

## THESIS ABSTRACT

Master of Science in Applied Computer Science

Adventist University of Africa

School of Postgraduate Studies

**TITLE: A COMPENSATORY APPROACH TO ANTI-VIRUS SHORTFALLS**

Researcher: Tom Ombaba Ongaro

Primary Adviser: Jules Pagna Disso, PhD

Date Completed: November 2019

Computer systems security has become an increasingly important field. In the quest to provide the much-needed security many options exist. Systems have however continued to suffer attacks from malware despite the existing controls that have been put in place. One such control is the use of Anti-viruses which are widely used in many systems.

Today malware exists that can bypass anti-viruses and cause harm to systems. Many controls exist to try to combat malware infiltration. Organizations and small businesses may not always be in a position to choose the best option for their environment when it comes to dealing with malware. They may not also be able to configure system security tools that may be available to deal with malware detection and prevention.

One freely available tool is Sysmon. Sysmon logs critical events in a windows environment and can send them out for further analysis and classification. This research seeks to understand why some malware can bypass anti-viruses and seeks to

close the gap by providing tangible recommendations. The end goal provides results that can be adopted by anyone to try to identify malicious activity in their systems by using freely available tools.

Adventist University of Africa  
School of Postgraduate Studies

A COMPENSATORY APPROACH TO ANTI-VIRUS SHORTFALLS

A thesis  
presented in partial fulfillment  
of the requirements for the degree  
Master of Science in Applied Computer Science

by  
Tom Ombaba Ongaro

May 2020

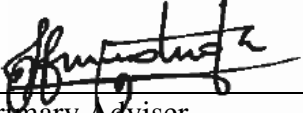



A COMPENSATORY APPROACH TO ANTI-VIRUS SHORTFALLS

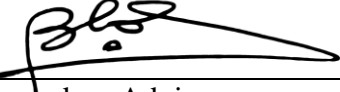
A thesis  
presented in partial fulfillment  
of the requirements for the degree  
Master of Science in Applied Computer Science

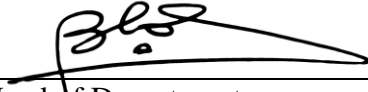
by  
Tom Ombaba Ongaro

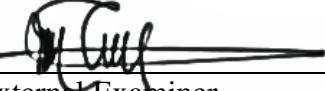
APPROVAL BY THE COMMITTEE:

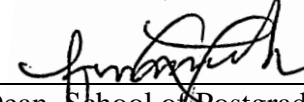
  
\_\_\_\_\_  
Primary Adviser  
Jules Pagna, PhD

  
\_\_\_\_\_  
Programme Director, MScACS  
Lossan Bonde, PhD

  
\_\_\_\_\_  
Secondary Adviser  
Lossan Bonde, PhD

  
\_\_\_\_\_  
Head of Department  
Lossan Bonde, PhD

  
\_\_\_\_\_  
External Examiner  
Paul-Marie Moulema Douala, DSc

  
\_\_\_\_\_  
Dean, School of Postgraduate Studies  
Daniel Ganu, DrPH

AUA Main Campus

Date: May 2020

## TABLE OF CONTENTS

LIST OF FIGURES .....	vii
ACKNOWLEDGMENTS .....	ix
CHAPTER	
1. INTRODUCTION .....	1
Motivation.....	1
Statement of the Problem.....	2
Research Objectives.....	2
2. LITERATURE REVIEW .....	3
A Brief History of Anti-Viruses .....	3
How Anti-Viruses Work.....	4
Methods of Malware Detection .....	4
Current Tools for Bypassing Anti-Viruses .....	5
Marble .....	5
Grasshopper .....	6
HIVE .....	6
Anti-Virus Evasion Tool (AVET) .....	6
peCloak.py .....	6
PowerShell .....	7
Veil-Evasion .....	7
Shellter .....	8
MsfVenom .....	9
Metasploit .....	9
Techniques Behind The Tools .....	10
Obfuscation .....	10
Polymorphism .....	12
Garbage code insertion. ....	12
Instruction substitution techniques. ....	12
Code-transposition. ....	12
Register-reassignment.....	12
Tools using polymorphism. ....	12
Countering effects of polymorphism. ....	13
Encryption.....	13
Encoding .....	15
Sysmon: A Compensatory Tool.....	17
Sysmon Capabilities.....	17
Events to Investigate in Sysmon Logs .....	18

3. METHODOLOGY .....	20
Research Methodology Justification.....	20
Overview .....	20
Objectives .....	21
Kali Linux Virtual Environment.....	22
Metasploit Framework .....	22
Windows 7 Virtual Machine .....	23
Payloads .....	24
Generated Payloads.....	24
Acquired Payloads .....	24
Somoto payload. ....	24
Artemis payload. ....	24
Dyre payload. ....	24
NiVdort payload.....	25
njRAT payload.....	25
ELK (Elasticsearch, Logstash, Kibana) Stack .....	25
Sysmon.....	25
Winlogbeat.....	25
ElastAlert .....	26
4. RESULTS AND ANALYSIS .....	27
Malware Dataset .....	27
Analysis of the Data.....	28
mir755.exe Malware .....	28
power.exe Malware.....	29
Event Execution Sequence and Timing Analysis .....	35
Mitre ATT&CK .....	38
Event Alerting.....	41
5. CONCLUSION AND RECOMMENDATIONS .....	43
Contributions.....	43
Limitations .....	44
Recommendations.....	45
Future Work Expectations .....	45
Summary .....	46
APPENDICES .....	47
A. PAYLOAD GENERATION.....	48
B. INSTALLING AND CONFIGURING ELK .....	51
C. INSTALLING AND CONFIGURING SYSMON .....	58
D. INSTALLING AND CONFIGURING WINLOGBEAT.....	61
E. INSTALLING AND CONFIGURING ELASALERT .....	66

F. GRAPHS FOR MALWARE EVENT SEQUENCE .....	69
REFERENCES .....	77



## LIST OF FIGURES

1. Traditional vs Modern Malware .....	10
2. Sysmon Events.....	19
3. VirusTotal for mirc755.exe.....	31
4. Antiscan.me for mirc755.exe .....	32
5. VirusTotal for power.exe .....	33
6. Antiscan for power.exe .....	34
7. Events Triggered by Each Malware.....	35
8. Network Connection .....	39
9. Remote Thread Creation .....	39
10. Registry Event and File Create Stream Hash.....	40
11. Sample Telegram Alert .....	42
12. Java-version Check .....	51
13. Elasticsearch Status Check .....	52
14. Elasticsearch Service Enabling .....	53
15. Kibana File Edit .....	53
16. Kibana Service Check.....	54
17. Nginx Status Check.....	55
18. Nginx Configuration Commands .....	55
19. Nginx Status Check and Private Keys .....	56
20. Logstash Configuration.....	57
21. Logstash Output File Configuration .....	57
22. Logstash Status Check .....	57
23. Sysmon Version Check.....	59

24. Sysmon Installation.....	59
25. Successful Sysmon Install message .....	60
26. Sample Operation Sysmon Event Viewer.....	60
27. Checking Winlogbeat Version.....	61
28. Possible Error During Installation .....	62
29. Fixing The Error .....	63
30. Pointing to Sysmon Logs .....	64
31. Defining the Host.....	64
32. Winlogbeat Status Check.....	65
33. Starting Winlogbeat .....	66
34. ELK operational Check.....	67
35. Sample Logs from Windows 7 Machine .....	67
36. mirc755 Event Sequence.....	69
37. Power.exe Event Sequence .....	70
38. Nlvdort Event Sequence .....	71
39. Njrat Event Sequence.....	72
40. Somoto1 Event Sequence .....	73
41. Somoto2 Event Sequence .....	74
42. Artemis1 Event Sequence .....	75
43. Artemis2 Event Sequence .....	76
44. Artemis3 Event Sequence .....	76

## ACKNOWLEDGMENTS

This thesis was made possible by the support of various family members, colleagues, and friends. I am indebted to them for their valuable support that made this work possible.

To my dear wife thanks for the wonderful support that I got from you and for guiding the kids when I was not at home. To my wonderful son and daughter who always added a smile to my face when I came back home probing me as to what defense meant.

To my parents. My Dad who taught us that nobody owed us a living and that we ought to never give up in life. I would have wished for my Mom to be alive at this moment but I treasure the advice she gave me to value education and use it as the strongest tool in the world to eradicate poverty. To my siblings far and near who always called me to ask me how things were.

To the AUA Computer Science Department for providing a chance for me to learn and acquire new skills and even inspire me to come up with this thesis. Thanks to Dr. Lossan Bonde for pushing us even when we thought there was no more distance to cover. You have done an excellent job in directing this programme.

Thanks, Dr. Jules Pagna. Your expertise and vast experience in the field of Cyber Security made me feel privileged to have you as my primary advisor. Your guidance has been extremely valuable. Your malware expertise was essential in helping me formulate this project. I could not have asked for a better advisor.

And to the many professors over the years from various academic institutions who helped shape our ideas and prepare the ground for this research.

To the Adventist University of Africa and all those involved in making this happen, thank you for thinking of the need to provide qualified cybersecurity professionals. This degree fills a critical gap in the field of computer science. I will be a proud graduate of this degree from the Adventist University of Africa.

Above all, I want to thank the Almighty who gave me the power and wisdom to do all that I did. Nothing afore-mentioned would have been possible without you.

## CHAPTER 1

### INTRODUCTION

#### **Motivation**

In the era of the widespread of internet usage and the multiplication of computer systems around the world, people need some form of protection from malicious activities. The increase of attacks both local and international have also increased the need to have our devices protected whenever we are exposed to the internet.

On the other hand, hackers have also intensified their activities because of the high hopes of getting their intentions met with minimal evidence which is different from the old way of stealing things. All that hackers need to do now is just get into your machine instead of getting into your house. This has led to the development of malware that sometimes goes unnoticed until their desired end is met.

Internet users have often used Anti-viruses in the view of getting protection from viruses and other malware that may be existing on the network. It is however not the case that all viruses get caught by the Anti-viruses that people choose to install in their systems. End users have been disappointed when they get attacked by Viruses whereas they had installed Anti-viruses in their systems.

The loss that comes because of Cybercrime continues to rise at unprecedented speeds. It is estimated that in the next five years the cost of cybercrime will be \$ 5.2 trillion [1].

## **Statement of the Problem**

It is therefore imperative to look at how Anti-viruses work and also how hackers have been able to bypass Anti-Viruses. It will be the intention of this thesis to try to address the gap between the two in the view of increasing the efficiency of malware detection and boost compensation control and reduce financial crime.

## **Research Objectives**

In this thesis, I have briefly reviewed how Anti-viruses work and the techniques used in popular tools to bypass Anti-viruses. The techniques used will give insight into how malware can get into legitimate systems and how they accomplish their tasks. The research looks at ways of compensating this weakness through logging. I conclude with recommendations on what can be done to increase the efficiency of Anti-viruses against malware detection, especially from the enterprise level perspective.

## CHAPTER 2

### LITERATURE REVIEW

#### **A Brief History of Anti-Viruses**

In the early days of the computer industry attacking computers and software was not feasible because most of the computers and software were isolated. Things started changing around 1968 when modems and multiplexors were developed. This made it easier for people to remotely access other machines. With the advance in similar technology, the internet era came into being. In 1982 a group of hackers broke into 60 computer systems. According to Duncan [2], this attack led to the first congressional hearings on computer security and also to new laws against cybercrime.

Hackers started forming into groups that shared data and information. These groups started to appear in the early 1980s. With the rise of viruses, Anti-viruses started to become popular. According to techlineinfor.com [3] by 1987, there were two Anti-Virus utilities available; namely Flushot plus and Anti4us. Between 1987 and 1989 a group called “Virus-L” was being used to update individuals about security and sharing information tools, and shareware to help remove the virus infection. Two individuals were on the list and that is John McAfee and Eugene Kaspersky. From these two individuals, we have two popular Anti-virus software names by their last names. In 1989 John McAfee went on to develop his own business that was selling software that protected both hardware and software.

## **How Anti-Viruses Work**

Anti-virus software is designed to detect, prevent, and even take action against malicious software that may be found on computers. It is supposed to remove viruses, worms, and Trojan horses. Anti-viruses can further be used to remove unwanted spyware and adware. Anti-virus software begins by checking your computer and comparing it to known attacks. The known attacks normally have peculiar signatures in their code that are used to identify them. It can also check for behaviors or activities that are unusual in your computer. In the proceeding section, I will discuss in detail these methods and how they work to accomplish their work.

## **Methods of Malware Detection**

Signature-based malware detectors work by comparing malicious codes with known signature databases. A binary that appears to be untrusted is scanned to find out unique byte-sequences. When a binary is confirmed as malicious its signature is stored in a database which is then used to update Anti-virus software. When that malicious code appears somewhere else it is then identified as malicious and is then either quarantined or deleted based on the preferred action and the previous configuration of the Anti-Virus system. This is one of the most common methods of malware detection. The escalating rate of new malware and the advent of self-mutating polymorphic malware have given rise to the development of automated data mining techniques for new malware. See [4].

The other type of detection method used is behavior-based detection. In this type of detection, the behavior of the system when something malicious happens is taken into consideration. When such an action is detected it is then flagged off as malicious and then separated and scrutinized further. A good example could be to monitor the system calls and functions that become active when malware is executed



in the system. A pattern can then be established which can give a general trend of the behavior of malware when infecting a system. According to [5] the sequence of a system call is reliable and can be used as a reliable method of detecting malicious software. This method is rooted in the fact that however, the appearance of the malware may be, it will still behave badly for it to accomplish its purposes in infecting the computer.

Lastly, we have Anomaly-based detection. In this type of detection, the detector uses its knowledge of normal behavior to decide whether a program is malicious or not. It has some kind of rule set of the normal system behavior and thus uses that knowledge to identify the unusual or rather anomalous behavior.

### **Current Tools for Bypassing Anti-Viruses**

Many techniques are used nowadays to bypass Anti-viruses. Some were made for good purposes like penetration testing and others were specifically made for malicious purposes. In this section, I will go through a number of the tools that have come to light in bypassing Anti-viruses.

The tools are described in the sections that follow. This list is not exhaustive but tries to get the most common tools used at the moment by looking at resources from a variety of sources including hacking reports.

#### **Marble**

This tool was published by wiki leaks as part of the tools that were used by a governmental agency to bypass malware detection. It uses the Marble framework and according to the report it is used to obfuscate or scramble malware code so that Anti-Virus firms cannot understand the code. This framework included a de-obfuscator to reverse the code. Without any relevant academic publication, a comprehensive list of the tools released by WikiLeaks can be found on the WikiLeaks site [6].

## **Grasshopper**

Grasshopper was also part of the other tools that were revealed by wiki leaks and it is used to build customized and persistent malware payloads for Microsoft Windows Operating systems. This tool was developed to avoid anti-malware detection [7].

## **HIVE**

This tool also forms part of the set of tools released by WikiLeaks. This tool according to WikiLeaks was used by the American Central Intelligence Agency (CIA) and it had the capability of developing a back-end infrastructure with a public-facing https interface. This interface was used by the CIA to transfer information from target desktops computers and machines back to the CIA. These devices would then be open to receive further commands from the CIA operators to execute specific tasks.

## **Anti-Virus Evasion Tool (AVET)**

The Anti-Virus Evasion tool was developed for making life easier for pen-testers and for experimenting with Anti-Virus evasion techniques. A detailed explanation for its use can be found on GitHub [8]. This tool uses an XOR encryption process for hiding its payload.

## **peCloak.py**

The peCLOak.py is a python script that automates the process of hiding malicious windows executable from Anti-Virus detection. This tool was created as an experiment in Antivirus evasion and the experiment was naturally successful with all AV software under analysis being evaded [9].

## **PowerShell**

Powershell has been a great benefit for windows systems automation. It however also gives hackers leverage. This tool almost gives us access to windows features in a programmatic way. This tool is extendable and it can be used to administrate Active Directory, email systems, SharePoint, and more. It also gives us access to .NET libraries giving it such power in flexibility. These capabilities have also given hackers leverage in hacking the windows systems [10].

## **Veil-Evasion**

Veil-Evasion is a tool that is used to generate payload executables that are used to bypass Anti-Virus software. This tool works in a framework called veil-framework which is written in python. This tool was written by Chris Truncer and the framework consists of two tools: Evasion and Ordnance [11]. Evasion aggregates various techniques into the framework that simplifies management while Ordnance generates the shellcode for supported payloads which are then used to create payloads from known vulnerabilities.

Some key features of Veil-Evasion include:

1. It can integrate third-party tools such as Hyperion (which encrypts an EXE file with AES 128-bit encryption), PEScrambler, and BackDoor Factory
2. Payloads can be generated and seamlessly substituted into all PsExec calls
3. Users can reuse shellcode or implement their encryption method
4. Minimal Python installation to invoke shellcode; it uploads a minimal Python.zip installation and the 7Zip binary. The Python environment is unzipped, invoking the shellcode. Since the only files that interact with the victim are trusted Python libraries and the interpreter, the victim's AV does not detect or alarm on any unusual activity.
5. Veil-Evasion allows testers to use a safe check against VirusTotal. When any payload is created, a SHA1 hash is created and added to hashes.txt, located in the /veil-output directory. Testers can invoke the checkvt script to submit the hashes to VirusTotal, which will check the SHA1 hash values against its malware database. If a Veil-Evasion payload triggers a match, then the tester knows that it

may be detected by the target system. If it does not trigger a match, then the exploit payload will bypass the antivirus software [11].

A standalone payload in the veil has options that make it work well. It will be important to note that the only files that interact with the victim are trusted python libraries and the interpreter. In this scenario, the victims' AV does not detect any unusual activity. The Set backdoor configures the victims' registry to launch the RDP sticky keys backdoor.

Veil-Evasion uses a safe check against VirusTotal which is a free online tool for checking malicious codes by comparing their signatures with existing signatures in their database. Testers can invoke a checkvt script to submit their hashes to VirusTotal which will check the SHA1 values against its malware database.

## **Shellter**

Shelter is a shellcode injection tool. This shellcode can be something else that is already generated. The full features of the program can be found on the shelter website [12].

One of the notable features of Shelter is its ability to analyze the flow of execution in the legitimate program and place the shellcode in a natural point in the flow. This gives it a huge advantage because there is not a sudden redirection to somewhere else in the code or a weird memory request, like one may see in a non-dynamically-infected executable [13]. This makes the code look like nothing was injected into it. So it gives the appearance of doing what it was always intended to do. Shelter incorporates shellcode into the natural flow of execution in such an imperceptible way that makes it almost impossible to detect. One of the malware used in this research is a manually made malware that injects code a legitimate windows application for internet relay chat.

## **MsfVenom**

Msfvenom is an exploit packing tool. It comes with the Metasploit framework which will be explained in the section that follows. This tool can build everything from simple exploits to complex exploits. These exploits contain code that is used to obfuscate/hide the exploits that are used to bypass Anti-Viruses. According to [14], this tool is the de-facto tool in the Metasploit framework to create and encode various payloads.

## **Metasploit**

The Metasploit Framework is an open-source tool found inside Kali Linux distribution. It can be used for vulnerability analysis and penetration testing. It was created by HD Moore in 2003 using the Perl language and later it was modified using the Ruby language [15]. This framework can help you write, test, and execute exploit code. It can be summed up as a collection of commonly used tools that provide a complete environment for penetration testing [16]. The difference between traditional and modern malware is shown in Figure 1 [19].

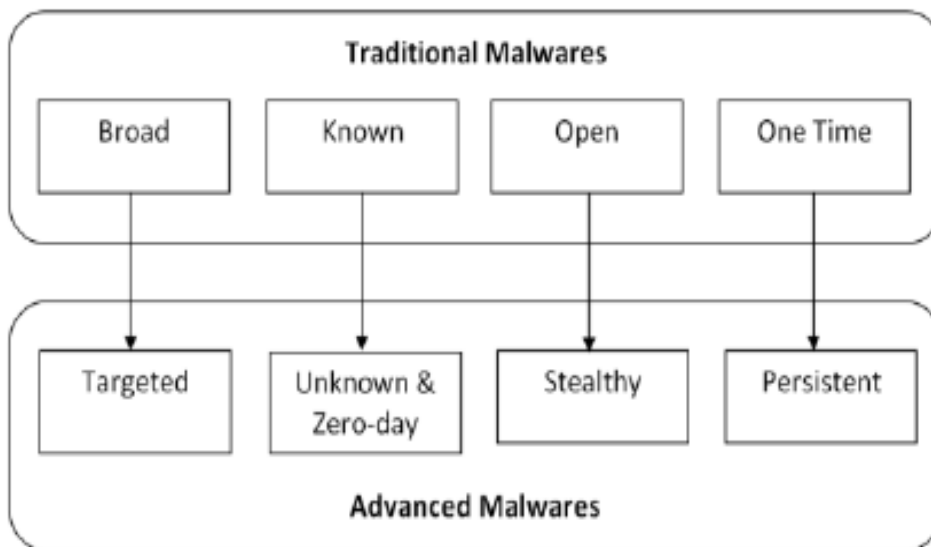


Figure 1. Traditional vs Modern Malware

### Techniques behind the Tools

In this subsection, I will go in detail as to what techniques are used to make the aforementioned tools effective in bypassing Anti-viruses. These techniques are mostly what shows up as the characteristics of the malware. Blackhat USA has done some vast analysis of malware which reveals most of the characteristics in the empirical data that they have provided [17]. It will be important to note at this point that some tools employ more than one technique or rather have many options that use different techniques to accomplish their goal.

### Obfuscation

From ancient times obfuscation has been used to hide the obvious meaning of something. Obfuscation generally refers to the process of hiding or changing the structure of something so that its intent or appearance is not obvious. It can be used for good purposes but it can also be used for bad reasons. [18], defines obfuscation as a term of art that describes a set of techniques used to evade antivirus products that rely heavily on signatures. In this section, I shall focus on how obfuscation has been

used in aiding the hiding of Malware Detection by Anti-viruses. There is a big difference between traditional malware and modern malware. [19]. Below is a good summary of the two [19]:

This difference has made obfuscated malware to be harder to detect. The nature of the new malware has posed challenges even to machine learning techniques that are used for malware detection [20]. According to [21] code obfuscation changes malware syntax but not its intended behavior. This behavior has to be preserved.

Obfuscation techniques can be divided into two categories; anti-static and anti-dynamic analysis techniques. Anti-static obfuscation techniques hide the malware in the light of static malware detections techniques aforementioned. Static analysis involves analyzing the malware without executing it. Anti-dynamic analysis techniques hide their activities in the light of dynamic/heuristic malware detection techniques.

Dynamic analysis involves running the malware in a controlled environment and studying its behavior. The activities that are monitored during this process include things like the creation and deletion of new files, new log entries, registry entries, URL accessed, and data transmitted. Obfuscation techniques used include dead code insertion, register re-assignment, subroutine re-ordering, instruction substitution, code transposition, and code integration [22]. Others include packers that compress or “pack” a malware program and crypters which encrypt a malware or parts of a malware.

Some malware obfuscation techniques transform the malware binaries to self-compressed and uniquely structured binary files. This is designed to resist reverse engineering making static analysis to be very expensive and unreliable [23].

## **Polymorphism**

This technique is very hard to detect because of the way it manifests itself. This is the ability for malware to take many forms. This poses a huge challenge by making it harder to make a signature of malware that can be used by anti-malware solutions. According to [24] polymorphism was initially adopted by malware writers to counter the simple string searches that Anti-Virus engines employed to detect malware. Some of the methods used for polymorphism are listed below:

**Garbage code insertion.** This means useless code is inserted into the malware after infection. It's the simplest form of code obfuscation done by inserting NOPs (No Operation Performed) [25]. This method aims to make it hard to compare the existing code with the previous code.

**Instruction substitution techniques.** This involves the technique of replacing the code with an equivalent but a different one. This technique according to [26] evades most malware detection techniques.

**Code-transposition.** This technique changes the execution order by using jumps. It changes the program structure by reordering the program instruction or flow without changing the execution flow. This can be done on a single instruction level of a code of block [27].

**Register-reassignment.** This technique simply re-arranges the registers. The register of the code is thus replaced by unused registers while the program code and its behaviors remaining the same [25].

**Tools using polymorphism.** One of the tools that use polymorphism is veil evasion. Veil evasion has an encoder that uses a polymorphic XOR additive feedback encoding against a 4-byte key. So, it changes its shape using an XOR encrypting scheme. This encoder can be made handier by iterating it several times. The iteration



must however be handled carefully as each additional iteration increased the size of the payload. At the time of its implementation, it was ranked as “excellent” [28] by Metasploit.

One other payload generator that uses polymorphism is Shellter. Shellter has a threat context-aware polymorphic engine. The user can also use a custom polymorphic code of their own.

**Countering effects of polymorphism.** Several methods have been employed to try to study and counter the effects of polymorphic malware. One of them is to compare the changes that polymorphic malware exhibits to genetic changes that take place biologically. These changes are similar to mutations of biological sequences that occur over successive generations [29].

According to [30] network-based security is needed to evade zero-day polymorphic Malware as most host-based securities that are implemented cannot detect a well-crafted attack. Iyhothi Kumar [31] suggested a framework that uses machine learning to develop a defense system against polymorphic malware. This experiment used the *shikata-ga-nai* encoder found in the veil framework to generate the polymorphic malware.

Fractal analysis which is the study of shapes or patterns in data that are not easily described by simple geometry has been shown [32] to be a promising domain in coping with next-generation threats that obfuscate their signatures and also learn to depict themselves as legitimate processes.

## **Encryption**

There are two main reasons for doing malware encryption. The first one is to prevent or make it hard for the malware to be detected and the other one is to make it hard for someone to analyze the activities of the malware. To encrypt a malware one

needs some components. The actual malware which is encrypted, a module to perform the encryption/decryption and a key. Encryption has been used in many places to produce malware that has been successful in infecting computers. CryptoLocker was one of the most widely known viruses that were found early. This virus was discovered in 2013 and in addition to encrypting sensitive files, it would communicate with the command and control server and even take a screenshot off the infected machine [33].

Several modern tools use encryption to hide their malware from being detected by Anti-Viruses. Veil evasion uses an encryptor called Hyperion. This executable encryptor uses Advanced Encryption Standard (AES) a current industry standard for encryption. After encrypting the executable Hyperion throws away the encryption keys. So when the executable runs it brute forces the encryption key to decrypt itself back to the original executable [34].

Advanced Encryption Standard is considered a pretty strong encryption standard. Hyperion however uses brute force to get the encryption key and to cater to this. Hyperion greatly reduces the possible keyspace for the encryption key and this fact should be taken into consideration in trying to analyze malware that is generated by Hyperion.

Powershell has an amazing built-in remoting system. This allows users to handle most remoting tasks in many different kinds of configurations. There are many options available for authentication namely: Basic, Digest, Kerberos, Negotiate, and CredSSP. According to [35], in all the mentioned Authentication types, the payload for the message you send is encrypted directly by the remoting protocol except the Basic authentication.

PowerShell gives access to almost all of the windows features in a programmatic way. It also gives access to the .NET libraries from a scripting standpoint, making it one of the most flexible tools you can use in a windows environment. So anything we can write in .NET we can write in PowerShell. This is a very interesting feature because it means that we can go beyond basic scripting and interact with kernel functions and more. This gives additional flexibility that would normally require the use of separate programs. Powershell scripts can be loaded and run from memory without ever writing files to the hard drive [36]. This allows PowerShell not to leave file-based evidence making it hard for detection from file-based antivirus protection.

One other feature of PowerShell that is worth noting here is its ability to use Internet Explorer options and so things like proxy support are built into PowerShell. With this, we can use built-in web libraries to load code remotely without having to download code to the target system. This allows attackers to be stealthier since the file-system will not be able to show the pulls from the website [10].

Modules in PowerShell are very portable and this makes it possible for them to be loaded in a variety of ways. This gives us the ability to load system install modules and modules in other locations. This feature gives a very conducive environment for hackers since one of their main aims is to leave as few traces as possible of their actions. Items that are frequently used can be left on the SMB share or even on a website and then referenced from there. Bypassing Anti-Viruses is easier because code can be obfuscated and decoded on the fly.

## **Encoding**

An encoder takes input/shellcode and transforms it. The encoding is like adding a layer of shells around the payload to make it hard to be detected by the Anti-

Virus. This changes the way the code looks without changing the underlying functionality. The MSFvenom tool has the option of encoding the payload. MSFvenom has different options for encoders that you can choose from and how many times you can do the encoding.

Shellter uses encoding as one of its methods to try to make its payloads undetectable by Anti-Virus programs. Shellter 4.0 provided its dynamic encoder. The encoding engine will apply a random amount of XOR, ADD, SUB, NOT operations and it will generate a decoder each time based on the chosen operations. The sequence of these operations is also randomly chosen. The use of registers is also randomized to provide a more dynamic output. According to [28], the majority of antiviruses will not be able to identify the malicious executable, depending on how the attackers re-encode the endless number of signatures.

MSFvenom utility has the option of encoding payloads and iterating them several times to try to make them stealthier. The MSFvenom framework has different polymorphic encoders to stimulate polymorphic malware [31]. One of the encoders it uses is the *Shikata\_ga\_nai* encoder which has been mentioned before. With this encoder even the decoder stub is polymorphic.

Metasploit has a variety of encoders that can also be used in addition to the other encoders mentioned here. To make the payload stealthier multiple encoders can be combined and used together. The summary of the tools and techniques employed in bypassing anti-virus software is shown in Table 1.

*Table 1. Summary of Tools and Techniques Employed in Bypassing Anti-Virus Software*

	Obfuscation	Polymorphism	Encryption	Encoding	Injection
Veil-Evasion		✓	✓	✓	
PowerShell	✓	✓	✓	✓	
peCloak.py				✓	
Shellter		✓			✓
Metasploit				✓	
MSFvenom			✓		
Marble	✓				
Grasshopper		✓			
HIVE	✓				

### **Sysmon: A Compensatory Tool**

#### **Sysmon Capabilities**

Sysmon (System monitor) is a Windows system service that also acts as a driver. The system monitor tool logs the windows events. This tool was developed by Mark Russinovich and Thomas Garnier [37]. It is especially important to note that this tool logs additional activity to the event log such as network connections, running processes, and file changes. Mark Russinovich wrote this tool to track the potentially malicious activity on individual computers and across the network [38].

Sysmon is configured as a boot-start driver and it begins capturing information early in the boot [38]. This is very key as it helps track system events at a very early stage. This tool compliments some of the Windows' shortfalls. In the normal Windows machines, network connection information is simultaneously too limited and verbose [39]. Sysmon provides more information by giving details that can help track malicious connections. This tool can also help forensics to trace intruder activity across the network [39]. It is also important to note that Sysmon was written for use within the Microsoft corporate network.

Sysmon runs silently in the background and the event viewer records its findings. The advantages of Sysmon are as follows:

1. It can log the process creation for both current and parent processes
2. It can record the hash of process image files using SHA1 (default), MD5 or SHA256
3. It has a unique GUID in the process to create events. This allows for a proper correlation of events. The windows events log differs in that it can re-use process IDs.
4. For a network connection, it can provide source processes, IP addresses, port numbers, and hostnames.
5. It can detect changes in the file creation time. This is key in determining when a file was created. Malicious users can change the file creation time to bypass system security.
6. It can generate events even in the early boot process. This helps in detecting kernel-mode malware.

Sysmon logs different events and gives them unique Ids that can be used to study and identify the activity that takes place within the system and a specific period. Below is a summary of the IDs and the categories that they represent:

### **Events to Investigate in Sysmon Logs**

Mark Russinovich gives a good summary in [40] that gives indicators of events that one needs to check while analyzing the Sysmon logs. I will highlight them below as processes that should be investigated when they show up in the logs:

1. Processes that have no icon
2. Processes that have no description of the company name
3. Processes that have unsigned Microsoft images
4. Processes that live in windows directory of user profile
5. Processes that are packed
6. Processes that have strange URLs in their strings
7. Processes that have open TCP endpoints

## 8. Processes that host suspicious DLLs or services

Figure 2 shows the Sysmon Events.

Category	Event ID	Category	Event ID
Sysmon Service Status Changed	0	Process Access	10
Process Create	1	File Create	11
File Creation Time Changed	2	Registry Object CreateDelete	12
Network Connection	3	Registry Value Create	13
Sysmon Service State Change	4	Registry Object Rename	14
Process Terminated	5	File Create Stream Hash	15
Driver Loaded	6	Sysmon Configuration Changed	16
Image Loaded	7	Pipe Created	17
CreateRemoteThread	8	Pipe Connected	18
RawAccessRead	9	Error	255

Figure 2. Sysmon Events [39]

The strategy commonly used by hackers that can be captured by Sysmon is as follows: Attackers can change the file timestamps in an attempt to cover their tracks [41]. If configured well Sysmon can capture the file creation time change that can give a good signal of something bad.

## CHAPTER 3

### METHODOLOGY

The literature review identified numerous techniques that are used to bypass Anti-viruses. This has led to attacks that bypass Anti-viruses. This research intends to try to close the gap that leads to the attacks. This gap led to the research question “Why do Anti-Viruses fail to protect from Malware?” An appropriate research methodology will go a long way in answering the research question. In the quest to answer the research question an applied experimentation methodology was selected. In this chapter, the model that guided the execution of the study is well outlined in detail with its implementation requirements.

#### **Research Methodology Justification**

At the core of this research was the quest to study Anti-viruses and come up with ways that any known weakness can be compensated. This led to the search of a research methodology that studies an existing system and tries to compensate for its weaknesses. Applied experimentation that is largely geared toward understanding the behavior of a system best fits this scenario. According to [42] Applied experimentation is the “process of evaluating performance or effectiveness of an engineered system in solving a problem under rigorously controlled systems.”

#### **Overview**

Atomic test cases of real malware are considered in determining their behavior with the view of detecting them. This will be viewed in the light of the events that they generate in windows events that are further channeled to Sysmon for further



investigation and analysis. Sufficient coverage of the problem space was considered by looking at freely available portals for checking the efficiency of Anti-viruses against malware. Seven malware was considered for this benchmark. Five of them came from the freely available malware bank known as theZoo.

It consists of a depository of live malware that is freely available for research purposes [43]. Two of them were manually made. The process used to make the manual ones will be given later. It is important to note at this point that the five that came from theZoo are detectable by most anti-viruses but the two are not. The overall behavior of all of them was noted. This is key in determining whether the results are consistent when they go through Sysmon. The events triggered by malware whether detectable or not will give insight to the capability of Sysmon to detect malware behavior through the various events triggered.

The two manually produced malware that bypasses the antivirus will be tested through freely available online scanners. One of them is the virus total. Virustotal is a free online, virus, malware, and URL scanner [44]. Other sites used for scanning malware include nodistribute.com and antiscan.com.

The tools that were used to generate payloads during the tests were from Kali Linux distribution. I used the windows 7 operating system that was installed on a Virtual Machine. The windows system was used to check the events that are triggered when malware is installed and running in the system.

## **Objectives**

The clear objectives of this study were as follows:

1. To create awareness that Anti-Viruses can be bypassed by malware and give insight into the tools and techniques that are employed.
2. Identify windows event analysis that can help to identify malware or malware patterns

3. Give scientific recommendations as to what can be done to increase the efficiency of the Anti-viruses in a compensatory approach, to increase the overall security of the enterprise.

The tools below have been specifically selected with the aim and view of meeting the objectives.

### **Kali Linux Virtual Environment**

Kali Linux is a popular Linux distribution. It's mostly used for Penetration testing but it can also be used to generate payloads that can be used for other purposes. It has over 600 security tools built into the distribution. It's also open source and therefore it can be used for free as long as you know how to use it. I used some of the tools especially in the area of payload generation.

This distribution of Linux is especially geared towards people who want to engage in security. It can be of interest to anyone who wants to engage in security testing, exploit development, reverse engineering, or even digital forensics. For my purpose, I installed Kali Linux in a virtual environment hosted by Ubuntu Linux distribution. I used a Virtual box to run the Kali Linux.

### **Metasploit Framework**

Metasploit is a very widely used penetration testing tool that is part of the Kali Linux distribution. It's used by both attackers and defenders. It has a couple of libraries and modules. At the heart of Metasploit are three libraries namely REX, MSF CORE, and MSF BASE. REX handles most of the core functions like setting up sockets, formatting, and other raw functions. MSF CORE provides the underlying API and the actual core that describes the framework. MSF BASE provides friendly API support to modules [45].

There are many modules in Metasploit and these modules differ in functionality. Some modules are used to create access channels to exploited systems and we have auxiliary modules that are used to carry out operations such as information gathering, fingerprinting, fuzzing an application, and logging to various services. Two of them will be of major interest in attaining the objectives. The payload module will be employed in creating a meterpreter shell that will give us access to the target machine and also help us to maintain access to the exploited machine. The auxiliary module will also be used for information gathering from the exploited machine.

### **Windows 7 Virtual Machine**

For this research, I used windows 7 virtual machine. The windows virtual machine resided in the same virtual box and also hosted Kali Linux as a virtual machine. The advantage of using windows in this environment is the fact that someone can make snapshots of the same windows machine and use it for different test purposes. In this lab, I first made the initial snapshot that had windows 7 with free Avast Anti-Virus installed.

This snapshot was also installed with Sysmon and Winlogbeat which will be discussed in the sections to follow. Before the introduction of the payload, the Anti-Virus was fully updated. The payloads were then introduced and scanned using the Avast free version of the Anti-Virus. Avast was picked after a careful study of free anti-viruses suggested it to be one of the best [46]. The payloads were also tested online at virus total and antiscan.me. The results were recorded and tabulated.

## Payloads

### Generated Payloads

At the core of this research is creating awareness that Anti-Viruses can be bypassed. Many methods can be used to generate payloads that bypass Anti-Viruses. For this experiment, we used the aforementioned tool Shellter and another one called zirikatu to create the payloads. These payloads were used to bypass a current free Anti-virus that has been fully updated. The payloads were also tested in free to use online sites that test how different Anti-viruses react to different payloads. In total seven payloads were used with five coming from theZoo and two manually made. The Process of generating the two manually made payloads is discussed in the appendix.

### Acquired Payloads

The following payloads were acquired from theZoo and their respective hashes in sha256:

**Somoto payload.** Somoto is a browser hijacker malware. Mostly associated with video applications like FLV I players [47].

sha256:ddf2542dc5ac74a98d5ee9e55497572104d6c880aad9137caf884d10ca5953ce

**Artemis payload.** This malware prevents users from using the computer, run windows registry, or install anti-malware [48].

sha256:834d1dbfab8330ea5f1844f6e905ed0ac19d1033ee9a9f1122ad2051c56783dc

**Dyre payload.** Dyre otherwise known as Dyre Banking Trojan harvests credentials primarily targeting online banking websites [49].

sha256:a6f10947d6c37b62a4c0f5e4d0d32cc826a957c7d1026f316d5651262c4f0b24

**NiVdort payload.** This malware steals victims credentials [50].

sha256:3fbdede25a0eb245357501033b64adcd9380e592f386ef05748ca3d9b42910af

**njRAT payload.** This malware gives a simple backdoor to the victim's machine. It was according to [51], considered the most active network malware in 2017.

sha256:5ff121c57e4a2f2f75e4985660c9666a44b39ef2549b29b3a4d6a1e06e6e3f65

### **ELK (Elasticsearch, Logstash, Kibana) Stack**

The Elastic Stack is a collection of three amazing open-source projects namely Elasticsearch, Logstash, and Kibana. These have been built to work exceptionally well together. Logstash is used to collect and transform logs from different sources. In my setting, the winlogbeat will be working to send the logs to logstash. Elasticsearch as the name suggests is used to search and analyze logs and finally, Kibana is used to visualize and manage the logs by creating fantastic dashboards. We will incorporate all the three in our setting. I will discuss below how we installed them in our lab setting. The host for the stack is ubuntu 18.04.2 LTS

### **Sysmon**

This tool lies at the core of this research in that it can detect changes in the system that have been specially related to malicious activities that are taking place within the system. Sysmon will also be used to gather the logs and send them over to the ELK stack discussed in the section that follows.

### **Winlogbeat**

Winlogbeat is a data shipper that ships windows event logs to Logstash or Elastic search cluster. It can read different windows event logs and forward them

promptly. It can send different types of events like Hardware events, Security Events, System Events, and Application events. This tool can convert raw event data into a structured format that is easy for filtering and aggregation.

### **ElastAlert**

The logs that come to the ELK stack can be numerous and huge. For this, I configured an alert system that can detect particular alerts and send them to us directly. In this research, I used ElastAlert, an open-source Alerting system that can be configured to send specific alerts. This tool can be customized to send many types of Alerts. For my research, I was more interested in the events that are triggered when someone connects remotely to a computer. In the appendix sections, I have explained how I installed and configured ElastAlert to meet the research objectives.

An alert system was implemented that sends alerts to computers and cell phones when a certain event or combination of events is triggered and sent through Sysmon. No capability of this nature was identified while reviewing existing literature.

## CHAPTER 4

### RESULTS AND ANALYSIS

#### **Malware Dataset**

As discussed in Chapter 2 malware infiltration has become a threat to organizations and individuals. This malware has proven to be evasive in spite of the anti-viruses that may be in place, [52]. To achieve relevant and repeatable research results, a malware dataset was needed in this research to demonstrate the effectiveness of the developed system in profiling malware behavior using Sysmon logs. Modern and freely accessible malware was the first consideration that was taken. Secondly manually developed malware was taken into consideration to cover the problem space. Zero-day attacks have wreaked havoc on systems that were otherwise deemed secure [53].

The five malware samples that were selected for this research had a significant impact when they were effected and previous data in this research has indicated their vast exploitations. The two that were manually made served the purpose of trying to identify malware behavior that can bypass legitimate anti-viruses. In my case avast free anti-virus was chosen. Avast Free Antivirus won the product of the year award in 2018 [54]. The two manually made malware we able to bypass a fully updated free Avast Antivirus. For consistency, a snapshot of the fully updated Anti-virus was taken and used for the other malware.

The choice of publicly available malware ensured that this research can be repeated elsewhere and the process of producing the two manually produced malware

is well documented in the appendix for the same purpose. With these others can easily repeat, verify, and expand on the results of this research. The use of publicly available malware that has affected real organizations and the use of malware that was able to bypass a widely known and used anti-virus makes this research applicable and relevant to organizations.

### **Analysis of the Data**

Sysmon event logging was used to analyze malware behavior in this research. The literature review identified a gap that led to the research question. It is in the quest to answer the research question that Sysmon events logging was chosen to identify malware behavior that may not be picked by anti-viruses. According to [55] by analyzing Sysmon logs it is possible to detect threats that otherwise would go undetected by traditional network intrusion detection systems and firewalls such as network traffic. Though only avast free anti-virus was considered other freely available malware scanners were used to show how the same malware samples would fare in other Anti-virus environments. These free online scanning tools confirmed the findings of our manually made malware that we're able to bypass Avast free anti-virus. The findings are shown below.

#### **mir755.exe Malware**

The first of the manually made malware that was made using Shellter recorded the following scan results.

Scan results from VirusTotal

sha256:9b382b007e5d61d5e6c9a2378f207fcd6f329cd551975d67114472ba4561b190

Scan results from Antiscan.me



## **power.exe Malware**

The second manually made malware that was made using Zirikatu recorded the following scan results:

Scan results from VirusTotal

sha256:da9121e03de8cb6374c77fac5000527c601ed5cf0a21961e2d623415c1809142

Scan results from Antiscan.me

The above scan engines confirmed that the manually made malware were able to bypass avast free anti-virus.

The next set of results was the Sysmon events that were triggered when all the seven malware were executed in different instances of the same snapshot of a windows 7 virtual machine.

Figures 3 through 7 show the events that were triggered by each malware. From the figures, the Sysmon event that was created by all the payloads is the process creation event and this event is triggered when there is a new process created in the system. In and of itself, it is not sufficient proof that there is malware in the system but the task was also to see the behavior patterns that are triggered near or at the same time this process is triggered. It is however important to note that every malware that was tested started by creating a process. This begs for a careful investigation of what happens before, after, or even during that process or any noticeable behavior related to the process.

The next Sysmon event that was noticed was event number three which is triggered when there is a network connection noticed. Additional log information from the process revealed that the IP address of the remote Machine which acted as a command and control center was also revealed. This is very essential when tracing malicious network connections. These two payloads were the ones that we're able to

bypass anti-virus software. These payloads were used to make remote connections to the target machine. It is worth noting that malware that needs remote connections to trigger Sysmon event three.

Sysmon event number five which is triggered when a process terminates. It's interesting to note that all the tested payloads at some point had their processes terminated.

Six of the seven payloads triggered Sysmon event number eleven which is triggered when a file is created. This event is very important when hunting for malware because it can help to monitor autostart locations like the startup folder as well as temporary and download directories which are common places for malware drops during initial infection [56]. This event, therefore, has a high correlation to malware, and the events that occur concerning this one and their sequences and patterns of occurrence beg for investigation.

One malware Artemis triggered Sysmon event number twelve which is the Registry Event (object create and delete) which can be useful for monitoring changes to registry autostart locations or specific malware registry modifications. Registry entries according to [57] can be useful in even finding the possible tools that were executed during an incident investigation.

The other event that was triggered by one malware was the FileCreateStreamHash event number fifteen. This event logs when a named file stream is created. The event can also be used in malware or malware behavior because it can track malware variants that drop their executables or configuration settings via browser downloads. It captures that based on the browser attaching a Zone.Identifier "mark of the web" stream.

Community Score

10 engines detected this file

9b382b007e5d61d5e6c9a2378f207fd6f329cd551975d67114472ba4561b190

mir755-2.exe

nsis overlay peexe

2.67 MB  
Size


2019-09-08 15:36:53 UTC  
a moment ago

DETECTION	DETAILS	BEHAVIOR	COMMUNITY
SecureAge APEX	Malicious	CrowdStrike Falcon	Win/malicious_confidence_7...
Cylance	Unsafe	Endgame	Malicious (high Confidence)
Ikarus	Trojan.Win32.Rozena	Kaspersky	HEUR:Trojan.Win32.Generic
McAfee-GW-Edition	BehavesLike.Win32.Dropper...	Microsoft	Trojan:Win32/Swrort.A
Sophos AV	Mal/Shellter-AF	ZoneAlarm by Check Point	HEUR:Trojan.Win32.Generic
Acronis	Undetected	Ad-Aware	Undetected
AegisLab	Undetected	AhnLab-V3	Undetected
Alibaba	Undetected	ALYac	Undetected
Antiy-AVL	Undetected	Arcabit	Undetected
Avast	Undetected	Avast-Mobile	Undetected
AVG	Undetected	Avira (no cloud)	Undetected
Baidu	Undetected	BitDefender	Undetected
Bkav	Undetected	CAT-QuickHeal	Undetected
ClamAV	Undetected	CMC	Undetected
Comodo	Undetected	Cybereason	Undetected
Cyren	Undetected	DrWeb	Undetected
eGambit	Undetected	Emsisoft	Undetected
eScan	Undetected	ESET-NOD32	Undetected
F-Prot	Undetected	F-Secure	Undetected
FireEye	Undetected	Fortinet	Undetected
GData	Undetected	Jiangmin	Undetected
K7AntiVirus	Undetected	K7GW	Undetected
Kingsoft	Undetected	Malwarebytes	Undetected
MAX	Undetected	McAfee	Undetected
NANO-Antivirus	Undetected	Palo Alto Networks	Undetected
Panda	Undetected	Qihoo-360	Undetected
Rising	Undetected	SentinelOne (Static ML)	Undetected
Sophos ML	Undetected	SUPERAntiSpyware	Undetected
Symantec	Undetected	TACHYON	Undetected
Tencent	Undetected	Trapmine	Undetected

Figure 3. VirusTotal for mir755.exe

<b>Filename</b> mirc755-2.exe	<b>MDS</b> e30439b90e0b1909222dcb6edb43a288
<b>★ Detected by</b> 5/26	<b>Scan Date</b> 08-09-2019 15:26:51


Your file has been scanned with 26 different antivirus software (**no results have been distributed**). The results of the scans has been provided below in alphabetical order.



NOTICE: Some AV can work unstably and scan take more time.

Ad-Aware Antivirus: <b>Clean</b>	Fortinet: <b>Clean</b>
AhnLab V3 Internet Security: <b>Clean</b>	F-Secure: <b>Clean</b>
Alyac Internet Security: <b>Clean</b>	IKARUS: <b>Trojan.Win32.Rozena</b>
Avast: <b>Clean</b>	Kaspersky: <b>HEUR:Trojan.Win32.Generic</b>
AVG: <b>Clean</b>	McAfee: <b>Clean</b>
Avira: <b>Clean</b>	Malwarebytes: <b>Clean</b>
BitDefender: <b>Clean</b>	Panda Antivirus: <b>Clean</b>
BullGuard: <b>Clean</b>	Sophos: <b>Mal/Shellter-AF</b>
ClamAV: <b>Clean</b>	Trend Micro Internet Security: <b>Clean</b>
Comodo Antivirus: <b>Clean</b>	Webroot SecureAnywhere: <b>Clean</b>
DrWeb: <b>Clean</b>	Windows 10 Defender: <b>Trojan:Win32/Swrort.A</b>

Figure 4. Antiscan.me for mirc755.exe



Community Score

**34 engines detected this file**


da9121e03de8cb6374c77fac5000527c601ed5cf0a21961e2d623415c1809f42

power.exe

assembly overlay peexe

144.55 KB Size

2019-09-08 15:51:18 UTC  
10 days ago




DETECTION	DETAILS	RELATIONS	BEHAVIOR	COMMUNITY
Ad-Aware	Gen:Variant.Razy.149220	AhnLab-V3	Trojan/Win32.Dynamer.R199...	
ALYac	Gen:Variant.Razy.149220	SecureAge APEX	Malicious	
Arcabit	Trojan.Razy.D246E4	Avira (no cloud)	HEUR/AGEN.1042533	
BitDefender	Gen:Variant.Razy.149220	ClamAV	Win.Malware.Razy-6915301-0	
Comodo	TrojWare.MSIL.Tiny.K@7cyd4s	CrowdStrike Falcon	Win/malicious_confidence_1...	
Cybereason	Malicious.ab6820	DrWeb	BackDoor.Siggen2.2068	
Emsisoft	Gen:Variant.Razy.149220 (B)	Endgame	Malicious (high Confidence)	
eScan	Gen:Variant.Razy.149220	ESET-NOD32	A Variant Of MSIL/Tiny.F	
F-Secure	Heuristic.HEUR/AGEN.1042...	FireEye	Generic.mg.96aef85ab6820...	
Fortinet	MSIL/Tiny.Filtr	GData	Gen:Variant.Razy.149220	
Ikarus	Trojan.MSIL.Tiny	Kaspersky	HEUR:Trojan.Win32.Generic	
Malwarebytes	Trojan.Agent	MAX	Malware (ai Score=81)	
McAfee	GenericRXEA-QF196AEF85...	McAfee-GW-Edition	GenericRXEA-QF196AEF85...	
SentinelOne (Static ML)	DFI - Malicious PE	Sophos AV	Troj/Tiny-DI	
Sophos ML	Heuristic	Symantec	ML.Attribute.HighConfidence	
Trapmine	Malicious.moderate.ml.score	TrendMicro	HT_TINY_GK070020.UVPM	
TrendMicro-HouseCall	HT_TINY_GK070020.UVPM	ZoneAlarm by Check Point	HEUR:Trojan.Win32.Generic	
Acronis	Undetected	AegisLab	Undetected	
Alibaba	Undetected	Antiy-AVL	Undetected	
Avast	Undetected	Avast-Mobile	Undetected	
AVG	Undetected	Baidu	Undetected	
Bkav	Undetected	CAT-QuickHeal	Undetected	
CMC	Undetected	Cylance	Undetected	
Cyran	Undetected	eGambit	Undetected	
F-Prot	Undetected	Jiangmin	Undetected	
K7AntiVirus	Undetected	K7GW	Undetected	
Kingsoft	Undetected	MaxSecure	Undetected	

Figure 5. VirusTotal for power.exe

<b>Filename</b> power.exe	<b>MDS</b> 96aef85ab6820653008e59b03fb6c533
<b>★ Detected by</b> 15/26	<b>Scan Date</b> 19-09-2019 10:18:34

Your file has been scanned with 26 different antivirus software (**no results have been distributed**). The results of the scans has been provided below in alphabetical order.



NOTICE: Some AV can work unstably and scan take more time.

Ad-Aware Antivirus: <b>Gen:Variant.Razy.149220</b>	Fortinet: <b>MSIL/Tiny.Fltr</b>
AhnLab V3 Internet Security: <b>Trojan/Win32.Dynamer.R199417</b>	F-Secure: <b>Heuristic.HEUR/AGEN.1042533</b>
Alyac Internet Security: <b>Gen:Variant.Razy.149220</b>	IKARUS: <b>Trojan.MSIL.Tiny</b>
Avast: <b>Clean</b>	Kaspersky: <b>HEUR:Trojan.Win32.Generic</b>
AVG: <b>Clean</b>	McAfee: <b>GenericRXEA-QF96AEF85AB682</b>
Avira: <b>HEUR/AGEN.1042533</b>	Malwarebytes: <b>Clean</b>
BitDefender: <b>Clean</b>	Panda Antivirus: <b>Clean</b>
BullGuard: <b>Clean</b>	Sophos: <b>Troj/Tiny-DI</b>
ClamAV: <b>Win.Malware.Razy-6915301-0</b>	Trend Micro Internet Security: <b>Clean</b>
Comodo Antivirus: <b>TrojWare.MSIL.Tiny.K@445021948</b>	Webroot SecureAnywhere: <b>Clean</b>
	Windows 10 Defender: <b>Clean</b>

Figure 6. Antiscan for power.exe

PAYLOAD STATISTICS THROUGH SYSMON EVENT LOGGING		
SYSMON EVENT	DESCRIPTION	SAMPLES THAT TRIGGERED EVENT
1	Process Creation	mirco755,power.exe,Nivdort,Dyre,Somoto,Artemis,njRAT
2	Process Changed File Creation Time	
3	Network Connection	mirco755,power.exe,
4	Sysmon Service State Changed	
5	Process Terminated	mirco755,Nivdort,Dyre,Somoto,Artemis,njRAT,power.exe
6	Driver Loaded	
7	Image Loaded	
8	CreateRemoteThread	Dyre
9	RawAccessRead	
10	ProcessAccess	
11	FileCreate	power.exe,Nivdort,Dyre,Somoto,Artemis
12	Registry Event(Object create and delete)	
13	Registry Event(Value Set)	Artemis,
14	Registry Event(Key and Value Rename)	
15	FileCreateStreamHash	Nivdort,
17	PipeEvent(Pipe Created)	

Figure 7. Events Triggered by Each Malware

### Event Execution Sequence and Timing Analysis

The events that were triggered by Sysmon were further investigated as to the sequence of the events and time intervals that took place between one event and another one. It is important to note at this point that the time a malware takes for execution is very key as this can determine whether a system will be compromised by the malware or not.

According to [58] by the time malware is detected by the scanning software, some damage could have taken place. The next step was to investigate the timings between the various events that were triggered by the malware that was tested. In the appendix, I have added graphs for the timing for each malware. Tabulated results can be found in the list of figures:

For the njRAT malware, a process was created at the execution of the malware, and another process was created after 27 milliseconds. Two files were created after one minute and 752 milliseconds which was shortly followed by the start of another

process in 98 milliseconds. The process of process creation and file creation followed in two successions with the time difference being in milliseconds. Finally, there was a process termination in 10 minutes. One quick observation is that the malware process as it works in a system can be so fast that in less than one second a lot could have happened.

The event execution in the next malware which was Nivdort created a similar pattern where there were two process creations, two file creations followed by two process creations and finally the process was terminated. The process creations and file creations all occurred in a matter of fewer than two seconds with the intervals between each particular event being in milliseconds. The process was also terminated 18 seconds after the execution of the last event which was process creation.

The next malware which was from Dyre showed a pattern of process creation and termination in pairs that all occurred within the same second with the differences being milliseconds. The somoto malware started by creating files followed by process creation, network connection, and process termination. The first three events all occurred in the same second with the difference being in milliseconds. Another network connection was made after 10 seconds and the process terminated after 12 seconds.

Artemis malware exhibited a unique feature of first creating a file stream followed by process creation in a matter of milliseconds and then creating 12 files in the same second which was followed by the creation of 62 processes within the same minute with the difference between each of the created processes being less than a second. Towards the end, the registry value was set twice and then followed by process termination.



The last two malware that was analyzed was the ones that were manually made. These were the ones that successfully bypassed the free avast anti-virus software. The first one was the power.exe that was made by the help of Zirikatu and it was supposed to provide a remote connection to a target machine. There was a sequence of process creations followed by network connections and process termination that all occurred in a matter of about 3 seconds with the time interval between the individual processes being less than a second or a few seconds. The other one was mirc755 which was made with the help of shellter. This malware exhibited a similar pattern of process creation followed by a network connection and then a process termination. The interval between the events was a matter of seconds with the highest being 12 seconds and the others being in milliseconds.

From the samples that were tested, it was evident that malware triggers different processes when in operation and a noticeable feature was that this can take place in a matter of seconds. The following patterns were observed:

1. Process creation followed by some events within a short period may be an indication of malware in a system.
2. 3 out of seven malware that was tested showed a correlation between process creation and file creation within a short period and then a process termination at the end.
3. One malware showed a series of pairs of process creation and termination in a short period.
4. One malware showed a series of process creation that followed a file stream creation and file creation. 62 processes were consecutively created.
5. Two malware that was targeted at bypassing anti-viruses and connecting to a command and control center exhibited a pattern of process creation and network connection within a short period followed by process termination.

Based on the 7 malware that was tested the following can be deduced:

1. Events that combine process creation with file creation or network connections and then process termination in a short period could be an indication of malware in the system.

2. Events that produce file streams and file creations followed by a multiple/numerous process creation and then termination within a short period could be an indication of malware in the system.
3. Events that produce multiple process creations and terminations within a short period could be an indication of malware in the system.

### **Mitre ATT&CK**

The next step was to analyze the events using a threat model. In my case, I chose to use the Mitre ATT&CK model. It describes the actions an adversary takes in an enterprise. This threat model was informed by credible sources like public intelligence reporting, penetration testing, red teaming, and security research [59]. This framework also provides valuable mitigation and detection guides on the attack vectors.

Figures 8 to 10 are a summary of some of the events that were triggered and what are the suggested mitigations.

SYSMON EVENT & ID	PAYLOAD	SYSMON LOG DETAILS	MITRE ATT&CK TACTIC	TECHNIQUE USED	SUGGESTED MITIGATION	
<b>NETWORK CONNECTION(3)</b>	mirc755	Destination IP is Kali Machine	Command and Control	polymorphism	Network Intrusion Detection and Prevention	
		Destination port is 4444	Command and Control	Uncommonly used port		
		PE run from Desktop				
	power.exe	PE run from Downloads				
		Destination IP is Kali Machine	Command and Control	Encoding	Network Intrusion Detection Prevention	
		Destination port is 4444	Command and Control	uncommonly used port	Network Intrusion Detection Prevention	

Figure 8. Network Connection

SYSMON EVENT & ID	PAYLOAD	SYSMON LOG DETAILS	MITRE ATT&CK TACTIC	TECHNIQUE USED	SUGGESTED MITIGATION
<b>CREATE REMOTETHREAD(8)</b>	Dyre	source image	Privilege Escalation	Process injection	Monitoring Windows API calls Such as Creat Remote Thread
		C:\Windows\System32\rundll32.exe			
		6 threads in one minute			
		Taget image			
		C:\Windows\System32\SearchIndexer.exe			
		Targer image			
		C:\Windows\System32\svchost.exe			

Figure 9. Remote Thread Creation

SYSMON EVENT & ID	PAYLOAD	SYSMON LOG DETAILS	MITRE ATT&CK TACTIC	TECHNIQUE USED	SUGGESTED MITIGATION
REGISTRY EVENT (VALUE SET)13	Artemis	Image C:\Users\IEUser\Desktop\Artemis\InstallBC201401.exe	Defense Evasion	Registry Modification	Regisrict Registry permissions
		TargetObject HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\{1E453EA8-BB42-419D-8067-D2477A36B761}\InstallSource			
		Two events in one minute			
FILECREATE STREAMHASH(15)	sample.exe	Image C:\Program Files\Google\Chrome\Application\chrome.exe	Collection	Man in the Browser	User account management and user training.
		TargetFilename C:\Users\IEUser\Downloads\sample.exe			

Figure 10. Registry Event and File Create Stream Hash

## **Event Alerting**

The final step in this research was to make sure that events of interest can be sent to an alert system that can do it live. An alert system ElastAlert was chosen and configured to send specific alerts to a phone or any other gadget that can use telegram. Previous literature did not indicate an alert system existing on such a framework as the one I developed.

However, there was some evidence of it being used in other systems as a reliable alerting system [60]. Clear instructions on how to install and configure the system are found in the appendix. One of the malware mirc755.exe which was used to create a remote connection was used to test the system by configuring an alert system for any network connections. Below is a sample of the output of the alert which was sent to an iPhone and also sent to a configured MacBook system with Telegram.



tomoxwell

△ network\_event\_detect △

network\_event\_detect

At least 1 events occurred between 2019-09-06 12:49 EAT and 2019-09-06 13:19 EAT

```
@timestamp: 2019-09-06T10:19:01.322Z
_id: 6aUVBm0BKyLL-WgznhKv
_index: winlogbeat-7.3.1-2019.09.06
_type: _doc
message: Network connection detected:
RuleName:
UtcTime: 2019-09-06 10:19:01.498
ProcessGuid: {365ABB72-328D-5D72-0000-00106A8E0C00}
ProcessId: 4088
Image: C:\Users\IEUser\Desktop\mir755\mir755.exe
User: IEWIN7\IEUser
Protocol: tcp
Initiated: true
SourceIsIpv6: false
SourceIp: 10.4.0.70
SourceHostname: IEWIN7
SourcePort: 49222
SourcePortName:
DestinationIsIpv6: false
DestinationIp: 10.4.1.87
DestinationHostname:
DestinationPort: 4444
DestinationPortName:
num_hits: 2
num_matches: 2
```

Figure 11. Sample Telegram Alert

## CHAPTER 5

### CONCLUSION AND RECOMMENDATIONS

This research provides tangible results to organizations on how they can use Symon logs to investigate malware in general and also detect the bypassing of anti-virus solutions. This chapter will discuss the contributions that the research has made to the incident detection community. I have also highlighted the recommendations that can help guide organizations that may desire to use this research in their environment. I will also articulate the limitations of the research application. Finally, I give future research ideas that can encourage other researchers to further explore this research and expand on the ideas and results presented here.

#### **Contributions**

This research contributes to knowledge that can be applied by organizations to implement effective logging on windows Microsoft systems using Sysmon. The method used to implement this research can be used by organizations to compensate for their anti-virus weaknesses.

This research contributed by proving that anti-viruses use signature-based detection methods. The two malware that was made was able to bypass anti-viruses but once they were loaded to publicly available scan engines they were able to be detected by the anti-viruses. It is therefore important to provide compensatory approaches to the anti-viruses thus improving the overall security of the organizations. The detection also took place after some days meaning that there is a dangerous gap between malware detection and the updating of the signature databases.

This research contributes by providing a system that can provide real-time alerting of Sysmon events that can be detected on phone systems. Due to the nature of the speed that malware use during execution it requires one to have an alert system that is real-time and able to send notifications on your tablet on your phone.

This research used purely open-source software and it is a huge contribution because organizations can adopt it in their environment at zero cost. This research used Sysmon events and the order and sequence of their execution were seen from the empirical data provided. The most triggered events and the prevalent behaviours are recorded. The events that captured process creations, network connections, file creations, and process terminations were most frequently triggered by the data set.

The research also proved that the time of execution of the malware by different events proved to be very fast mostly in seconds or milliseconds. The dataset used for this research provided proof that the system used in this study could be successfully used to investigate malware and detect malware that can bypass anti-viruses.

### **Limitations**

This research has limitations that must be looked at to apply the results appropriately. The first limitation is that the research relied primarily on open-source software. The research did not use any commercially available solutions to address the research problem. This means there could be commercially available solutions that may address the research problem.

It is important to note at this point that this research was not meant to replace anti-virus solutions but rather compensate the anti-virus software. It is intended to work with anti-virus solutions to help investigate and detect incidences that may bypass the capability of anti-virus solutions.



This research is limited by the versions of software used. The software versions used can be upgraded anytime and the nature of the systems may not be the same. This research also used Sysmon logs to evaluate logs and it did not consider other logging software using the same malware dataset.

### **Recommendations**

In the process of implementing this research, several recommendations were brought forth. In light of the growing malware threats, organizations need to have logs in place that can be looked at when an incident occurs. It is also important that organizations detect malware as soon as possible. Due to the nature of the malware attacks and also attacks from zero-day attacks organizations need systems that can detect malware on the onset. Many anti-virus softwares use signatures to detect their malware and these signatures may take some time before they are updated. This research recommends using Sysmon to analyze logs with an Elastic stack to stay proactive against malware threats.

### **Future Work Expectations**

This research used the Sysmon events logs. The Appendices contain source code and configuration files and instructions to help other researchers leverage the progress made by this research. Researchers are encouraged to expand the sample space of the malware used and implement any other instantiations that can be used to analyze Sysmon logs.

This research used a manual method of analyzing malware dataset. More work can be done by automating the processes and improving the data set. This research also used the events created by Sysmon and researchers are encouraged to do the dynamic analysis of the malware to determine if there are additional activities that Sysmon does not currently monitor. Means could be explored that can lead to adding

this in the Sysmon software. Further work can be done to the alerting system to include all the activities that are suspect of malware behavior.

### **Summary**

The results of this research showed that Sysmon logging can be used to provide logging information that is used when investigating malware. Using this system malware can be detected and their activities triggered immediately. A dataset was picked from publicly available malware. This malware was identified as a major threat that has had serious effects on organizations making the research extremely relevant and applicable today.

This research adds to the overall knowledge about incidence response and anti-virus compensation. The contributions of this research fill those gaps that exist in the anti-viruses and provide detailed configuration files that can enable organizations to implement, verify, and enhance this research. Researchers are encouraged to build upon the research created in this study and further explore and expand Sysmon logging capabilities.

## APPENDICES

## APPENDIX A

### PAYLOAD GENERATION

#### A.1 Generating Shellter payload

Shellter is a dynamic shellcode injection tool. According to [56] it was the first truly dynamic PE (Portable Executable), infector ever created. It is used to inject shellcode into native Windows applications and as per the time of this research, it could only inject 32-bit applications.

The payload generated using shellter is injected into a legitimate windows process. In my case, I used an Internet Relay Chat (IRC) installer otherwise know as mIRC. The payload is injected into this installer and then used to infect a computer and create a Remote Connection. The mIRC installer was downloaded from their official site. According to, [65] mIRC is used by both organizations and individuals to communicate widely. This makes it a good tool for hiding malware by injection as it creates less suspicion when being installed in a system. The following steps were taken to generate the payload:

- Install shellter in Kali Linux and run it by typing the words shellter  
Run the shellter program by just typing the works shellter on the command line in Kali Linux
- There are two options for Automatic mode and manual mode For this research I chose manual mode because it helps me to choose what to exactly do with the Portable Executable that I want to inject
- Choose PE Target

The next step is to choose PE Target. This is the genuine executable that I injected with malicious code. In my case, this was the mIRC which is mirc755.exe which was the latest version available when doing this research.

- Enable stealth mode or not The stealth mode allows a user to do a couple of things with the executable like obfuscation and using polymorphic code. The stealth mode allows for the original functionality of the program to be maintained but if you just want to gain access to a computer you can decide not to use it. In my case, I did not choose stealth mode as I was just trying to gain access to another computer. The stealth mode allows the malware to behave as normal while the intruders use it to do other functions. Thus the user is fully unaware that the executable file is infected.

- Gather Dynamic Thread Context

In my case, I chose no

- Choose the number of Instructions

I chose 250 instruction since mIRC is small

I also chose not to check for self-modifying code while tracing and also not to trace all threads

- Choose whether to use listed payload or Custom

In my case, I chose to use the first listed payload which is Meterpreter Reverse TCP

- Set LHOST and LPORT

The next thing was to set the IP address of the local host that will be acting as the command and control center and also the port it will be using.

- Prepend Polymorphic code or not

This is an important step as it allows the malware to change the way it works while maintaining its functionality. This is key in anti-virus evasion as it helps the code to change making it hard for the anti-virus to track its behavior.

In my case, I appended it 1500 times. After its done, the payload is ready for execution and it can be sent to the target machine using any social engineering method.

- Set up the handler in the Kali Machine

Once the payload is ready you need to set up the handler from the Kali machine which makes it possible to catch the connection that the payload makes back to the Kali machine.

- Connect and to the target machine and exploit

After these steps, you can run the malware in the target machine, and then from the Kali machine you can for example get the information of the target machine or even send files to Kali from it.

## **A.2 Generating Zirikatu payload**

Another exploit tool that I used was zirikatu which can be downloaded from GitHub [62]. zirikatu is a fully undetectable payload generator. The steps below were used to generate the payload.

- Run the payload in Kali Linux

The payload can be run by just typing `./zirikatu.sh`

- Choose the type of payload to use and the number of iterations to apply

In my case, I chose number 7 which is for a multi encoded payload and then iterated it 40 times.

- Provide Kali Linux IP address and port number

After that, I provided the Kali Linux IP address and LHOST and also a port number to be used as LPORT. In my case, I used port 4444

- Change an icon or not and provide a file name

In my case, I chose not to change the icon and then provided a file name.

- Create a simple server to help you transport the payload to the client's machine

I created a simple python server by simply going to the folder where the payload we named is and issuing the command; `python -m SimpleHTTPServer`

After that, you can download the payload in any machine in the network by putting its IP address and port 8000

- Lastly, start the payload handler

I then started the payload handler from the Kali Linux machine and I was able to get a meterpreter session once I executed the payload in the windows machine.

## APPENDIX B

### INSTALLING AND CONFIGURING ELK

The following steps were taken to configure the ELK stack that I used in the research. The following steps were taken to install the ELK stack in Ubuntu 18.

- Elasticsearch and Logstash need java so our first step was to install java. As at this lab, Logstash does not support Java 10 so we used java 8. Install java 8 by running the command below:

`sudo apt install openjdk-8-jre apt-transport-https wget nginx`. You can check the java version installed by issuing the command: `java -version`. The output should be like below from our ubuntu machine.

```
[root@lab-Studio-XPS-435MT:/home/lab# java -version
java version "1.8.0_201"
Java(TM) SE Runtime Environment (build 1.8.0_201-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.201-b09, mixed mode)
root@lab-Studio-XPS-435MT:/home/lab# █
```

*Figure 12. Java-version Check*

- Elastic provides a complete repository for Debian systems that includes the whole stack. So first we added the GPG key by issuing the command below:

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
```

- We then created a file at `/etc/apt/sources.list.d/elastic.list` and then added the repo by issuing the command below:

```
deb https://artifacts.elastic.co/packages/6.x/apt stable main
```

- Save the file and exit. Then just update the program by issuing the command: `sudo apt-get update`.

- The next step is to install the Elasticsearch according to the Elasticsearch installation guide at

<https://www.elastic.co/guide/en/elasticsearch/reference/current/install-elasticsearch.html>.

The deb package is recommended so we install it by first getting the PGP key. Issue the command:

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -  
You should get an OK. message if all goes well.
```

- Set elastic package definition to our source list. Elastic search recommends that we have “apt-transport-https” installed first. So issue the command:

```
sudo apt-get install apt-transport-https
```

- Add elastic packages source list definitions to your source list(This allowed us to install Elasticsearch, Kibana and Logstash directly). Issue the command:

```
add-apt-repository “deb https://artifacts.elastic.co/packages/6.x/apt stable main”
```

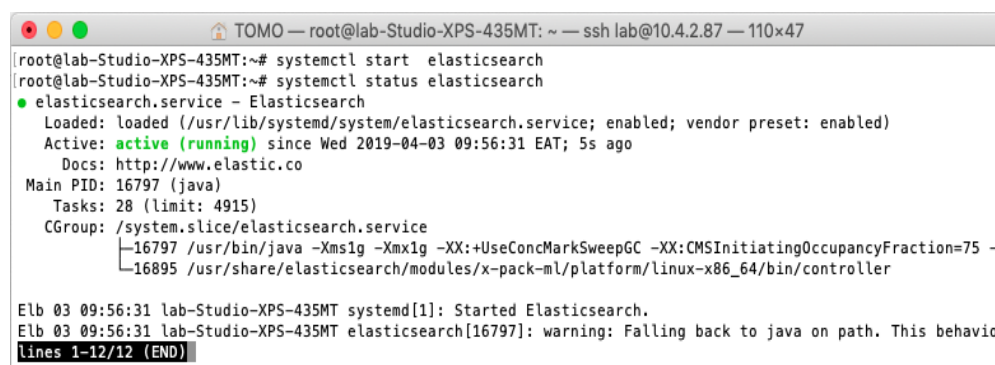
- After updating the system again install elasticsearch and Kibana by issuing the command:

```
sudo apt install elasticsearch kibana
```

- Its best practise to restrict access to elasticsearch on port 9200 from outside.
- edit the elasticsearch config file at /etc/elasticsearch/elasticsearch.yml

Remove the # symbol before the network host and add the IP address of the ubuntu machine or just put localhost.

- Remove the # symbol before the port section http.port : 9200
- Start the elasticsearch service and check its status to confirm if its running



```
TOMO — root@lab-Studio-XPS-435MT: ~ — ssh lab@10.4.2.87 — 110x47  
root@lab-Studio-XPS-435MT:~# systemctl start elasticsearch  
root@lab-Studio-XPS-435MT:~# systemctl status elasticsearch  
● elasticsearch.service - Elasticsearch  
   Loaded: loaded (/usr/lib/systemd/system/elasticsearch.service; enabled; vendor preset: enabled)  
   Active: active (running) since Wed 2019-04-03 09:56:31 EAT; 5s ago  
     Docs: http://www.elastic.co  
   Main PID: 16797 (java)  
    Tasks: 28 (limit: 4915)  
   CGroup: /system.slice/elasticsearch.service  
           └─16797 /usr/bin/java -Xms1g -Xmx1g -XX:+UseConcMarkSweepGC -XX:CMSInitiatingOccupancyFraction=75 -  
             └─16895 /usr/share/elasticsearch/modules/x-pack-ml/platform/linux-x86_64/bin/controller  
  
Elb 03 09:56:31 lab-Studio-XPS-435MT systemd[1]: Started Elasticsearch.  
Elb 03 09:56:31 lab-Studio-XPS-435MT elasticsearch[16797]: warning: Falling back to java on path. This behavior  
lines 1-12/12 (END)
```

Figure 13. Elasticsearch Status Check

- Enable the service so that it will start when the computer boots up.



```
TOMO — root@lab-Studio-XPS-435MT: ~ — ssh lab@10.4.2.87 — 110x47
root@lab-Studio-XPS-435MT:~# systemctl enable elasticsearch
Synchronizing state of elasticsearch.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable elasticsearch
root@lab-Studio-XPS-435MT:~# █
```

Figure 14. Elasticsearch Service Enabling

- Next we edit the kibana config file kibana.yml at /etc/kibana/kibana.yml.

Delete the # sign from the line server.host and put localhost as below:

Delete the # from server.port as below:

Delete the # elasticsearch.hosts as below:

- Start the Kibana service, check its status and enable it to start at system boot up.
  - The next step was to install Nginx. Issue the command: apt-get -y install nginx
- We configured Nginx as a reverse proxy.

```
TOMO — root@lab-Studio-XPS-435MT: /etc/kibana — ssh lab@10.4.2.87 — 110x47
# Kibana is served by a back end server. This setting specifies the port to use.
server.port: 5601

# Specifies the address to which the Kibana server will bind. IP addresses and host names are both valid value
s.
# The default is 'localhost', which usually means remote machines will not be able to connect.
# To allow connections from remote users, set this parameter to a non-loopback address.
server.host: "localhost"

# Enables you to specify a basePath at if you are running behind a proxy.
# Use the 'server.rewriteBasePath' setting to tell Kibana if it should remove the basePath
# from requests it receives, and to prevent a deprecation warning at startup.
# This setting cannot end in a slash.
#server.basePath: ""

# Specifies whether Kibana should rewrite requests that are prefixed with
# 'server.basePath' or require that they are rewritten by your reverse proxy.
# This setting was effectively always 'false' before Kibana 6.3 and will
# default to 'true' starting in Kibana 7.0.
#server.rewriteBasePath: false

# The maximum payload size in bytes for incoming server requests.
#server.maxPayloadBytes: 1048576

# The Kibana server's name. This is used for display purposes.
#server.name: "your-hostname"

# The URLs of the Elasticsearch instances to use for all your queries.
elasticsearch.hosts: ["http://localhost:9200"]

# When this setting's value is true Kibana uses the hostname specified in the server.host
# setting. When the value of this setting is false, Kibana uses the hostname of the host
# that connects to this Kibana instance.
#elasticsearch.preserveHost: true

# Kibana uses an index in Elasticsearch to store saved searches, visualizations and
# dashboards. Kibana creates a new index if the index doesn't already exist.
#kibana.index: ".kibana"

# The default application to load.
#kibana.defaultAppId: "home"

# If your Elasticsearch is protected with basic authentication, these settings provide
# the username and password that the Kibana server uses to perform maintenance on the Kibana
# index at startup. Your Kibana users still need to authenticate with Elasticsearch, which
# is proxied through the Kibana server.
```

Figure 15. Kibana File Edit

- Check the status of the nginx by issuing the command: `systemctl status nginx`

- Next we create an admin user to log on to our Kibana web interface. Issue the command:

```
echo "kibadmin:'openssl passwd -apr1'" | sudo tee -a /etc/nginx/htpasswd.users
```

follow the prompts to put a password and verify it.

- Remove the old nginx configuration and create a new one. Issue the commands below:

```
rm -r /etc/nginx/sites-available/default
```

`touch /etc/nginx/sites-available/kibana` We called ours Kibana but you can use any name.

- edit the new config file as below:
- Test and check the status as below:

```
TOMO — root@lab-Studio-XPS-435MT: ~ — ssh lab@10.4.2.87 — 110x47
root@lab-Studio-XPS-435MT:~# systemctl start kibana
root@lab-Studio-XPS-435MT:~# systemctl status kibana
● kibana.service - Kibana
   Loaded: loaded (/etc/systemd/system/kibana.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2019-04-03 10:15:29 EAT; 5s ago
 Main PID: 17229 (node)
    Tasks: 11 (limit: 4915)
   CGroup: /system.slice/kibana.service
           └─17229 /usr/share/kibana/bin/./node/bin/node --no-warnings --max-http-header-size=65536 /usr/shar

Elb 03 10:15:29 lab-Studio-XPS-435MT systemd[1]: Started Kibana.
[
root@lab-Studio-XPS-435MT:~# systemctl enable kibana
Synchronizing state of kibana.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable kibana
root@lab-Studio-XPS-435MT:~# systemctl status kibana
● kibana.service - Kibana
   Loaded: loaded (/etc/systemd/system/kibana.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2019-04-03 10:15:29 EAT; 1min 5s ago
 Main PID: 17229 (node)
    Tasks: 11 (limit: 4915)
   CGroup: /system.slice/kibana.service
           └─17229 /usr/share/kibana/bin/./node/bin/node --no-warnings --max-http-header-size=65536 /usr/shar

Elb 03 10:15:43 lab-Studio-XPS-435MT kibana[17229]: {"type":"log","@timestamp":"2019-04-03T07:15:43Z","tags":[
Elb 03 10:15:43 lab-Studio-XPS-435MT kibana[17229]: {"type":"log","@timestamp":"2019-04-03T07:15:43Z","tags":[
Elb 03 10:15:43 lab-Studio-XPS-435MT kibana[17229]: {"type":"log","@timestamp":"2019-04-03T07:15:43Z","tags":[
Elb 03 10:15:43 lab-Studio-XPS-435MT kibana[17229]: {"type":"log","@timestamp":"2019-04-03T07:15:43Z","tags":[
Elb 03 10:15:43 lab-Studio-XPS-435MT kibana[17229]: {"type":"log","@timestamp":"2019-04-03T07:15:43Z","tags":[
Elb 03 10:15:43 lab-Studio-XPS-435MT kibana[17229]: {"type":"log","@timestamp":"2019-04-03T07:15:43Z","tags":[
Elb 03 10:15:43 lab-Studio-XPS-435MT kibana[17229]: {"type":"log","@timestamp":"2019-04-03T07:15:43Z","tags":[
Elb 03 10:15:44 lab-Studio-XPS-435MT kibana[17229]: {"type":"log","@timestamp":"2019-04-03T07:15:44Z","tags":[
Elb 03 10:15:44 lab-Studio-XPS-435MT kibana[17229]: {"type":"log","@timestamp":"2019-04-03T07:15:44Z","tags":[
root@lab-Studio-XPS-435MT:~# █
```

Figure 16. Kibana Service Check

```

TOMO — root@lab-Studio-XPS-435MT: ~ — ssh lab@10.4.2.87 — 110x47
[root@lab-Studio-XPS-435MT:~# systemctl status nginx
● nginx.service – A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2019-03-31 17:53:51 EAT; 2 days ago
     Docs: man:nginx(8)
  Main PID: 1036 (nginx)
    Tasks: 9 (limit: 4915)
   CGroup: /system.slice/nginx.service
           └─1036 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
             └─1037 nginx: worker process
               └─1038 nginx: worker process
                 └─1039 nginx: worker process
                   └─1040 nginx: worker process
                     └─1041 nginx: worker process
                       └─1042 nginx: worker process
                         └─1043 nginx: worker process
                           └─1044 nginx: worker process

Bit 31 17:53:47 lab-Studio-XPS-435MT systemd[1]: Starting A high performance web server and a reverse proxy se
Bit 31 17:53:51 lab-Studio-XPS-435MT systemd[1]: Started A high performance web server and a reverse proxy ser
root@lab-Studio-XPS-435MT:~# █

```

Figure 17. Nginx Status Check

- Go to a browser and point it to the IP address of the ubuntu server and then log in with the credentials we created before for accessing Kibana through the web interface:
- Next we need to configure logstash that we installed before. We first need to generate SSL Certificates in order to secure the connections between our endpoints and our ELK stack. Create the directories that are needed to store our certificates

```

[root@lab-Studio-XPS-435MT:~# cat /etc/nginx/sites-available/kibana

server {
    listen 80;

    server_name 10.4.1.87;

    auth_basic "Restricted Access";
    auth_basic_user_file /etc/nginx/htpasswd.kibana;

    location / {
        proxy_pass http://localhost:5601;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}

root@lab-Studio-XPS-435MT:~# █

```

Figure 18. Nginx Configuration Commands

```

TOMO — root@lab-Studio-XPS-435MT: ~ — ssh lab@10.4.2.87 — 110x47
[root@lab-Studio-XPS-435MT:~# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[root@lab-Studio-XPS-435MT:~# systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2019-03-31 17:53:51 EAT; 2 days ago
     Docs: man:nginx(8)
  Main PID: 1036 (nginx)
    Tasks: 9 (limit: 4915)
   CGroup: /system.slice/nginx.service
           └─1036 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
             └─1037 nginx: worker process
             └─1038 nginx: worker process
             └─1039 nginx: worker process
             └─1040 nginx: worker process
             └─1041 nginx: worker process
             └─1042 nginx: worker process
             └─1043 nginx: worker process
             └─1044 nginx: worker process

Bit 31 17:53:47 lab-Studio-XPS-435MT systemd[1]: Starting A high performance web server and a reverse proxy se
Bit 31 17:53:51 lab-Studio-XPS-435MT systemd[1]: Started A high performance web server and a reverse proxy ser
[root@lab-Studio-XPS-435MT:~# █

```

Figure 19. Nginx Status Check figure and Private Keys

```
sudo mkdir -p /etc/pki/tls/certs
```

```
sudo mkdir /etc/pki/tls/private
```

- Generate the SSL certificate and private key in the locations we set before. Issue the commands:

```
cd /etc/pki/tls
```

```
sudo openssl req -config /etc/ssl/openssl.cnf -x509 -days 3650 -batch -nodes
-newkey rsa:2048 -keyout private/logstash-forwarder.key -out
certs/logstash-forwarder.crt
```

- Next we create the custom Logstash configuration files. These files are located in /etc/logstash/conf.d. The configuration consists of three sections: Inputs, Filters and Out.

First we create our input file which sets the way on how logstash is going to receive logs being sent to our ELK stack.

Create the file: `sudo vi /etc/logstash/conf.d/02-beats-input.conf`

Your file should look like the one below:

In my case I turned off SSL because of an existing bug in the ELK stack.

```
TOMO — root@lab-Studio-XPS-435MT: /home/lab — ssh lab@10.4.2.87 — 110x47

input {
  beats {
    port => 5044
    add_field => {"[@metadata][source]" => "winlogbeat"}
    ssl => false
    ssl_certificate => "/etc/pki/tls/certs/logstash-forwarder.crt"
    ssl_key => "/etc/pki/tls/private/logstash-forwarder.key"
  }
}

```

Figure 20. Logstash Configuration

- Next we create the output file: `sudo vi /etc/logstash/conf.d/50-elasticsearch-output.conf`

Your file should look like the one below:

```
TOMO — root@lab-Studio-XPS-435MT: /home/lab — ssh lab@10.4.2.87 — 110x47

output{
  if [@metadata][source] == "winlogbeat"{
    elasticsearch{
      hosts => ["localhost:9200"]
      sniffing => true
      manage_template => false
      index => "%{[@metadata][beat]}-%{[@metadata][version]}-%{+YYYY.MM.dd}"
      ssl_certificate_verification => false
      document_type => "%{[@metadata][type]}"
    }
  }
}

```

Figure 21. Logstash Output File Configuration

It's ideal to note at this point that the index part already creates an index for the data being sent to elasticsearch and there is therefore no need to upload a winlog beat template to our elasticsearch instance.

- At this point we start logstash and check its status.

```
TOMO — root@lab-Studio-XPS-435MT: /home/lab — ssh lab@10.4.2.87 — 110x47

[root@lab-Studio-XPS-435MT:/home/lab# systemctl start logstash ]
[root@lab-Studio-XPS-435MT:/home/lab# systemctl status logstash ]
● logstash.service - logstash
   Loaded: loaded (/etc/systemd/system/logstash.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2019-04-03 12:21:14 EAT; 4s ago
     Main PID: 17876 (java)
       Tasks: 24 (limit: 4915)
    CGroup: /system.slice/logstash.service
            └─17876 /usr/bin/java -Xms1g -Xmx1g -XX:+UseConcMarkSweepGC -XX:CMSInitiatingOccupancyFraction=75 -

Elb 03 12:21:14 lab-Studio-XPS-435MT systemd[1]: Started logstash.
[root@lab-Studio-XPS-435MT:/home/lab# ]

```

Figure 22. Logstash Status Check

## APPENDIX C

### INSTALLING AND CONFIGURING SYSMON

#### C.1 Installing Sysmon

As at this lab I used Sysmon v9.0 which can be downloaded from the link below: <https://docs.microsoft.com/en-us/sysinternals/downloads/Sysmon> [63] The steps for installing Sysmon are as follows:

- Extract the contents of the zipped Sysmon file to a directory of preference. In our case we extracted it to the Downloads folder of the Windows 7 Virtual Machine. To check the options available for Sysmon navigate to the folder where Sysmon was installed and issue the command: `Sysmon.exe /?`

The option gives the available options that you can use when installing Sysmon. At the very top it also shows the version of Sysmon that you are using and in our case we are using version 9.0.

- We picked a basic installation. The first command `-accepteula` allows the program to automatically accept the EULA. The second option `-i` is for the installation as seen from the previous picture. Remember also to point it to the configuration file which we downloaded and stored in the same folder which in our case is the downloads folder. To do that change directory to where the Sysmon folder is and issue the following command: A successful install will give the following message:
- To check if your Sysmon config is already working navigate to windows event viewer. Follow the path as follows: Applications and Service Logs > Microsoft > Windows > Sysmon > Operational. If its operational you should have something like the window below from our lab.
- Keep customizing your config file as many times as you may wish until you get your desired end. For a quick start compare with existing config files that you can freely find online. Just make sure you understand the rules in the config file for optimal results. You can start with a few and keep adding them as you test.

```

Administrator: C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd C:\Users\IEUser\Downloads\Sysmon

C:\Users\IEUser\Downloads\Sysmon> Sysmon.exe \?

System Monitor v9.0 - System activity monitor
Copyright (C) 2014-2019 Mark Russinovich and Thomas Garnier
Sysinternals - www.sysinternals.com

Usage:
Install: Sysmon.exe -i [<configfile>]
        [-h <[sha1|md5|sha256|imphash|*],...>] [-n [<process,...>]]
        [-l [<process,...>]]
Configure: Sysmon.exe -c [<configfile>]
        [--![-h <[sha1|md5|sha256|imphash|*],...>] [-n [<process,...>]]
        [-l [<process,...>]]]
Uninstall: Sysmon.exe -u
-c Update configuration of an installed Sysmon driver or dump the
  current configuration if no other argument is provided. Optionally
  take a configuration file.
-d Specify the name of the installed device driver image.
  Configuration entry: DriverName.
  The service image and service name will be the same
  name of the Sysmon.exe executable image.
-h Specify the hash algorithms used for image identification (default
  is SHA1). It supports multiple algorithms at the same time.
  Configuration entry: HashAlgorithms.
-i Install service and driver. Optionally take a configuration file.
-l Log loading of modules. Optionally take a list of processes to track.
-m Install the event manifest (done on service install as well).
-n Log network connections. Optionally take a list of processes to track.
-r Check for signature certificate revocation.
  Configuration entry: CheckRevocation.
-s Print configuration schema definition of the specified version.
  Specify 'all' to dump all schema versions (default is latest).
-u Uninstall service and driver.

The service logs events immediately and the driver installs as a boot-start
driver to capture activity from early in the boot that the service will write
to the event log when it starts.

On Vista and higher, events are stored in "Applications and Services
Logs/Microsoft/Windows/Sysmon/Operational". On older systems, events are
written to the System event log.

If you need more information on configuration files, use the '-? config'
command. More examples are available on the Sysinternals website.

Specify -accepteula to automatically accept the EULA on installation,
otherwise you will be interactively prompted to accept it.

Neither install nor uninstall requires a reboot.

C:\Users\IEUser\Downloads\Sysmon>_

```

Figure 23. Sysmon Version Check

```

C:\Users\IEUser\Downloads\Sysmon>Sysmon.exe -accepteula -i C:\Users\IEUser\Down
loads\sysmonconfig-export.xml_

```

Figure 24. Sysmon Installation

## C.2 Configuration File

Sysmon uses a configuration that can be customised to meet the needs of different situations. You can find pre-configured configurations that can suit your needs or you can configure your own. For our case we used a configuration from Github customized by swift on security. This configuration captures most of the events to keep an eye on for security control. It can be downloaded on the following Github link: <https://github.com/SwiftOnSecurity/Sysmon-config/blob/master/Sysmonconfig-export.xml> [64].

```

Configuration file successfully applied.
Sysmon installed.
SysmonDrv installed.
Starting SysmonDrv.
SysmonDrv started.
Starting Sysmon..
Sysmon started.

```

Figure 25. Successful Sysmon Install message

The screenshot shows the Windows Event Viewer interface. On the left, the event log tree is expanded to 'Operational'. The main pane displays a list of events from the Sysmon source. The details pane for 'Event 1, Sysmon' is open, showing the following information:

Level	Date and Time	Source	Event ID	Task C...
Information	4/2/2019 9:34:20 PM	Sysmon	1	Proces...
Information	4/2/2019 9:29:48 PM	Sysmon	3	Networ...
Information	4/2/2019 9:29:15 PM	Sysmon	1	Proces...
Information	4/2/2019 9:28:46 PM	Sysmon	1	Proces...
Information	4/2/2019 9:28:46 PM	Sysmon	5	Proces...
Information	4/2/2019 9:28:39 PM	Sysmon	1	Proces...
Information	4/2/2019 9:28:34 PM	Sysmon	5	Proces...
Information	4/2/2019 9:28:31 PM	Sysmon	1	Proces...
Information	4/2/2019 9:28:02 PM	Sysmon	1	Proces...
Information	4/2/2019 9:27:50 PM	Sysmon	1	Proces...
Information	4/2/2019 9:19:27 PM	Sysmon	1	Proces...
Information	4/2/2019 9:19:20 PM	Sysmon	1	Proces...
Information	4/2/2019 9:11:09 PM	Sysmon	5	Proces...
Information	4/2/2019 9:11:09 PM	Sysmon	1	Proces...
Information	4/2/2019 9:09:36 PM	Sysmon	1	Proces...
Information	4/2/2019 8:51:33 PM	Sysmon	1	Proces...
Information	4/2/2019 8:19:19 PM	Sysmon	1	Proces...
Information	4/2/2019 8:00:57 PM	Sysmon	1	Proces...
Information	4/2/2019 7:57:39 PM	Sysmon	1	Proces...
Information	4/2/2019 7:57:38 PM	Sysmon	1	Proces...
Information	4/2/2019 7:51:34 PM	Sysmon	1	Proces...
Information	4/2/2019 7:19:18 PM	Sysmon	1	Proces...
Information	4/2/2019 7:12:13 PM	Sysmon	1	Proces...
Information	4/2/2019 7:12:12 PM	Sysmon	1	Proces...
Information	4/2/2019 7:12:12 PM	Sysmon	1	Proces...

**Event 1, Sysmon**

General | Details

Process Create:  
RuleName:  
UtcTime: 2019-04-02 18:34:20.885

Log Name: Microsoft-Windows-Sysmon/Operational

Source: Sysmon      Logged: 4/2/2019 9:34:20 PM

Event ID: 1      Task Category: Process Create (rule: ProcessCreate)

Level: Information      Keywords:

User: SYSTEM      Computer: IEWIN7

OpCode: Info

More Information: [Event Log Online Help](#)

Figure 26. Sample Operation Sysmon Event Viewer



## APPENDIX D

### INSTALLING AND CONFIGURING WINLOGBEAT

The steps for downloading and installing winlogbeat are as follows:

- Download winlogbeat from: <https://www.elastic.co/downloads/beats/winlogbeat>
- Unzip the downloaded folder and copy it to C: Program Files

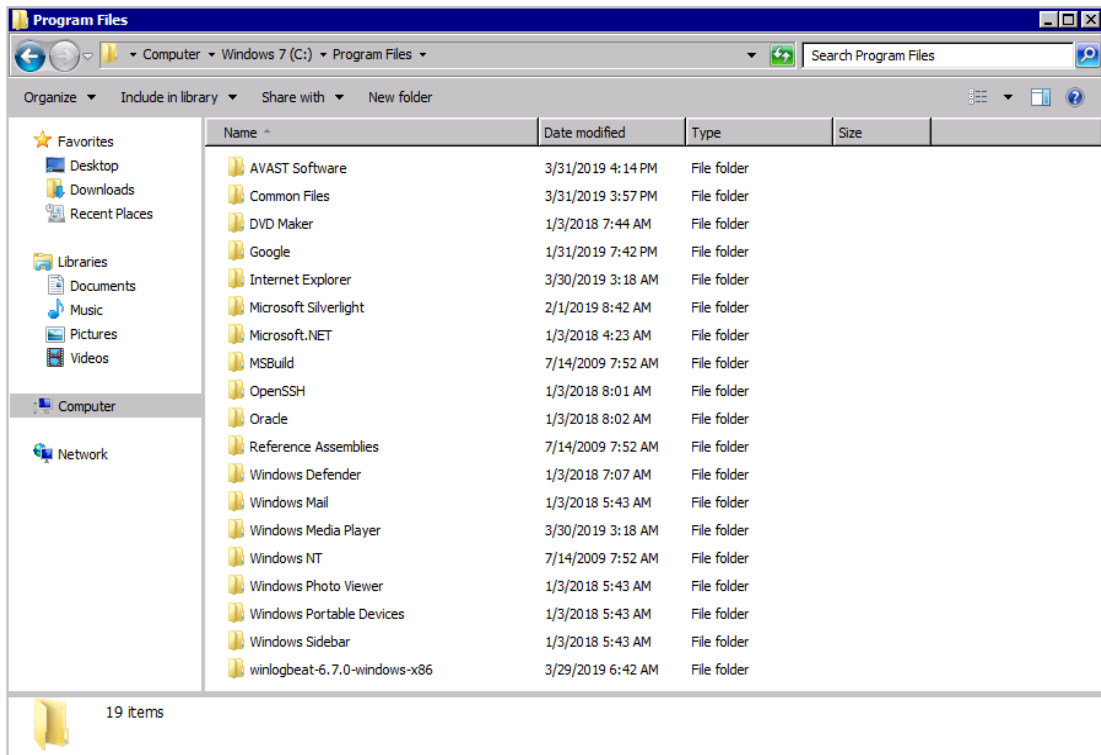


Figure 27. Checking Winlogbeat Version

Our winlogbeat is version 6.7 as from the diagram.

- Open powershell as Admin and navigate to the winlogbeat folder contents that we copied in the previous step.
- Install winlogbeat by issuing the command: `.\install-winlogbeat-service.ps1` You may get the error below:

The fix is to run Set-ExecutionPolicy and change the Execution Policy setting as below and type Y for yes when prompted.

- We now need to edit the winlogbeat config file so as it will work with logstash. Run a text editor of your choice as an admin. In our case we used wordpad instead of notepad because the xml file is more structured and easy to read line by line.
- Open the winlogbeat config file (winlogbeat.yml) in the winlogbeat folder. The first thing we do is to edit the log type files that it collects and so we need to point it to the Sysmon logs. We need to add the line name: Microsoft-windows-Sysmon/operational as shown below:
- Optionally you can create a certificate from ubuntu and use it by putting it in the section for certificates and also manually running it in the windows machine that will be sending logs so that they are on a secure connection. For this experiment the certificates were not used.

```

Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\IEUser> cd 'C:\Program Files\winlogbeat-6.7.0-windows-x86'
PS C:\Program Files\winlogbeat-6.7.0-windows-x86> .\install-service-winlogbeat.ps1
File C:\Program Files\winlogbeat-6.7.0-windows-x86\install-service-winlogbeat.ps1 cannot be loaded. The file C:\Program
Files\winlogbeat-6.7.0-windows-x86\install-service-winlogbeat.ps1 is not digitally signed. The script will not execute
on the system. Please see "get-help about_signing" for more details..
At line:1 char:33
+ .\install-service-winlogbeat.ps1 <<<<
+ CategoryInfo          : NotSpecified: (:) [], PSSecurityException
+ FullyQualifiedErrorId : RuntimeException

PS C:\Program Files\winlogbeat-6.7.0-windows-x86> _

```

Figure 28. Possible Error During Installation

```

Administrator: Windows PowerShell
Files\winlogbeat-6.7.0-windows-x86\install-service-winlogbeat.ps1 is not digitally signed. The script will not execute
on the system. Please see "get-help about_signing" for more details..
At line:1 char:33
* .\install-service-winlogbeat.ps1 <<<<
* CategoryInfo          : NotSpecified: (:) [1], PSSecurityException
* FullyQualifiedErrorId : RuntimeException

PS C:\Program Files\winlogbeat-6.7.0-windows-x86> cd
PS C:\Program Files\winlogbeat-6.7.0-windows-x86> cd
PS C:\Program Files\winlogbeat-6.7.0-windows-x86> Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass

Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose
you to the security risks described in the about_Execution_Policies help topic. Do you want to change the execution
policy?
[Y] Yes [N] No [S] Suspend [?] Help (default is "Y"): y
PS C:\Program Files\winlogbeat-6.7.0-windows-x86> .\install-service-winlogbeat.ps1

__GENUS          : 2
__CLASS          : __PARAMETERS
__SUPERCLASS    :
__DYNASTY       : __PARAMETERS
__RELPATH       :
__PROPERTY_COUNT : 1
__DERIVATION    : {}
__SERVER        :
__NAMESPACE     :
__PATH          :
ReturnValue      : 0

__GENUS          : 2
__CLASS          : __PARAMETERS
__SUPERCLASS    :
__DYNASTY       : __PARAMETERS
__RELPATH       :
__PROPERTY_COUNT : 1
__DERIVATION    : {}
__SERVER        :
__NAMESPACE     :
__PATH          :
ReturnValue      : 0

Status          : Stopped
Name            : winlogbeat
DisplayName     : winlogbeat

PS C:\Program Files\winlogbeat-6.7.0-windows-x86>

```

Figure 29. Fixing The Error

- Make sure the output.logstash section is uncommented by defining the host where the logstash host will be running. In our case the IP address of the ubuntu server is used. The ubuntu server 18 was used to install the ELK stack.

```

# event_logs specifies a list of event logs to monitor as well
as any
# accompanying options. The YAML data type of event_logs is a
list of
# dictionaries.
#
# The supported keys are name (required), tags, fields,
fields_under_root,
# forwarded, ignore_older, level, event_id, provider, and
include_xml. Please
# visit the documentation for the complete details of each
option.
# https://go.es.io/WinlogbeatConfig
winlogbeat.event_logs:
  - name: Application
    ignore_older: 72h
  - name: Security
  - name: System
  - name: Microsoft-windows-sysmon/operational

#===== Elasticsearch template setting
=====

setup.template.settings:
  index.number_of_shards: 3
  #index.codec: best_compression
  #_source.enabled: false

#===== General
=====

# The name of the shipper that publishes the network data. It
can be used to group
# all the transactions sent by a single shipper in the web
interface.
#name:

# The tags of the shipper are included in their own field with

```

Figure 30. Pointing to Sysmon Logs

```

#----- Logstash output
-----
output.logstash:
  # The Logstash hosts
  hosts: ["██████████:5044"]

  # Optional SSL. By default is off.
  # List of root certificates for HTTPS server verifications
  # ssl.certificate_authorities: ['C:\Users\IEUser\logstash-
forwarder.crt']

  # Certificate for SSL client authentication
  #ssl.certificate: "/etc/pki/client/cert.pem"

  # Client Certificate Key
  #ssl.key: "/etc/pki/client/cert.key"

```

Figure 31. Defining the Host

- Test your winlogbeat config by running: `.\winlogbeat.exe -c .\winlogbeat.yml -configtest -e`

If everything is fine you should see a line that says config OK as shown below from our PowerShell output.

```

PS C:\Program Files\winlogbeat-6.7.0-windows-x86> .\winlogbeat.exe -c .\winlogbeat.yml -configtest -e
Flag --configtest has been deprecated, use test config subcommand
2019-04-02T23:18:16.921+0300 INFO instance/beat.go:612 Home path: [C:\Program Files\winlogbeat-6.7.0-windows-x86]
61 Config path: [C:\Program Files\winlogbeat-6.7.0-windows-x86] Data path: [C:\Program Files\winlogbeat-6.7.0-windows-x86\data]
6 data] Logs path: [C:\Program Files\winlogbeat-6.7.0-windows-x86\logs]
2019-04-02T23:18:16.921+0300 INFO instance/beat.go:619 Beat UUID: 5b32bd07-c381-475f-9b07-e70600215733
2019-04-02T23:18:16.921+0300 INFO [beat] instance/beat.go:932 Beat info {"system_info": {"beat": {"path": {"config": "C:\\Program Files\\winlogbeat-6.7.0-windows-x86", "data": "C:\\Program Files\\winlogbeat-6.7.0-windows-x86\\data", "home": "C:\\Program Files\\winlogbeat-6.7.0-windows-x86", "logs": "C:\\Program Files\\winlogbeat-6.7.0-windows-x86\\logs"}, "type": "winlogbeat", "uuid": "5b32bd07-c381-475f-9b07-e70600215733"}}}
2019-04-02T23:18:16.922+0300 INFO [beat] instance/beat.go:941 Build info {"system_info": {"build": {"commit": "14ca49c28a6e10b84b4ea8cdebdc46bd2eab3130", "libbeat": "6.7.0", "time": "2019-03-21T14:50:24.000Z", "version": "6.7.0"}}}
2019-04-02T23:18:16.922+0300 INFO [beat] instance/beat.go:944 Go runtime info {"system_info": {"go": {"os": "windows", "arch": "386", "max_procs": 1, "version": "go1.10.8"}}}
2019-04-02T23:18:16.940+0300 INFO [beat] instance/beat.go:948 Host info {"system_info": {"host": {"architecture": "x86", "boot_time": "2019-04-02T17:23:35.99+03:00", "name": "IEWIN7", "ip": ["10.4.0.70/22"], "mac": "127.0.0.1/8", "fe80::5efe:a04:46:128"}, "kernel_version": "6.1.7601.24387 (win7sp1_ldr_escrow.190305-1700)", "mac": "08:00:27:10:b8:d0", "os": "08:00:00:00:00:00"}, "os": {"family": "windows", "platform": "windows", "name": "Windows 7 Enterprise", "version": "6.1", "major": 1, "minor": 0, "patch": 0, "build": "7601.24385", "timezone": "EAT", "timezone_offset_sec": -10800, "id": "365abb72-f240-45fe-b90-41a82626497"}}}
2019-04-02T23:18:16.943+0300 INFO [beat] instance/beat.go:977 Process info {"system_info": {"process": {"cwd": "C:\\Program Files\\winlogbeat-6.7.0-windows-x86", "exe": "C:\\Program Files\\winlogbeat-6.7.0-windows-x86\\winlogbeat.exe", "name": "winlogbeat.exe", "pid": 884, "ppid": 2628, "start_time": "2019-04-02T23:18:15.244+0300"}}}
2019-04-02T23:18:16.943+0300 INFO [publisher] pipeline/module.go:110 Beat name: IEWIN7
2019-04-02T23:18:16.952+0300 INFO [beater] winlogbeat.go:68 State will be read from and persisted to C:\Program Files\winlogbeat-6.7.0-windows-x86\data\winlogbeat.yml
2019-04-02T23:18:16.953+0300 WARN [cfgwarn] instance/beat.go:383 DEPRECATED: -configtest flag has been deprecated, use configtest subcommand Will be removed in version: 6.0
Config OK
2019-04-02T23:18:16.953+0300 INFO [monitoring] log/log.go:117 Starting metrics logging every 30s
2019-04-02T23:18:17.106+0300 INFO [monitoring] log/log.go:152 Total non-zero metrics {"monitoring": {"metrics": {"beat": {"cpu": {"system": {"ticks": 80, "time": {"ms": 80}}, "total": {"ticks": 90, "time": {"ms": 90}, "value": 90, "user": {"tick": 10, "time": {"ms": 10}}}, "handles": {"open": 125}, "info": {"ephemeral_id": "466d05f-a97i-48ef-ad1i-7e71655385f1", "uptime": {"ms": 87}}, "memstats": {"gc_next": 4473924, "memory_alloc": 2411168, "memory_total": 2411168, "wsc": 13783040}}, "libbeat": {"config": {"module": {"running": 0}, "output": {"type": "logstash"}, "pipeline": {"clients": 0, "events": {"active": 0}}, "system": {"cpu": {"cores": 1}}}}}}
2019-04-02T23:18:17.107+0300 INFO [monitoring] log/log.go:153 Uptime: 240.5392ms
2019-04-02T23:18:17.107+0300 INFO [monitoring] log/log.go:130 Stopping metrics logging.
2019-04-02T23:18:17.107+0300 INFO instance/beat.go:385 winlogbeat stopped.
PS C:\Program Files\winlogbeat-6.7.0-windows-x86>

```

Figure 32. Winlogbeat Status Check

- As you can see the service is stopped and our action now is to start the service. Issue the command: start-service winlogbeat under powershell and you should have it running in the windows services if all went well as below:
- At this point its good to check if all our services are running.

Point your browser to the IP address of the ubuntu server where the ELK stack is installed. Log in with the credentials we created and create a new index in kibana called winlogbeat-\* under the management section. At this point if everything is running well we should have our logs coming from the windows VM to the elastic stack.

You should have something similar to the diagram.

From the diagram we can see that we are getting logs from our windows 7 virtual machine.

## APPENDIX E

### INSTALLING AND CONFIGURING ELASALERT

ElasticAlert requires some prerequisites listed below  
Elasticsearch

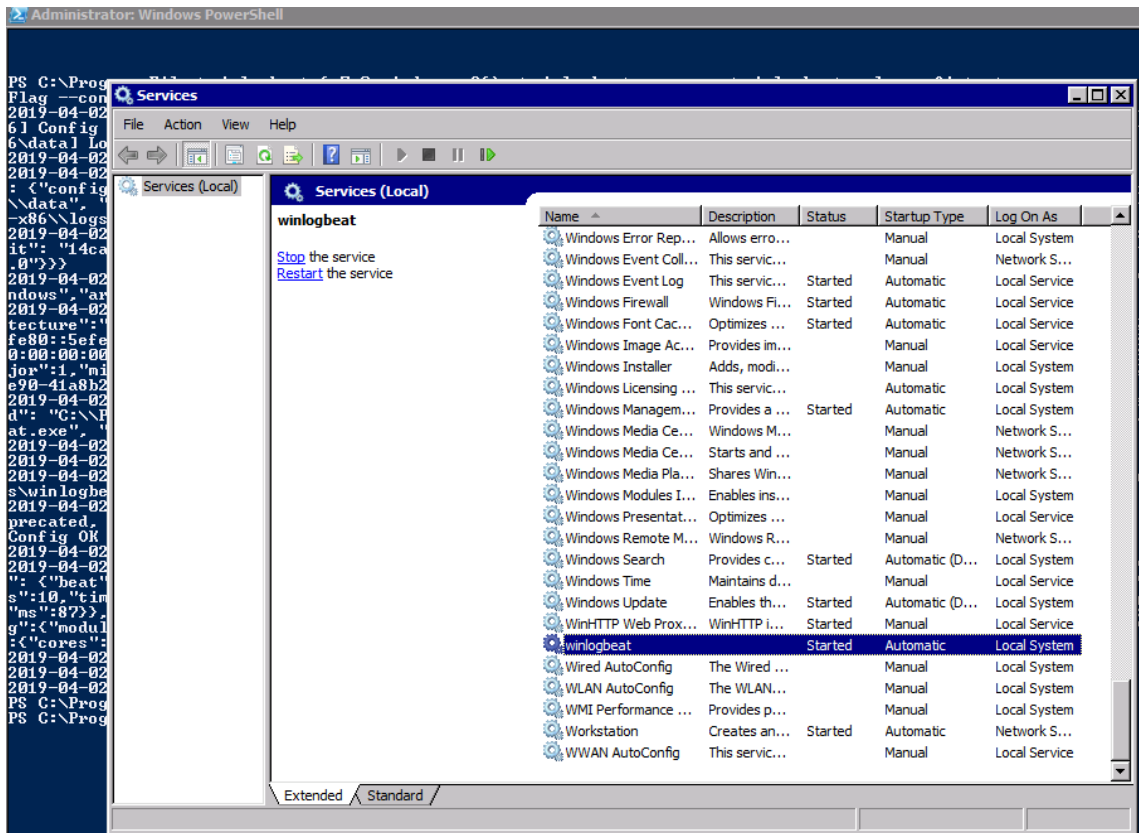


Figure 33. Starting Winlogbeat

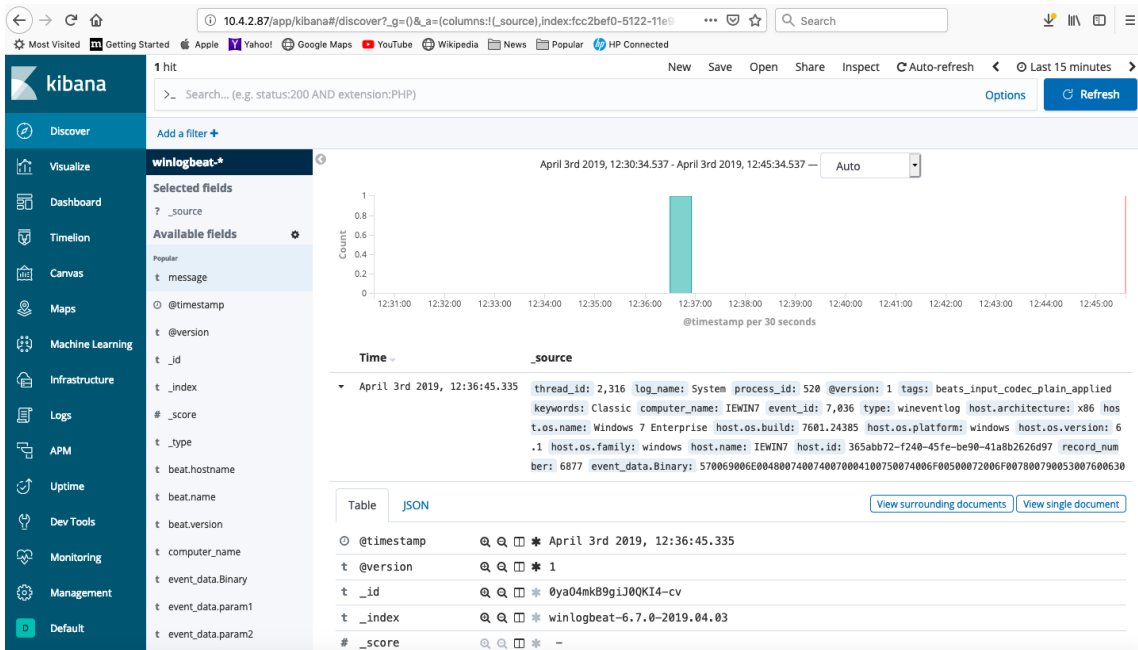


Figure 34. ELK operational Check

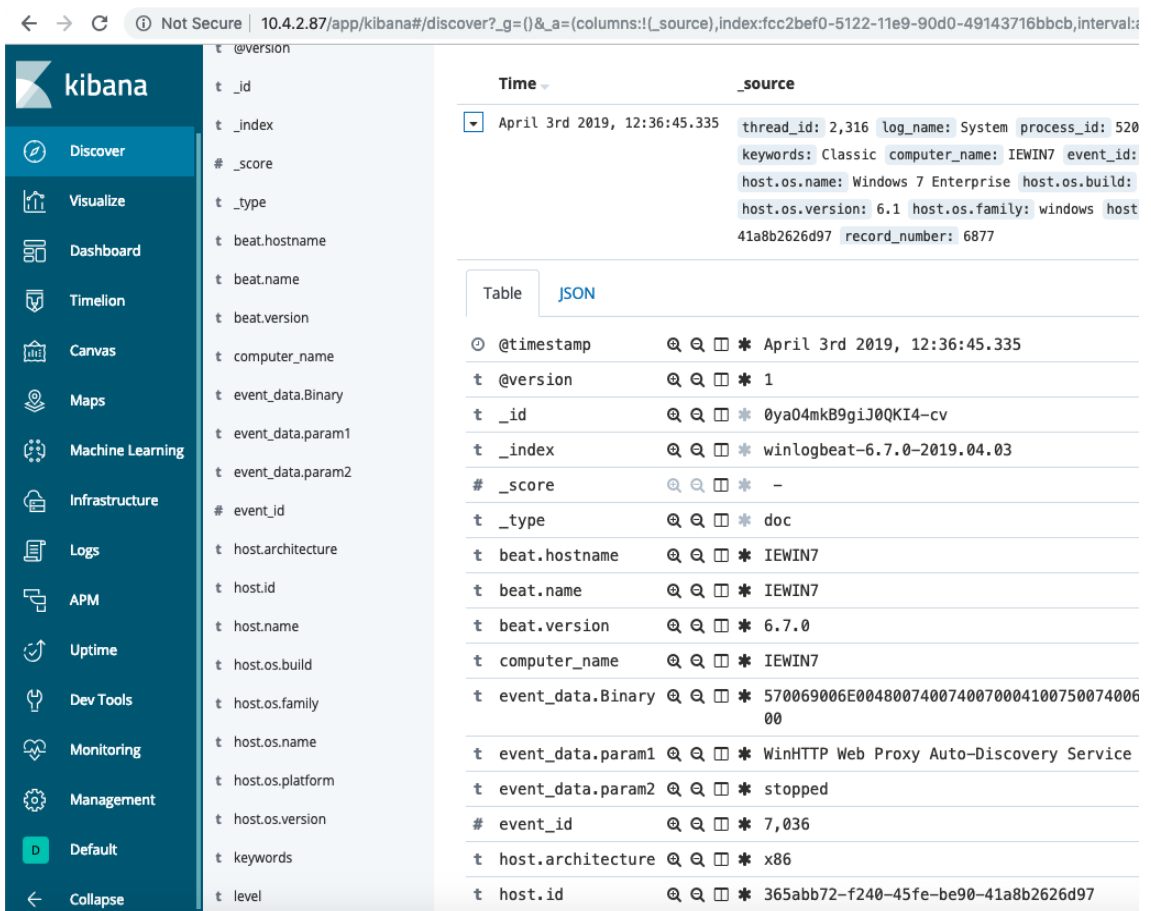


Figure 35. Sample Logs from Windows 7 Machine

ISO8601 or Unix timestamped data  
Python 2.7  
pip, see requirements.txt  
(<https://github.com/Yelp/elastalert/blob/master/requirements.txt>)  
Packages for ubuntu python-pip python-dev libffi-dev libssl-dev

The first thing is to install python 2.7. Issue the command : `sudo apt-get install python-minimal`

Install needed packages: `sudo apt-get install python-pip python-dev libffi-dev libssl-dev`

I did my installation by cloning the git repository. So we first install git by issuing the command: `sudo apt-get install git`

ElastAlert is installed to the “opt” folder so the directory needs to be changed: `sudo cd /opt`

Next Clone a git repository: `sudo git clone https://github.com/Yelp/elastalert.git`

Next install the module as below:

```
sudo pip install "setuptools>=11.3"  
sudo python setup.py install
```

A wrong version of elasticsearch may produce errors at this point. Make sure you use the latest version which above 5. In my case I used version 6.7

Configure ElastAlert

```
change directory to where you installed ElastAlert : cd /opt/elastalert/  
make a copy of config.yml.example : sudo cp config.yml.example  
config.yml
```

modify the config.yml : `vim config.yml` as below:

```
set the Elasticsearch hostname or Ip as follows : es_host : localhost  
set the ElasticServer port : es_port : 9200
```

other options are optional as follows:

```
es_username  
es_password  
in my case i did not set those ones.
```

save and close

Create ElasticAlert index : `sudo elastic-alert-create-index`

The last step is to create a rule for your alert. For my case I created a rule that sends an alert to my telegram App when a remote connection is established in a computer. Most hackers gain and maintain access in computers by creating remote access.



# APPENDIX F

## GRAPHS FOR MALWARE EVENT SEQUENCE

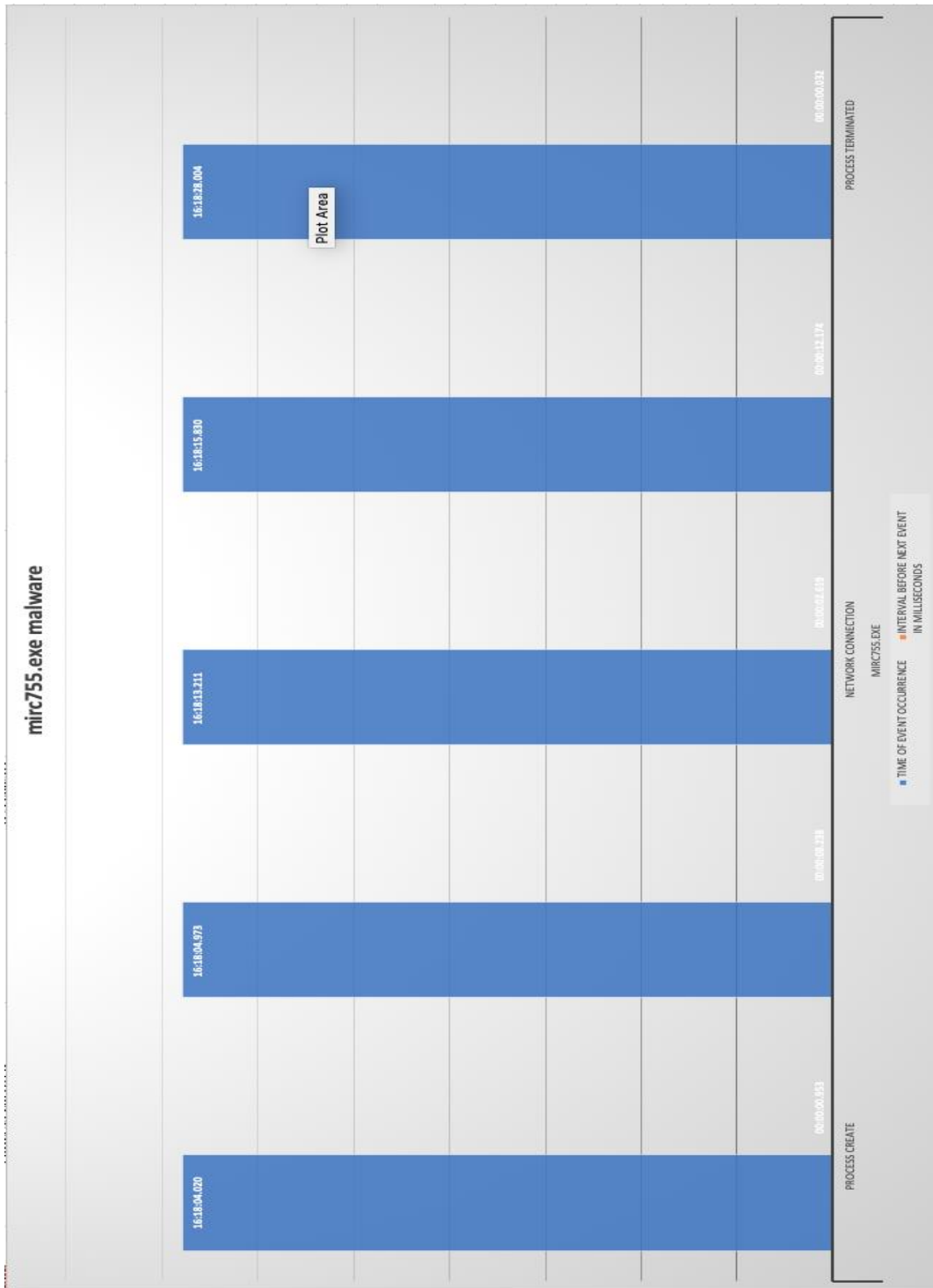


Figure 36. mir755 Event Sequence

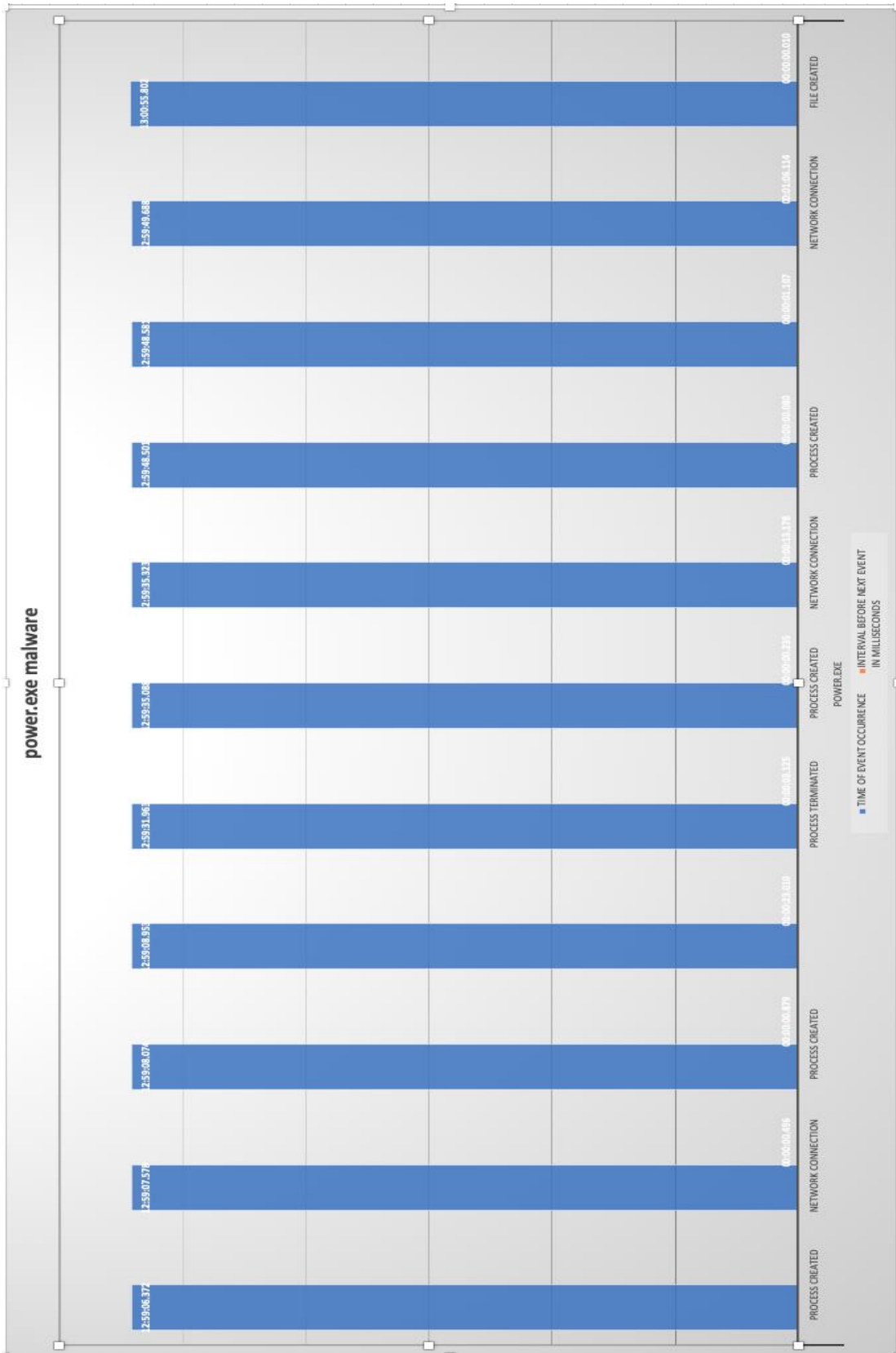


Figure 37. Power.exe Event Sequence



Figure 38. Nivdort Event Sequence

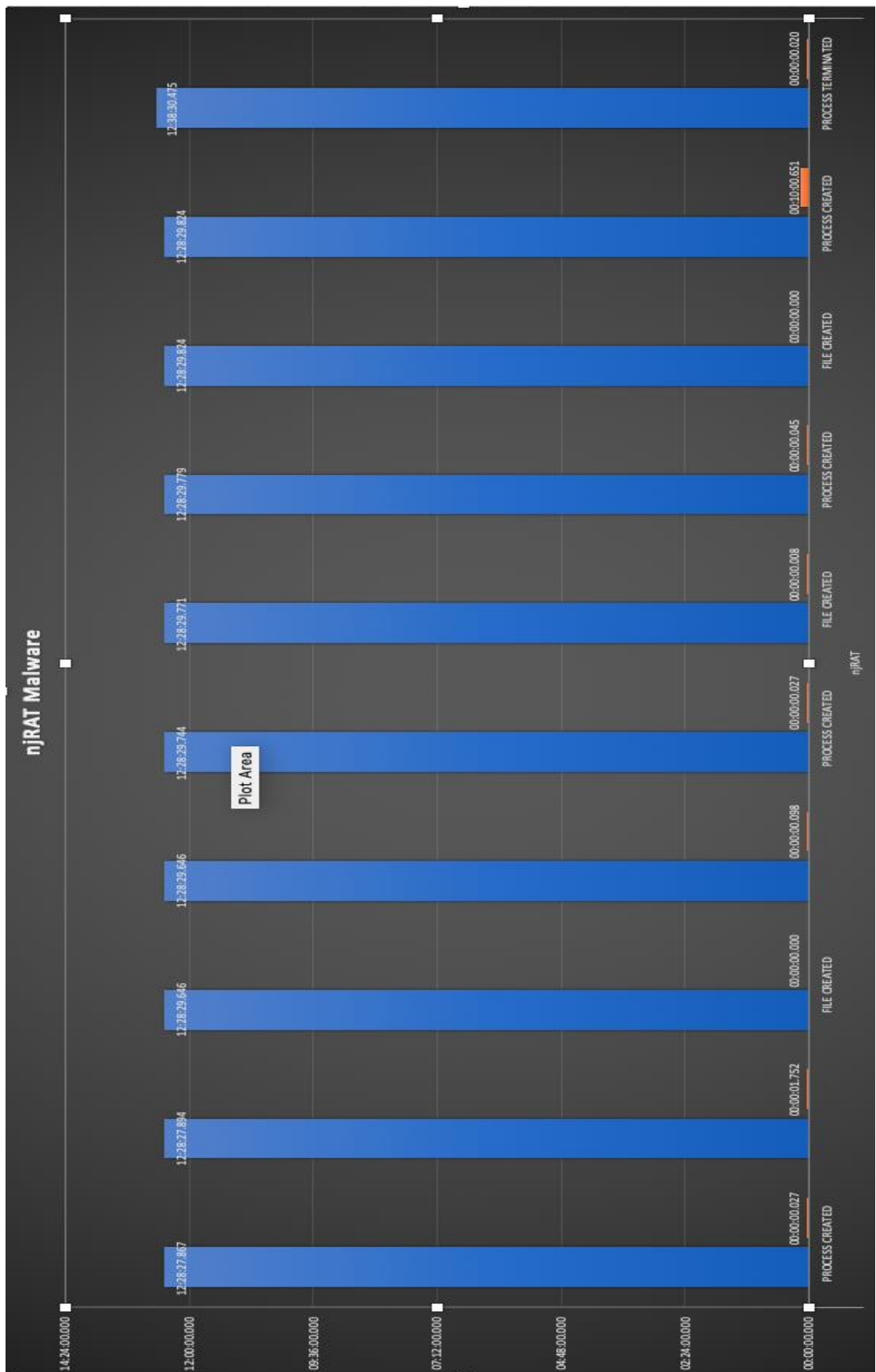


Figure 39. Njrat Event Sequence

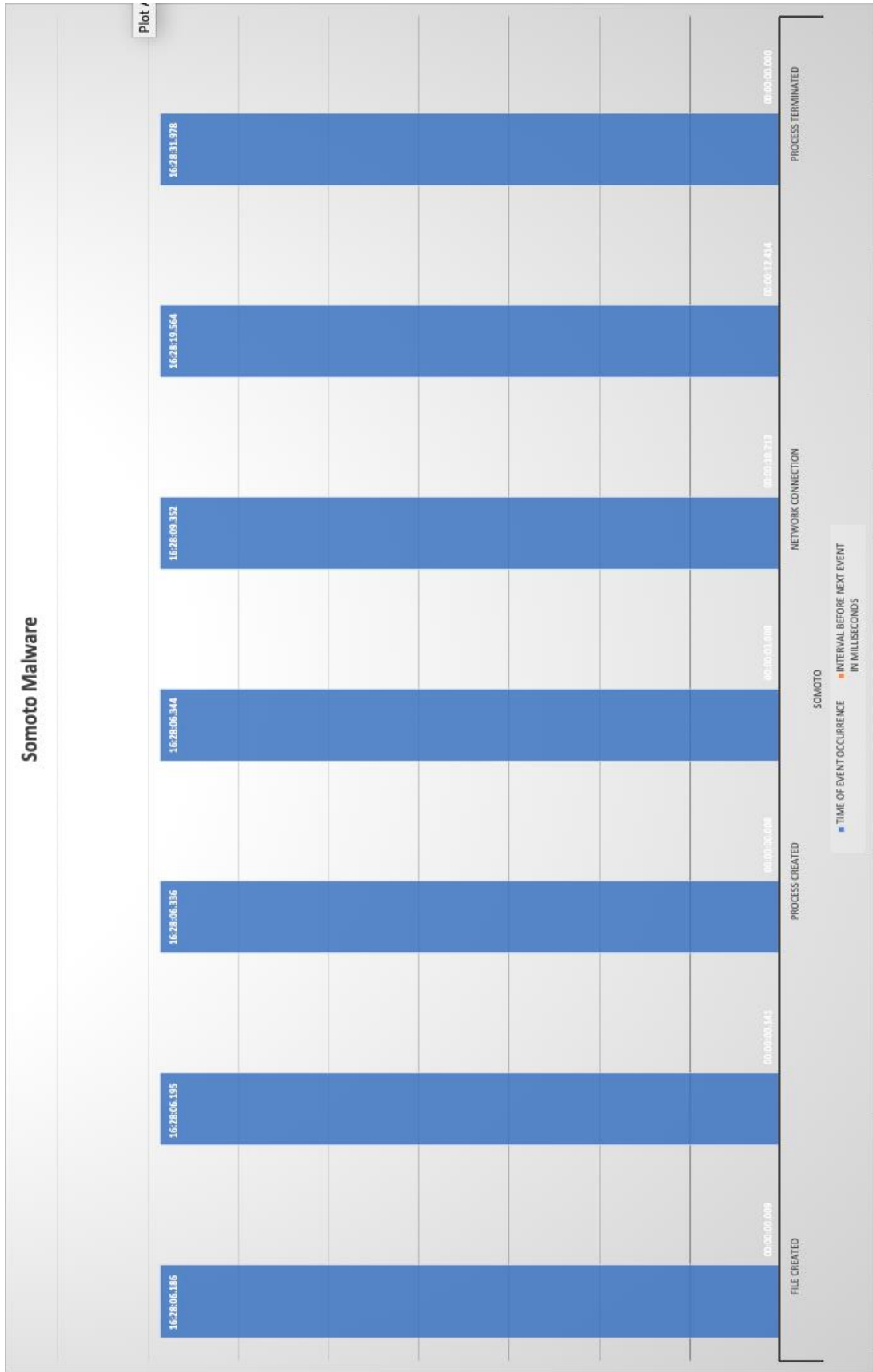


Figure 40. Somoto1 Event Sequence

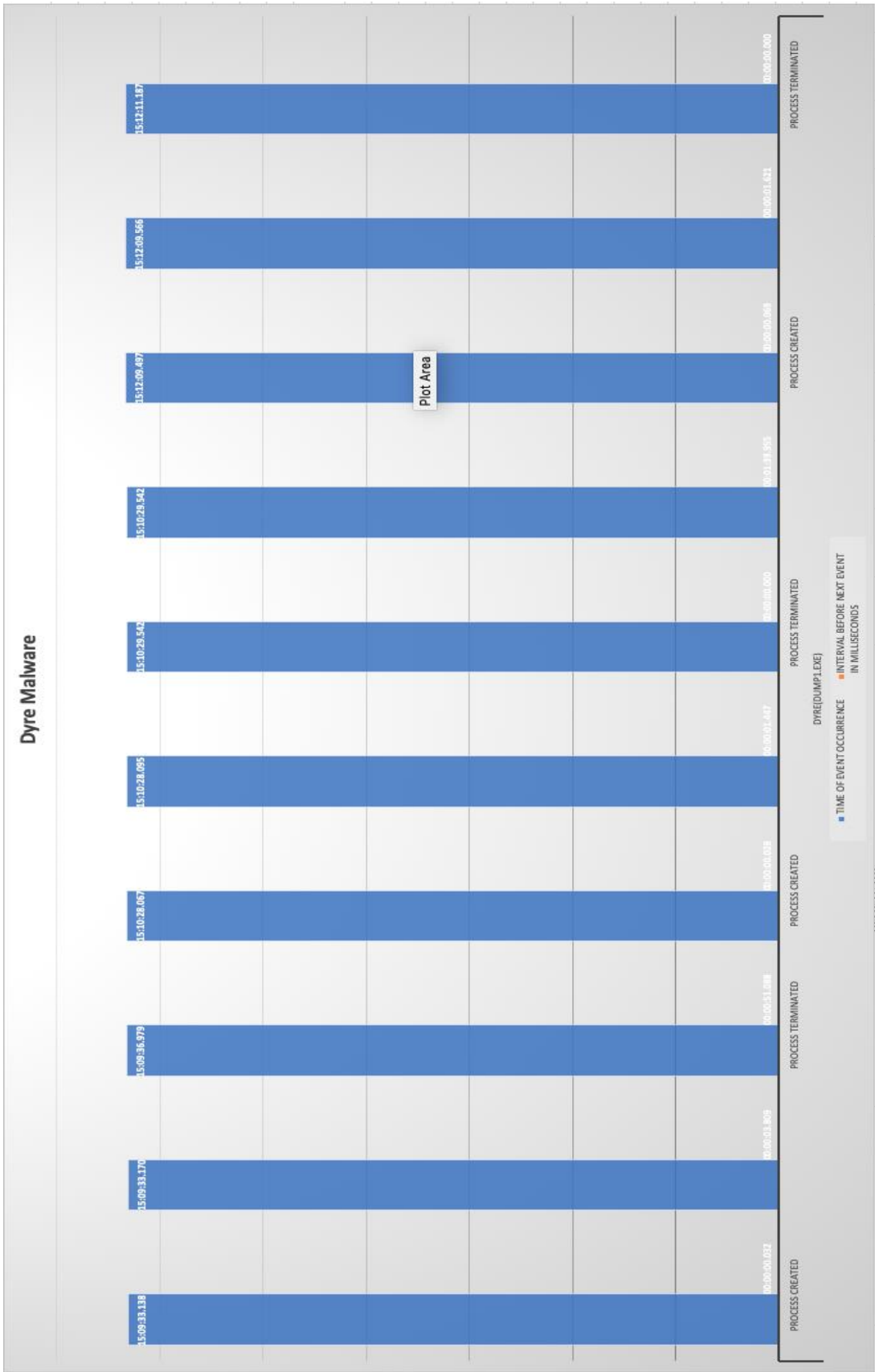


Figure 41. Somoto2 Event Sequence

<b>PAYLOAD</b>	<b>ORDER OF EVENT(S) TRIGGERED</b>	<b>TIME OF EVENT OCCURRENCE</b>	<b>INTERVAL BEFORE NEXT EVENT IN MILLISECONDS</b>
<b>Artemis</b>	FILESTREAM CREATED	14:16:11.718	00:01:03.395
	PROCESS CREATED	14:17:15.113	00:00:00.230
		14:17:15.343	00:00:00.592
	FILE CREATED	14:17:15.935	00:00:00.021
		14:17:15.956	00:00:05.494
		14:17:21.450	00:00:00.049
		14:17:21.499	00:00:00.009
		14:17:21.508	00:00:00.000
		14:17:21.508	00:00:00.011
		14:17:21.519	00:00:00.000
		14:17:21.519	00:00:03.701
		14:17:25.220	00:00:00.000
		14:17:25.220	00:00:00.009
		14:17:25.229	00:00:00.000
		14:17:25.229	00:00:00.051
		14:17:25.280	00:00:00.009
	PROCESS CREATED	14:17:25.289	00:00:00.009
		14:17:25.298	00:00:00.008
		14:17:25.306	00:00:00.135
		14:17:25.441	00:00:00.032
		14:17:25.473	00:00:00.223
		14:17:25.696	00:00:00.151
		14:17:25.847	00:00:00.219
		14:17:26.066	00:00:00.031
		14:17:26.097	00:00:00.164
		14:17:26.261	00:00:00.031
		14:17:26.292	00:00:01.061
		14:17:27.353	00:00:00.105
		14:17:27.458	00:00:00.648
		14:17:28.106	00:00:00.230
		14:17:28.336	00:00:00.160
		14:17:28.496	00:00:00.191
		14:17:28.687	00:00:00.583
		14:17:29.270	00:00:00.024

Figure 42. Artemis1 Event Sequence

		14:17:29.294	00:00:00.232
		14:17:29.526	00:00:00.163
		14:17:29.689	00:00:00.160
		14:17:29.849	00:00:00.095
		14:17:29.944	00:00:00.348
		14:17:30.292	00:00:00.166
		14:17:30.458	00:00:00.075
		14:17:30.533	00:00:00.078
		14:17:30.611	00:00:00.094
		14:17:30.705	00:00:00.088
		14:17:30.793	00:00:00.390
		14:17:31.183	00:00:00.055
		14:17:31.238	00:00:00.627
		14:17:31.865	00:00:00.035
		14:17:31.900	00:00:00.142
		14:17:32.042	00:00:00.026
		14:17:32.068	00:00:00.148
		14:17:32.216	00:00:00.061
		14:17:32.277	00:00:00.017
		14:17:32.294	00:00:00.051
		14:17:32.345	00:00:00.123
		14:17:32.468	00:00:00.148
		14:17:32.616	00:00:00.335
		14:17:32.951	00:00:00.143
		14:17:33.094	00:00:00.028
		14:17:33.122	00:00:00.093
		14:17:33.215	00:00:00.027
		14:17:33.242	00:00:00.085
		14:17:33.327	00:00:00.062
		14:17:33.389	00:00:00.028
		14:17:33.417	00:00:00.052
		14:17:33.469	00:00:00.051
		14:17:33.520	00:00:00.054
		14:17:33.574	00:00:00.031
		14:17:33.605	00:00:00.092
		14:17:33.697	00:00:00.022

Figure 43. Artemis2 Event Sequence

		14:17:33.719	00:00:00.084
		14:17:33.803	00:00:00.055
		14:17:33.858	00:00:00.183
		14:17:34.041	00:00:00.058
		14:17:34.099	00:00:00.052
		14:17:34.151	00:00:00.307
		14:17:34.458	00:00:00.164
		14:17:34.622	00:00:00.275
	REGISTRY VALUE SET	14:17:34.897	00:00:00.301
	PROCESS CREATED	14:17:35.198	00:00:00.189
	REGISTRY VALUE SET	14:17:35.387	00:00:05.329
	PROCESS TERMINATED	14:17:40.716	00:00:00.008
		14:17:40.724	

Figure 44. Artemis3 Event Sequence



## REFERENCES

- [1] K. Bissel, “Ninth Annual Cost of Cybercrime,” *Accenture*, 2019. [Online]. Available: <https://www.accenture.com/us-en/insights/security/cost-cybercrime-study>. [Accessed Nov 1, 2019].
- [2] S. Duncan, “The technical and social history of software engineering,” *Softw. Qual. Prof.*, vol. 16, no. 3, 2014.
- [3] Sujith, “A Brief History of Antivirus Software,” *TechlineInfo*, 13-Oct-2013. [Online]. Available: <http://www.techlineinfo.com/a-brief-history-of-antivirus-software/>. [Accessed Nov 1, 2019].
- [4] B. Thuraisingham, P. Parveen, M. M. Masud, and L. Khan, *Big Data Analytics with Applications in Insider Threat Detection*. Boca Raton, FL: CRC Press, 2017.
- [5] R. Pranamulia, Y. Asnar, and R. S. Perdana, “Profile hidden Markov model for malware classification — Usage of system call sequence for malware classification,” in *2017 International Conference on Data and Software Engineering (ICoDSE)*, 2017, pp. 1–5.
- [6] Assange, “Vault 7: Projects,” *WikiLeaks- Releases*. [Online]. Available: <https://wikileaks.org/vault7/releases/>. [Accessed Oct 1, 2019].
- [7] I. Alsmadi, *The NICE Cyber Security Framework: Cyber Security Intelligence and Analytics*. Oxford, UK: Springer, 2019.
- [8] Govolution, “AntiVirus Evasion Tool.” 2019 [Online]. Available: <https://github.com/govolution/avet>. [Accessed Aug 1, 2019].
- [9] J. Koret and E. Bachaalany, *The Antivirus Hacker’s Handbook*. Indianapolis, IN: Wiley, 2015.
- [10] D. Regalado et al., *Gray Hat Hacking: The Ethical Hacker’s Handbook*. New York, NY: McGraw-Hill Education, 2015.
- [11] V. K. Velu, *Mastering Kali Linux for Advanced Penetration Testing: Secure Your Network with Kali Linux - The Ultimate White Hat Hackers’ Toolkit*, 2nd Revised. Birmingham, UK: Packt, 2017.
- [12] KyREcon, “Shellter AV-Evasion Artware,” *Shellter*, 2017. [Online]. Available: <https://www.shellterproject.com/>. [Accessed July 1, 2019].

- [13] P. Bramwell, *Hands-On Penetration Testing on Windows: Unleash Kali Linux, PowerShell, and Windows Debugging Tools for Security Testing and Analysis*. Birmingham, UK: Packt, 2018.
- [14] P. Prasad, *Mastering Modern Web Penetration Testing*. Birmingham, UK: Packt, 2016.
- [15] S. Sinha, *Beginning Ethical Hacking with Kali Linux: Computational Techniques for Resolving Security Issues*. New York, NY: Springer, 2018.
- [16] Rapid7, “Metasploit Framework,” *Metasploit*, Aug-2019. [Online]. Available: <https://metasploit.help.rapid7.com/docs/msf-overview>. [Accessed July 1, 2019].
- [17] G. N. Barbosa and R. R. Branco, “Prevalent Characteristics in Modern Malware.” Blackhat, 2014 [Online]. Available: <https://www.blackhat.com/docs/us-14/materials/us-14-Branco-Prevalent-Characteristics-In-Modern-Malware.pdf>. [Accessed Nov 1, 2019].
- [18] K. Bissel, “PowerShell Obfuscation Using SecureString,” *Threat Vector*, 12-Sep-2018. [Online]. Available: [https://threatvector.cylance.com/en\\_us/home/unpacking-a-packer-powershell-obfuscation-using-securestring.html](https://threatvector.cylance.com/en_us/home/unpacking-a-packer-powershell-obfuscation-using-securestring.html). [Accessed July 8, 2019].
- [19] J. Singh and J. Singh, “Challenges of malware analysis: Obfuscation techniques,” *Int. J. Inf. Secur. Sci.*, vol. 7, no. 3, pp. 100–110, 2018.
- [20] B. Kolosnjaji et al., “Adversarial Malware Binaries: Evading Deep Learning for Malware Detection in Executables,” presented at the 26th European Signal Processing Conference (EUSIPCO), 2018, pp. 533–537.
- [21] K. Mathur, S. Hiranwal, and S. Balaji, “A survey on techniques in detection and analyzing malware,” *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 3, no. 4, 2013.
- [22] I. You and K. Yim, “Malware obfuscation techniques: A brief survey,” in *2010 International Conference on Broadband, Wireless Computing, Communication and Applications*, 2010, pp. 297–300.
- [23] E. Gandotra, D. Bansal, and S. Sofat, “Malware analysis and classification: A survey,” *J. Inf. Secur.*, vol. 5, no. 2, pp. 56–64, Feb. 2014.
- [24] M. A. Davis, S. M. Bodmer, and A. LeMasters, *Hacking Exposed: Malware & Rootkits Secrets & Solutions*. New York, NY: McGraw-Hill Education, 2009.
- [25] D. Uppal, V. Mehra, and V. Verma, “Basic survey on malware analysis, tools and techniques,” *Int. J. Comput. Sci. Appl. IJCSA*, vol. 4, no. 1, pp. 103–112, 2014.

- [26] D. K. Mahawer and A. Nagaraju, “Metamorphic malware detection using base malware identification approach,” *Secur. Commun. Netw.*, vol. 7, no. 11, pp. 1719–1733, 2014.
- [27] B. B. Rad, M. Masrom, and S. Ibrahim, “Camouflage in malware: From encryption to metamorphism,” *Int. J. Comput. Sci. Netw. Secur.*, vol. 12, no. 8, pp. 74–83, 2012.
- [28] R. W. Beggs, *Mastering Kali Linux for Advanced Penetration Testing*. Birmingham, UK: Packt, 2017.
- [29] J. Drew, M. Hahsler, and T. Moore, “Polymorphic malware detection using sequence classification methods and ensembles,” *EURASIP J. Inf. Secur.*, vol. 2017, no. 1, p. 2, Jan. 2017.
- [30] B. Wanswett and H. K. Kalita, “The threat of obfuscated zero day polymorphic malwares: An analysis,” in *2015 International Conference on Computational Intelligence and Communication Networks (CICN)*, 2015, pp. 1188–1193.
- [31] B. J. Kumar, H. Naveen, B. P. Kumar, S. S. Sharma, and J. Villegas, “Logistic regression for polymorphic malware detection using ANOVA F-test,” in *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, 2017, pp. 1–5.
- [32] M. S. Khan, S. Siddiqui, and K. Ferens, “Cognitive modeling of polymorphic malware using fractal based semantic characterization,” in *2017 IEEE International Symposium on Technologies for Homeland Security (HST)*, 2017, pp. 1–7.
- [33] C. Easttom, *Modern Cryptography: Applied Mathematics for Encryption and Information Security*. New York, NY: McGraw-Hill Education, 2015.
- [34] G. Weidman, *Penetration Testing: A Hands-On Introduction to Hacking*. San Francisco, CA: No Starch Press, 2014.
- [35] B. Payette, *Windows PowerShell in Action*. Shelter Island, NY: John Wiley, 2017.
- [36] D. Patten, “The Evolution to Fileless Malware.” East Carolina University, North Carolina, 2017.
- [37] A. Mohanta, K. Velmurugan, and M. Hahad, *Preventing Ransomware: Understand, Prevent, and Remediate Ransomware Attacks*. Birmingham, UK: Packt, 2018.
- [38] M. E. Russinovich and A. Margosis, *Troubleshooting with the Windows Sysinternals Tools*. New York, NY: Microsoft Press, 2016.
- [39] M. E. Russinovich, “How to Go from Responding to Hunting with Sysinternals Sysmon,” in *RSA Conference*, San Francisco, CA, 2017.

- [40] M. E. Russinovich, *Malware Hunting with the Sysinternals Tools*. 2012 [Online]. Available: <https://channel9.msdn.com/Events/TechEd/Europe/2012/SIA302>. [Accessed June 28, 2019].
- [41] M. E. Russinovich, "Tracking Hackers on Your Network with Sysinternals Sysmon," @*GrandomThoughts*, 12-Mar-2016. [Online]. Available: [http://graysonwalters.com/post/markrussinovich\\_Sysmon/](http://graysonwalters.com/post/markrussinovich_Sysmon/). [Accessed: 28-Jun-2019]
- [42] T. W. Edgar and D. O. Manz, *Research Methods for Cyber Security*. Rockland, MA: Syngress, 2017.
- [43] L. Y. Nativ, "theZoo- A Live Malware Repository." 2019 [Online]. Available: <https://github.com/ytisf/theZoo>. [Accessed Nov 1, 2019].
- [44] VirusTotal, "VirusTotal - Free Online Virus, Malware and URL Scanner," *Semantic Scholar*, 2011. [Online]. Available: </paper/VirusTotal-Free-Online-Virus%2C-Malware-and-URL-VirusTotal/bca1b9e9d65fdb8a80293e00cd936535aef8fe21>. [Accessed Nov 1, 2019]
- [45] N. Jaswal, *Mastering Metasploit*. Birmingham, UK: Packt, 2016.
- [46] P. Wagenseil, "Best Free Antivirus Software 2019," *Tom's Guide*, 10-Jul-2019. [Online]. Available: <https://www.tomsguide.com/us/best-free-antivirus,review-6003.html>. [Accessed Nov 7, 2019].
- [47] M. Marco, "PreciseSecurity.com - Internet Security & News," *PreciseSecurity.com*, 05-Jan-2019. [Online]. Available: <https://www.precisecurity.com/adware/pup-optional-somoto>. [Accessed Sep 13, 2019].
- [48] J. Spidle, "How to erase the artemis virus," *Chron.com*. [Online]. Available: <https://smallbusiness.chron.com/eraseartemis-virus-50865.html>. [Accessed Sep 13, 2019].
- [49] B. Stone-Gross and P. Khandhar, "Dyre Banking Trojan Threat Analysis," *SecureWorks*, 17-Dec-2014. [Online]. Available: <https://www.secureworks.com/research/dyre-banking-trojan>. [Accessed Oct 13, 2019].
- [50] R. H. Diwakar, "Nivdort: Data-Stealing Trojan Arrives via Spam," *McAfee Blogs*, 18-Feb-2016. [Online]. Available: <https://www.mcafee.com/blogs/other-blogs/mcafee-labs/nivdort-data-stealing-trojan-arrives-via-spam/>. [Accessed Sep 13, 2019].
- [51] C. Osborne, "NjRat Secures Top Spot as Most Active Network Malware in 2017," *ZDNet*, 23-Jan-2018. [Online]. Available: <https://www.zdnet.com/article/njrat-secures-top-spot-as-most-active-malware-on-networks-in-2017/>. [Accessed Sep 13, 2019].

- [52] N. Thamsirarak, T. Seethongchuen, and P. Ratanaworabhan, “A case for malware that make antivirus irrelevant,” in *2015 12th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, 2015, pp. 1–6.
- [53] B. Min, V. Varadharajan, U. Tupakula, and M. Hitchens, “Antivirus security: Naked during updates,” *Softw. Pract. Exp.*, vol. 44, no. 10, pp. 1201–1222, 2014.
- [54] Avast AVG Team, “Avast Free Antivirus Wins Product of the Year,” *CNET*, Feb-2019. [Online]. Available: <https://www.cnet.com/forums/discussions/avast-free-antivirus-wins-product-of-the-year/>. [Accessed Sep 13, 2019].
- [55] V. Mavroeidis and A. Jøsang, “Data-driven threat hunting using Sysmon,” in *Proceedings of the 2nd International Conference on Cryptography, Security and Privacy*, Guiyang, China, 2018, pp. 82–88.
- [56] P. Delgado, “Threat Hunting with Sysmon: Word Document with Macro,” *Syspanda*, 10-Oct-2017. [Online]. Available: <https://www.syspanda.com/index.php/2017/10/10/threat-hunting-Sysmon-word-document-macro/>. [Accessed Sep 13, 2019].
- [57] J. C. Center, “Detecting Lateral Movement Through Tracking Event Logs.” JPCERT Coordination Center, 12-Jun-2017 [Online]. Available: <https://msdnshared.blob.core.windows.net/media/2017/10/Detecting-Lateral-Movement-through-Tracking-Event-Logs.pdf>. [Accessed Nov 1, 2019].
- [58] A. Islam and Z. Bu, “Classifying Sets of Malicious Indicators for Detecting Command and Control Communications Associated with Malware,” US9635039B125-Apr-2017 [Online]. Available: <https://patents.google.com/patent/US9635039B1/en>. [Accessed Nov 1, 2019].
- [59] B. E. Strom *et al.*, “Finding Cyber Threats with ATT&CK-Based Analytics.” The MITRE, 07-Jul-2017 [Online]. Available: <https://www.mitre.org/publications/technical-papers/finding-cyber-threats-with-attck-based-analytics>. [Accessed Nov 1, 2019].
- [60] H. Virtanen, “Implementing Automated Log Based Alerts in a Patient Information System,” Degree Programme in Business Information Systems, Tampere University of Applied Sciences, Tampere, Finland, 2017 [Online]. Available: [https://www.theseus.fi/bitstream/handle/10024/128480/Virtanen\\_Henri.pdf?sequence=1&isAllowed=y](https://www.theseus.fi/bitstream/handle/10024/128480/Virtanen_Henri.pdf?sequence=1&isAllowed=y). [Accessed Nov 1, 2019].
- [61] mIRC.co, “About mirc.” [Online]. Available: <https://www.mirc.com/about.html>. [Accessed Nov 1, 2019].
- [62] Pasahitz, “Zirikatu.” [Online]. Available: <https://github.com/pasahitz/zirikatu>. [Accessed Nov 1, 2019].

- [63] Russinovich and Garnier, "Sysmon v11.0." [Online]. Available: <https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon>. [Accessed May 1, 2019].
- [64] SwiftOnSecurity, "sysmon-config." [Online]. Available: <https://github.com/SwiftOnSecurity/sysmon-config/blob/master/sysmonconfig-export.xml>. [Accessed May 1, 2019].