. cited By (since 1996)2.

[8] J. Andress. Advanced persistent threat, attacker sophistication continues to grow? *ISSA Journal*, 2011. cited By (since 1996)1.

[9] F. Li, A. Lai, and D. Ddl. Evidence of advanced persistent threat: A case study of malware for political espionage. pages 102–109, 2011. cited By (since 1996)3.

[10] R. Langner. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security and Privacy*, 9(3):49–51, 2011. cited By (since 1996)48.

[11] WebSense. Advanced persistent threats and other advanced attacks: Threat analysis and defense strategies for smb, mid-size, and enterprise organizations. September 2011.

[12] E. Hutchins, C. Cloppert, and R. Amin. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. pages 113–125, 2011. cited By (since 1996)1.

[13] Y. Altshuler, N. Aharony, A. Pentland, Y. Elovici, and M. Cebrian. Stealing reality: When criminals become data scientists (or vice versa). *IEEE Intelligent Systems*, 26(6):22–30, 2011. cited By (since 1996)3.

[14] T. Caldwell. Spear-phishing: How to spot and mitigate the menace. *Computer Fraud and Security*, 2013(1):11–16, 2013. cited By (since 1996)0.

[15] M. Balduzzi, V. Ciangaglini, and R. McArdle. Targeted attacks detection with spunge. pages 185–194, 2013. cited By (since 1996)0.

[16] A. Caglayan, M. Toothaker, D. Drapeau, D. Burke, and G. Eaton. Behavioral analysis of botnets for threat intelligence. *Information Systems and e-Business Management*, 10(4):491–519, 2012. cited By (since 1996)0.

[17] Bojan Simic and James Walden. Eliminating sql injection and cross site scripting using aspect oriented programming. In *Engineering Secure Software and Systems*, pages 213–228. Springer, 2013.

[18] Fortinet. Anatomy of a botnet. 2012.

[19] Thanassis Avgerinos, Sang Kil Cha, Alexandre Rebert, Edward J. Schwartz, Maverick Woo, and David Brumley. Automatic exploit generation. *Commun. ACM*, 57(2):74–84, February 2014.

[20] R. Lemos. Stuxnet attack more effective than bombs. January 2011.

[21] Boldizsár Bencsáth, Gábor Pék, Levente Buttyán, and Márk Félegyházi. The cousins of stuxnet: Duqu, flame, and gauss. *Future Internet*, 4(4):971–1003, 2012.

[22] Securelist. "red october" diplomatic cyber attacks investigation. January 2013.

[23] SANS Institute. Security consensus operational readiness evaluation. February 2014. https://www.sans.org/score/checklists/rootkits_investigation_procedures.pdf.

[24] C. Vatamanu, D. Gavriluţ, and R. Benchea. A practical approach on clustering malicious pdf documents. *Journal in Computer Virology*, 8(4):151–163, 2012. cited By (since 1996)0.

[25] C. The Guardian: Arthur. How the smartphone is killing the pc. June 2011.

[26] Y. Zeng, K.G. Shin, and X. Hu. Design of sms commanded-and-controlled and p2p-structured mobile botnets. pages 137–148, 2012. cited By (since 1996)5.

[27] Ming Song, Gang Xiong, Zhenzhen Li, Junrui Peng, and Li Guo. A de-anonymize attack method based on traffic analysis. In *Communications and Networking in China (CHINACOM), 2013 8th International ICST Conference on*, pages 455–460, Aug 2013.

[28] YoungHwan Lim, DongHwi Lee, WonHyung Park, and KwangHo Kook. Detection and traceback of illegal users based on anonymous network in bittorrent environment. *Wireless Personal Communications*, 73(2):319–328, 2013.

[29] L. Cao and X. Qiu. Asp2p: An advanced botnet based on social networks over hybrid p2p. pages 677–682, 2013. cited By (since 1996)0.

[30] Anthony Desnos, Éric Filiol, and Ivan Lefou. Detecting (and creating !) a hvm rootkit (aka bluepill-like). *J. Comput. Virol.*, 7(1):23–49, February 2011.

[31] D. Smith. Life's certainties: Death, taxes and apts. *Network Security*, 2013(2):19–20, 2013. cited By (since 1996)0.

[32] M. Potts. The state of information security. july 2012. ISSN=1353-4858.

[33] Verizon RISK Team et al. Verizon 2013 data breach investigations report. 2013.

[34] TrendLabs APT Research Team et al. Spear-phishing email: Most favored apt attack bait. *Last accessed September*, 2:2013, 2012.

[35] N.N.A. Molok, S. Chang, and A. Ahmad. Information leakage through online social networking: Opening the doorway for advanced persistence threats. pages 70–80, 2010. cited By (since 1996)1.

[36] Invincea. Spear-phishing, watering hole and drive-by attacks: The new normal. June 2013.

[37] R. InfoWorld: Grimes. Watch out for waterhole attacks – hackers' latest stealth weapon. May 2013.

[38] ProofPoint. Longline phishing: Email-borne threats, cloud computing, big data, and the rise of industrial phishing attacks. 2013.

[39] L. Lu, R. Perdisci, and W. Lee. Surf: Detecting and measuring search poisoning. pages 467–476, 2011. cited By (since 1996)2.

[40] M. Network World: Boodaei. Man-in-the-browser attacks target the enterprise. May 2011.

[41] M. Cova, C. Kruegel, and G. Vigna. Detection and analysis of drive-by-download attacks and malicious javascript code. pages 281–290, 2010. cited By (since 1996)42.

[42] John Aycock. What's in a name... generator? *Journal in Computer Virology*, 8(1-2):53–60, 2012.

[43] The Honeynet Project. Know your enemy: Fast-flux service networks. July 2007.

[44] BAE Systems. Snake campaign & cyber espionage toolkit. page 31, 2014.

[45] I. Gregorio-de Souza, V.H. Berk, A. Giani, G. Bakos, M. Bates, G. Cybenko, and D. Madory. Detection of complex cyber attacks. volume 6201, 2006. cited By (since 1996)2.

[46] *M Trends, the Advanced Persistent Threat*, 2010. cited By (since 1996)3.

[47] CLEARSWIFT. Behaviors and characteristics of advanced persistent threats. 2012. Presentation Slide.

[48] R.F. Erbacher and S.E. Hutchinson. Extending case-based reasoning to network alert reporting. pages 187–194, 2013. cited By (since 1996)0.

[49] Erik Brown, Bo Yuan, Daryl Johnson, and Peter Lutz. Covert channels in the http network protocol: Channel characterization and detecting man-in-the-middle attacks. In *The Proceedings of the 5th International Conference on Information Warfare and Security: The Air Force Institute of Technology, Wright-Patterson AFB, Ohio, USA, 8-9 April 2010*, page 56. Academic Conferences Limited, 2010.

[50] Y. Liang, G. Peng, H. Zhang, and Y. Wang. An unknown trojan detection method based on software network behavior. *Wuhan University Journal of Natural Sciences*, 18(5):369–376, 2013. cited By (since 1996)0.

[51] K.-H. Choi, W.H. Park, and K.J. Kim. A study of esmtc(enterprise security management system based on threshold classification). 2012. cited By (since 1996)0.

[52] SANS Institute InfoSec Reading Room. Dns sinkhole. 2010. https://www.sans.org/reading-room/whitepapers/dns/dns-sinkhole_33523?show=dns-sinkhole_33523.

[53] Anirudh Ramachandran, Nick Feamster, and David Dagon. Revealing botnet membership using dnsbl counter-intelligence. *Proc. 2nd USENIX Steps to Reducing Unwanted Traffic on the Internet*, pages 49–54, 2006.

[54] S. SANS Institute InfoSec Reading Room: Misenar. A virtually secure browser. June 2009.

[55] Manasi Bhattacharyya, Shlomo Hershkop, and Eleazar Eskin. Met: An experimental system for malicious email tracking. In *Proceedings of the 2002 workshop on New security paradigms*, pages 3–10. ACM, 2002.

[56] Ching-Hsiang Hsu, Chun-Ying Huang, and Kuan-Ta Chen. Fast-flux bot detection in real time. In *Recent Advances in Intrusion Detection*, pages 464–483. Springer, 2010.

[57] A. Elouafiq, A. Khobalatte, W. Benhallam, O. Iraqi, and Tajje-EddineRachidi. Aggressive and intelligent self-defensive network towards a new generation of semi-autonomous networks. *Journal of Emerging Technologies in Web Intelligence*, 5(1):12–17, 2013. cited By (since 1996)0.

[58] Jamie Van Randwyk, Ken Chiang, Levi Lloyd, and Keith B Vanderveen. Farm: An automated malware analysis environment. In *Security Technology, 2008. ICCST 2008. 42nd Annual IEEE International Carnahan Conference on*, pages 321–325. IEEE, 2008.

[59] Tal Garfinkel, Keith Adams, Andrew Warfield, and Jason Franklin. Compatibility is not transparency: Vmm detection myths and realities. In *Proceedings of the 11th USENIX Workshop on Hot Topics in Operating Systems*, HOTOS'07, pages 6:1–6:6, Berkeley, CA, USA, 2007. USENIX Association.

[60] Tom Liston and Ed Skoudis. On the cutting edge: Thwarting virtual machine detection. Url: (http://handlers.sans.org/tliston/ThwartingVMDetection_Liston_Skoudis.pdf).

[61] Kaspersky Lab. Unveiling "careto" - the masked apt. Url: http://kasperskycontenthub.com/wp-content/uploads/sites/43/vlpdfs/unveilingthemask_v1.0.pdf.

[62] Matsukawa Bakuei Nart Villeneuve David Sancho, Jessa dela Torre and Robert McArdle. Ixeshe an apt campaign. Url: http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp_ixeshe.pdf.

[63] David Sancho Nart Villeneuve. The "lurid" downloader. Url: http://la.trendmicro.com/media/misc/lurid-downloader-enfal-report-en.pdf.

[64] Silas Cutler. The mirage campaign. Url: http://www.secureworks.com/cyber-threat-intelligence/threats/the-mirage-campaign/.

[65] G Data SecurityLabs. Uroburos highly complex espionage software with russian roots. Url: https://public.gdatasoftware.com/Web/Content/INT/Blog/2014/02_2014/documents/GData_Uroburos_RedPaper_EN_v1.pdf.

[66] BAE Systems Applied Intelligence. Snake campaign & cyber espionage toolkit. Url: http://info.baesystemsdetica.com/rs/baesystems/images/snake_whitepaper.pdf.

[67] tecamac@gmail.com deresz@gmail.com. Uroburos: the snake rootkit. Url: http://artemonsecurity.com/uroburos.pdf.

[68] Chong Rong Hwa. Detailed analysis of sykipot. Url: http://www.sans.org/reading-room/whitepapers/malicious/detailed-analysis-sykipot-smartcard-proxy-variant-33919.

[69] Piotr Krysiuk Stephen Doherty. Trojan.taidoor: Targeting think tanks. Url: http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/trojan_taidoor-targeting_think_tanks.pdf.

[70] Trend Micro Threat Research Team. The taidoor campaign. Url: http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp_the_taidoor_campaign.pdf.

[71] Kaspersky Lab Global Research and Analysis Team. "winnti" more than just a game. Url: http://kasperskycontenthub.com/wp-content/uploads/sites/43/vlpdfs/winnti-more-than-just-a-game-130410.pdf.

[72] AhnLab. Etso apt attacks analysis. Url: http://image.ahnlab.com/global/upload/download/documents/1401223631603288.pdf.