TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Siim Sarv 182517IVCM

USING EVENT CORRELATION TO DETECT SECURITY INCIDENTS FROM WINDOWS WORKSTATIONS

Master's thesis

Supervisor: Risto Vaarandi
Ph.D

Tallinn 2020

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Siim Sarv 182517IVCM

TURVAINTSIDENTIDE TUVASTAMINE WINDOWSI TÖÖJAAMADEST KASUTADES SÜNDMUSTE KORRELATSIOONI

Magistritöö

Juhendaja: Risto Vaarandi
Ph.D

Tallinn 2020

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Siim Sarv

21.12.2020

Abstract

The goal of this thesis is to provide a solution that is able to forward Windows events to a centralised log collection system and analyse the incoming events for any potential security incidents. This solution needs to be done using free, light weight and easy to use tools. It also needs to be integratable to already existing centralised log collection systems.

To achieve the goal, we look at different log collection tools and determine the best tool that matches all our conditions. We also identify what events should be collected from a Windows system. In order to automatically detect potential security incidents from events, we use event correlation tool Simple Event Correlator. In order to create rules for event correlation, we analyse malware that has been popular this year and identify, what kind of events they trigger in a Windows 10 environment. We also create event correlation rules based on the techniques attackers use to compromise systems. The solution will be tested on a production environment.

As a result of the thesis we provide guidelines on how to configure a Windows workstation to provide the most auditing information, configuration for our selected event forwarding tool and rules for event correlation to automatically detect incidents. All this information is available in the appendix and our Github repository.

This thesis is written in English and is 78 pages long, including 4 chapters, 4 figures and 4 tables.

Annotatsioon

Turvaintsidentide tuvastamine Windowsi tööjaamadest kasutades sündmuste korrelatsiooni

Lõputöö eesmärgiks on luua lahendus, mis võimaldaks saata Windowsi sündmuseid tsentraalsesse logide kogumissüsteemi. Lahendus peab olema loodud kasutades tasuta, vähe resurse kasutavaid ja lihsasti implemeteeritavaid vahendeid. Pakutud lahendust peab olema võimalik lihtsalt integreerida olemasolevatesse logide kogumissüsteemidesse.

Tulemuse saavutamiseks me uurime erinevaid logide kogumise lahendusi ja selgitame välja parima tööriista, mis vastab meie nõuetele. Me uurime välja, milliseid sündmuseid peab koguma Windowsi keskkonnast. Turvaintsidentide automaatseks tuvastamiseks me kasutame sündmuste korrelatisooniks tööriista *Simple Event Correlator*. Korrelatsiooni reeglite loomiseks me analüüsime pahavara, mis on olnud populaarne selle aasta jooksul ja vaatame, milliseid sündmuseid nad vallandavad Windows 10 keskkonnas. Lisaks me loome reeglid, põhinedes meetoditele, mida kasutavad ründajad süsteemide nõrgendamisel. Loodud lahendus testitakse produktsiooni keskonnas.

Lõputöö tulemusena me loome suunised, kuidas seadistada Windowsi tööjaamu, et nendest saada võimalikult palju vajalikku informatsiooni; loome seadistused meie poolt valitud tööriistadele ja reeglid sündmuste korrelatsiooniks. Kogu see info on kättesaadav lisadest ja Githubist.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 78 leheküljel, 4 peatükki, 4 joonist, 4 tabelit.

List of abbreviations and terms

DPAPI	Data Protection Application Programming Interface
EQL	Event Query Language
GPO	Group Policy Object
GUI	Graphical User Interface
MPSSVC	Microsoft Protection Service
MS	Microsoft
MSRPC	Microsoft Remote Procedure Call
NAT	Network Address Translation
NIST	National Institute of Standards and Technology
OS	Operating System
OWASP	Open Web Application Security Project
RAT	Remote Access Trojan
SEC	Simple Event Correlator
SECwin	Simple Event Correlator Windows Integration
SIEM	Security Information and Event Management
UEM	Unified endpoint management
VM	Virtual Machine
WEC	Windows Event Collector
WEF	Windows Event Forwarding
XML	Extensible Markup Language

Acknowledgements

I would like to thank my supervisor Risto Vaarandi for providing advice and helpful ideas when researching and making this thesis. I would also like to thank my family and colleagues, who supported me and gave me the time I needed to complete this work.

Table of contents

List of figures

List of tables

1 Introduction

Windows is the most popular OS (Operating System) and has been so for years. Although its popularity is slowly descending, it still makes up 76.32% of worldwide market share [1]. The next closest rival is OS X at 17.65%. In some regions the popularity of Windows is noticeably lower, e.g. United States of America, where Windows makes up only 61,67% of the market, while OS X share is 28,39%. But still there are countries, where Windows is still highly demanded: for example, Estonia, where market share of Windows is 82,68%, while OS X is only 14,08%. In sum, globally Windows still stays at the first place.

Since January of 2018, Windows 10 has been the most popular type of Windows OS [2]. 77.31% of all used Windows desktops worldwide are Windows 10, next closest Windows rival is Windows 7 at 16.8%.

Collecting logging information is critically necessary in order to detect different kind of unlikely events in systems: whether it is hardware or application failure, systems running out of disk space, login failure, connection failure or other events. All of this information must be logged. Logging is not only essential for regular maintenance or up keeping of the systems, but also for detecting security incidents. OWASP (Open Web Application Security Project) ranks insufficient logging and monitoring as 10th on their top 10 vulnerabilities list for 2020 [3].

It is not only important to collect logs, it is also important to collect them centrally. This helps to efficiently manage the collected information and also provides security and integrity of the logs. Even if they are compromised in the original location, they are protected in a centralized logging environment. It may also be important to keep logs for a longer period of time. Centralized solution will provide more control over the duration of time you would wish to keep logs.

Just collecting logs centrally however is not enough. You also need to know what to do with them. Manually analysing log information is a possibility, although centralized log collection systems get a colossal amount of events from different systems.

While a lot of companies collect logs from critical sources, fewer collect centrally from Windows workstations and even fewer regularly monitor these logs [4]. This may be caused by the fact, that Windows generates a lot of logs and collecting all of them in a central location will use a lot of resources. It is more effective to use so called output-based collection. It means that it is impractical to collect logs that have no use, and you should only focus on logs that are useful. Unfortunately, although several research papers and vendor whitepapers provide recommendations on what log data to collect from Windows workstations, these recommendations are often outdated and no longer valid for Windows 10. Also, the information provided by more recent sources provide incomplete advice about what log events are important from a security perspective. One of the major purposes of this work is to analyse all previous recommendations in the context of Windows 10, and provide new recommendations for detecting traces of recent malware samples.

If you only collect logs that you have determined to be useful, you may still miss security incidents that are taking place in your system. It is not that difficult to collect events, but it's more difficult to follow the stream of incoming events and make meaningful decisions based on it. To detect incidents we can use event correlation. When using event correlation, we attach new meaning to certain events that happen in a specific order and in a specific time frame.

There are several tools that allow users to collect event logs from Windows systems. A lot of those tools have free versions and paid extensions and it is often unclear, if free version contains enough features for event collection in Windows. Unfortunately, although there are a number studies on open-source event log collection and correlation tools, these studies have either been conducted for operating systems other than Windows (for example, Linux) or older Windows workstation platforms (for example, Windows XP and Windows 7). Another major purpose of this study is to evaluate open-source event log collection and correlation tools for Windows 10 platform.

First goal of this thesis is to find a free, simple to use and light-weight solution to collect Windows 10 events and send them to a centralised Rsyslog-based log collection system. This will be done by analysing the documentation provided by the developers do determine suitable tools. Suitable tools will be tested while under high load to determine the tool that uses least amount of resources.

Second goal is to identify, what kind of events should be collected from Windows 10 environment. To achieve this we will look at different recommendations from Microsoft and other sources. We will also analyse currently popular malware to see, what kind of events will be triggered. It is also important to determine how a workstation should be configured, in order to give out the most useful information without producing overwhelming amounts of noise.

Third goal is to find a free, simple to use and light-weight tool that allows us to use event correlation to find security incidents from Windows 10 event logs. We will be analysing documents provided by the developers to determine the best tool.

Fourth goal is to create rules for event correlation tool that we have selected, so that the tool it is able to detect security incidents and notify security personnel about the incidents. We will create rules to detect the sequence of events a malware might generate. We will also create rules to detect different kind of attack techniques that can be used by hackers to compromise systems.

One of the overall goals is to keep the solution as simple as possible in order to implement it for already existing centralised systems. For that reason we will be avoiding tools that would need multiple other tools to give us the required functionality.

2 Related work

Antony proposes an output driven collection and analysis strategy. He also describes the usage of whitelist and blacklist to collect Windows event logs. A whitelist is used to specify, what events should be collected. If an event is not on the whitelist, it is discarded. A blacklist is a collection of events that are not useful. If an event is on the blacklist it is discarded [4]. However, he only mentions one tool, *eventlog-to-syslog*, for sending Windows events to syslog based centralised log collection system. *Eventlog-to-syslog* tool is outdated and does not officially support Windows 10 since there have been no updates for years. The paper by Antony covers only 13 events that should be collected from Windows 7 and XP.

Microsoft's own recommendations should also be taken into consideration while making a whitelist and blacklist for the collection system. Microsoft has combined their own list of security related Event ID's [5] [6]. These two lists contain a total of 765 events. Combining those lists and removing the 352 event duplicates leaves us with 391 unique events. We should also take into account the paper published by National Security Agency to help the United States Government and Department of Defence to collect Windows event logs related to different malicious activities [7]. In it they describe how to setup Windows based log collection system and describe what events are useful to discover different malicious activities. Malware Archaeology has created a "cheat sheet" based on the information gathered by MITRE ATT&CK on different methods that malicious actors use to compromise a system [8]. The "cheat sheet" includes sequences of Windows events that happen during different kind of attacks on a system.

Tixteco, Tixteco, Pérez and Medina write about how to analyse incidents that have happened and detect what type of event logs were involved in that [9]. This paper gives procedural advice on how incidents could be analysed to improve your incident detection systems. This paper, however, gives only procedural advice on how to analyse the events and they only use windows event IDs, while in reality it is more beneficial to use event description [4].

Events, related to malware, should also be taken into consideration. It can be difficult to define which events are related to malware. Sainju and Atkins look at 11 different malware in Windows XP and Windows 7 virtual machines. From experiments in the Windows 7 machine, they discover 10 unique event IDs related to tested malware [10]. From those 10 only 1 is on the two Microsoft recommended lists mentioned before. Mullinix analyses Windows event logs that have been generated after infecting the system with malware. From her research and experiments she point out over 30 events that are related to malware in Windows 7 systems [11]. From those 30 events only 4 are events that Microsoft recommends to collect.

Baráth writes about Windows 10 event log analysis and optimisation to reduce the amount of logs generated and get rid of unnecessary logs [12]. Author uses Windows 2012 R2 as log collection server and Event log explorer to analyse and to reduce the list of suggested events to collect.

Events by themselves can have little meaning and just reflect the natural work of the system. The meaning of events can change, when certain events happen together in a specific amount of time. For example, one single failed login attempt can just mean that someone typed their password wrongly. However, if you can see several failed logins in a short amount of time- this can be the indication, that someone is brute forcing the password. To give meaning to sequence of events we can use event correlation.

Vaarandi, Blumbergs and Kont cover the use of SEC (Simple Event Correlator) and LogCluster in a big organization with 543 Linux systems. In this paper, they use LogCluster to generate rules for SEC, but this is done only for syslog messages generated by Linux system. Windows event logs are not covered and it would be valuable to know, if the suggested approach would also work for Windows events. [13].

Petai covers some tools that can be used to monitor, analyse and visualise logs from different systems. In the thesis the author develops a custom tool to detect anomalies from target systems [14]. This thesis however does not cover security incidents and does not cover Windows based system, but rather focuses on UNIX based systems or custom programs and systems.

Gerges takes a look at log monitoring and event correlation [15]. This thesis covers the usage of SECwin (Simple Event Correlator Windows Integration), which is a Windows

application that runs SEC process as a Windows service. This tool is for Windows systems. It means, that if you use a centralised collection system it has to be in Windows based environment.

2.1 Overview of event log collection tools

In this section we will look at several different tools that are commonly used for building log collection systems. Most of these tools have a free and a paid versions. We will be looking at the free versions and if it is possible to use them for Windows event log collection. Tools that we will look at are following:

- NXLog,
- Winlogbeat,
- Rsyslog,
- Snare
- Syslog-ng,
- Solarwinds Eventlog Forwarder,
- Graylog,
- Eventlog-to-syslog,
- Windows Event Forwarding.

NXLog is a multiplatform log collection tool that brings together log processing, filtration, classification, correlation, forwarding and storage in to one tool. It allows several different input and output formats including syslog and Windows Event log. NXLog itself claims, that it can process events from thousands of sources and over 100 000 events per second [16]. It has three modules related to Windows event logs:

- **im_msvistalog** – available for Windows systems only. It can be used to collect information from the local system. Collecting from remote systems requires MSRPC (Microsoft Remote Procedure Call), which is supported in Enterprise Edition of NXLog only. Community version of NXLog, that is free, can collect only locally.
- **im_wseventing** – available for both- Linux and Windows. It can use WEF (windows event forwarding) to collect events, this means it supports agentless

system. It is the recommended module by NXLog but this module is only available in the Enterprise Edition and not in the free version of NXLog.

- **im_mseventlog** – it is for Windows only and is meant to be used to collect logs from older Windows system like Windows XP, Windows 2000, and Windows 2003.

In conclusion, it is possible to use NXLog free version to collect Windows event logs from target systems. For this NXLog needs to be installed in the system you want to collect from. NXLog will run as a service and it can be configured to send event logs to remote central log collection system in syslog format. Configuration of the system is located in a *nxlog.conf* file that can be edited with a text editor with system administrator privileges. For the configuration changes to take effect, the service needs to be restarted. If connection is lost for some reason, NXLog also sends a backlog of events whenever it reconnects.

Winlogbeat is used to send Windows events logs to Elasticsearch or Logstash. It is part of Elasticsearches Beats products [17]. Beats is a collection of tools that allow the user to send data to Elasticsearch [18].

Winlogbeats supports output to Elasticsearch, Logstash, Kafka and Redis. It does not support output directly to syslog. To send winlogbeats output to syslog you need a Logstash as intermediary. Winlogbeats would forward its output to Logstash and Logstash would be able to convert it into syslog and forward it. Winlogbeats is a fine option if you already have a Elasticsearch based system setup. However, if you have central collection system, that is based on some other technology, but is not supported by winlogbeats, you would have to setup Logstash as a middle point.

RSyslog is a syslog-based logging tool able to accept input from many different types of sources, transform them and output them where needed. Its features also include secure transport, output format control, precision timestamps and the ability to filter by any part of the message. It also has a Windows Agent able to filter and forward logs in syslog format to a remote central collection system. While RSyslog itself is free to use, the Windows Agent requires a paid license [19].

Snare agent allows collecting from several different sources including Windows. It has enterprise and open source or lite version. However, the open source version is no longer supported by the developer and does not support Windows 8 and newer [20].

Syslog-ng is a centralised log collection tool that is released in two versions:

- Premium edition – paid version,
- Open source – free version.

Both of the versions support event filtering, event forwarding, secure transfer, encrypted storage, support for several message formats including syslog. There are two ways to use Syslog-ng to collect event logs from Windows [21] [22].

First option is to use Syslog-ng Windows agent to send events to collection server. This will send the events in XML (Extensible Markup Language) format. You can then use Python parser to process the log [22]. Syslog-ng Windows agent is part of the Premium Edition of Syslog-ng. Last version of the agent that was released with Syslog-ng Premium Edition is version 6.0.20. There are no newer versions available.

Second option is using WEC (Windows Event Collector). While using WEC, there is no need to install anything on the Windows machines, whose logs you intent to collect. It uses the Windows event log subscriptions feature in order to collect events and pushes them to Syslog-ng [23]. However, WEC is a Syslog-ng Premium edition feature and therefore requires premium edition licensing.

Event Log Forwarder is a Windows Event forwarder, developed by Solarwinds. This software is an agent. It means, that it is required to be install on every machine, where logs are going to be collected. It allows forwarding of event logs to a syslog server. It is also able to filter by event type (error, warning, information), event ID, user- and computer names. It is possible to use it as a service or with a user interface. The user interface is used to configure the service.

Graylog is an open source log collection tool, that also supports real-time analysis. However, it cannot handle Windows Events on its own and recommends 3rd party collectors, such as NXLog or winlogbeat [24].

Eventlog-to-syslog is an improved version of Curtis Smiths's utility[1](designed to work with Windows NT). It runs as a Windows service. It is light-weight and designed to run on busy servers. It is used to send Windows event logs to UNIX-based syslog servers [25] [26]. While the original utility was designed to work with Windows NT class operating systems and event logs, the new one that is community developed is designed to work with Windows Vista based event logs. Originally, this utility was designed in order to work with Windows NT class Operating systems. Later, community developed version of the utility is able to work with event logs, based on Windows Vista. The last update was 3.10.2013 and there is no official support specifically for Windows 10.

Windows Event Forwarding is a built-in Windows feature to send event logs to a Windows Event Collector. It supports push and pull method for collecting events. To use push method, you initiate event forwarding from the client. You can configure this by using GPO (Group Policy Objects). For pull method, you would need to initiate it from the collector side [27]. In order to use this option you need a Windows Event Collector server, where you have to use other tools that forward logs in Syslog format.

2.2 Malware analysis

In their book, Sikorski and Honig demonstrate the method of analysing malware in order to clarify, what does the malware do, how it manifests and how to deal with it. [28] They cover the usage of virtual testing machines and several tools that can be used to analyse the selected malware, but it is not covered, how to discover malware using Windows event logs.

Robinson in his book [29] describes, how to build a portable and secure virtual lab environment to practice IT and security skills. He covers the usage of both- bare-metal and hosted hypervisors. He covers such tools as: Microsoft Client Hyper-V, Oracle VirtualBox, VMware Fusion Pro, VMware Workstation Pro and VMware vSphere Hypervisor.

[1] https://engineering.purdue.edu/ECN/Resources/Documents/UNIX/evtsys

In order to examine the malware activity we need a way to look at events as they are generated in real time.

Event Viewer is Windows native tool to examine events. By using Event Viewer, you can sort event into following groups: application, security, system etc. You can also see events, that are generated by applications and services. It is possible to create custom views to examine logs from several different sources together, but this feature is limited. After selecting more than 10 different type of logs, it already gives you a warning, that the tool might perform poorly and use large amount of processor time. After selecting all possible sources, Event viewer shows an error, that it cannot display this view. This leads us to a problem, that there is no way to observe real-time events from all sources at once. Basically, if you have created a custom view with some categories, than you will miss the rest of events since the tool is not able to display all of the categories at once.

Although, it is possible, if you use a tool called FullEventLogView. It is a freeware, that is developed by NirSoft. It is a Windows event log viewer that shows events from all sources in one single view. It also allows you to export events in XML format for later examination [30]. FullEventLogView allows you to see logs as they are generated, regardless of the event source.

2.3 Automated detection and notification

Automatic security incident detection from Windows event logs can be done by using event correlation. Jakobson and Weissman define event correlation as a procedure when a new meaning is assigned to a collection of events that happen in a specific time [31]. Although these events could all happen separately without any malicious intent. They become notable, if the events form a pattern and happen during a certain amount of time.

There are several tools that allow us to use event correlation to detect security incidents. We will be looking at the following tools:

- SEC,
- NXLog,
- ESPER,
- ElasticStack,
- Splunk.

The goal of examination of those tools is to find one that is easy to use and implement in an already existing Rsyslog-based log collection system. Secondly, some of these tools have paid versions in addition to free ones, we find it important, that event correlation functionality is available in the free version.

SEC is a free, open-source, rule-based event correlation tool. It is lightweight and it is developed in Perl, making it platform independent. SEC supports processing events from regular files, named pipes and standard inputs. It also supports shell commands as output and regular expressions for pattern matching [32] [33].

NXLog can be used to do event correlation with a module called *pm_evcorr*. It is inspired by SEC. This module, *pm_evcorr*, is a part of the Enterprise Edition feature set and therefore cannot be used for free [16].

Esper is an open-source Complex Event Processing framework. Esper is Java based and there is also a .Net based version called Nesper. It is light-weight, low latency and high throughput solution. Esper allows you to use Event Processing Language to compile executable jar packages [34]. Despite the power of this tool, it is not a simple option: it requires the user to create a solution using its Event Processing Language.

ElasticSearch supports event correlation with the use of EQL (Event Query Language). It allows the user to match sequences of events from different categories and in a certain time span. This is a beta feature in the latest 7.10 version of ElasticSearsh. EQL is not able to compare one field with another field, multi-value fields can return inconsistent results and it cannot search from nested fields [35].

Splunk is a solution that can be used to search, monitor and analyse logs [36]. Splunk allows you to make relations between events, based on time and geolocation. You can group together similar events and use search results in other searches. You can also join together events and correlate data with external sources [37]. It is available as a free version with up to 500MB of data per day [38].

3 Collecting Windows events, analysing malware and automating incident detection

In chapter 3 we will be assembling a collection of Windows event ID, that have been recommended to be collected by Microsoft and other sources. We will also analyse malware that has been common this year. The goal of malware analysis is to detect what events are triggered in the Event log. We will also select a tool and create configuration to collect events from Windows 10 hosts. Lastly we will select an event correlation tool and create rules that will detect security incidents related to malware and attack techniques used by hackers.

By default, Windows has a maximum log size of 20MB for each of the three main categories: application, security and system. Once the limit is reached, the oldest logs will be overwritten. Storing logs by size is good solution, if you would like to regulate the used space. Yet it is not the best option if it is required to keep logs for certain amount of time.

Table 1. Events from Windows workstations.

	Workstation 1			**Workstation 2**		
Event log category	**Number of Events**	**Oldest Event date**	**Date of log file creation**	**Number of Events**	**Oldest Event date**	**Date of log file creation**
Application	30795	21.04.2020	04.12.2019	31825	16.12.2019	26.08.2019
Security	33006	23.10.2020		28480	28.09.2020	
System	57492	01.10.2020		33767	26.08.2019	

In Table 1 we have captured from two different workstations:

- amount of events,
- oldest event date,
- when the event log file was first created.

The data was recorded from both stations at the same time on 29.10.2020. When comparing the data it can be seen that even from two examples the oldest log dates varies by several months. The amount of events also varies a lot. While *Workstation 1* is newer (date of log file creation is later), it has a lot more events stored than the older *Workstation 2*. When forwarding logs to a centralised collection system you have more control over how long you want logs to be stored.

If events are only stored in workstations, there may be no way to analyse them in case of a security incident, because event logs can be removed from workstations by users (if allowed), administrators, malicious actors etc.

In the case of security incidents the targeted systems may also be compromised so that you cannot access them or turn them on any more. In this case, you also cannot examine event logs, unless those have been forwarded to a centralised collection system.

By collecting events centrally you also have the benefit of cross examining events in several systems in order to determine, if similar events (or sequences of events) have also happened in other systems.

Performing normally, Windows can generate thousands of events on daily basis and even more, if the system has been compromised. So it is important to know, what types of events to collect and send to centralised collection system for analysis.

NIST (National Institute of Standards and Technology) categorises logs into three main groups:

- security,
- operation system,
- application [39].

Microsoft Windows also uses the same three main categories, but also has several others. In real situations those categorisations can be misleading. A lot of events that are related to a security incident could be stored all over different categories and that is why it is essential to recognize, which of the events should be collected [40]. It is not always as simple, as collecting events logs that Windows itself categorise as security events.

Collecting all logging information from workstations will lead to enormous volume of events and finding useful information from them will become difficult. We collected all events, that were generated in a week from two Windows 10 workstations. The first one was used daily for the entire week and the second workstation was turned on, but not actively used for the duration of the experiment. From the first workstation we collected 112 125 events and from the second workstation we collected 48 249 events. This means, that in average these two workstations generated around 16 017 and 6 892 events daily. This is why it is best to collect only a selection of useful events that will aid in analysing incidents. Since these events can be in several different log groups, it is best to collect by Event ID or description.

3.1 Collecting event logs

There are many tools for collecting log data from client systems. Some of them are free or have free versions and other require paid licensing to use.

For our purposes, we are looking for tools that are:

- free to use,
- have support for syslog protocol,
- can forward events,
- can filter events,
- support collecting events from Windows 10.

Another important feature of the tools are whether they use an agent in the client systems or not. Typically, there are two kinds of systems:

1. Agent-based that use agents to send data from client system. Event filtering is done by the agent. This leads to less traffic in the network. However, if these systems do not allow remote configuration, changing the filtering rules will require a lot of work.
2. Agentless systems, that use Windows native forwarding features and do filtering at the collection system. This means the clients send more events to the central collection system and generate more network traffic. However, since the filtering of events is done in only one place, changing the rules would be easier.

We are looking for agent-based system to reduce the traffic in the network. This tool should have the possibility to be managed remotely. We also want the tool to be able to send logs directly to a syslog based collection system without additional help of any other systems.

3.1.1 Suitable tools

In this section we will look at the information we gathered in sections 2.1.1 and select the tools, that meet the requirements set in previous section. In Table 2 we have collected the information form section 2.1.1 and cross-referenced it with the requirements we presented.

Table 2. Comparison of tools

Tool	**Free to use**	**Syslog support**	**Forward and filter Windows 10 events**	**Required features available in free version**
NXLog	Yes	Yes	Yes	Yes
Winlogbeat	Yes	Yes[1]	Yes	Yes
Rsyslog	Yes	Yes	Yes	No
Snare	Yes	Yes	No	No
Syslog-ng	Yes	Yes	Yes	No
Solarwinds Event Log Forwarder	Yes	Yes	Yes	Yes
Graylog	Yes	Yes	No	No
Eventlog-to-syslog	Yes	Yes	No	No
Windows Event Forwarding	Yes	No	Yes	Yes

[1] Does not output directly to syslog. Requires the use of Logstash.

From Table 2 we can see, that there are three applications that meet our requirements. Those three are:

- NXLog,
- Winlogbeat,
- Solarwinds Event Forwarder.

NXLog stores its configuration in a file and it runs as a service. With the use of a software, that allows to remotely manage Windows workstations, we can remotely send the targeted machine a new configuration file and restart the service. This allows us to remotely configure the agent.

By default, NXLog does not forward Event IDs, but it is possible to configure it. The tool supports filtering events by event ID, severity, event source, category, time etc.

Winlogbeat is a good option if you have already existing Elasticsearch based log collection system since it requires Logstash to forward events in syslog format. It does not output events in syslog format by itself. If you do not intend to use Elasticsearch based log collection, it would mean that you would need an additional intermediary system that converts Winlogbeat output to syslog. Configuration for Winlogbeats is stored in a file and it runs as a service. This means, it is possible to configure and restart remotely form another system.

We are looking for a tool that can output to syslog directly and using Winlogbeats means, that we would need to setup additional Logstash system to forward the gathered logs to our centralised log collection system. For this reason, Winlogbeats is not the best option for us.

Solarwinds Event Forwarder requires a GUI (Graphical User Interface) to configure. It allows you to configure event forwarding by: event sources, event IDs, keywords, users and computers. This tool also supports both- UDP and TCP. Solarwinds also saves its configuration in a file, so it is also possible to configure Solarwinds remotely. It does not forward event ID and we did not find a way to configure it to be forwarded.

3.1.2 Performance

In this section we will be looking at how much resources the tools (selected in chapter 3.1.1) use, while sending messages under heavy load. We will be sending 200 000 event messages from our test machine. We ran this test 10 times for all of the three selected tools. Test machine is an Oracle Virtualbox Windows 10 machine with 2 CPUs and 4096 MB of RAM. We will be looking at how much RAM and processor time is used to send the 200 000 messages. We will also look at the time of first and last message generation and arrival in the collection server. The collection server is a Solaris based Rsyslog server. It is running on Oracle Virtualbox machine with 1 CPU and 4096 MB of RAM.

To generate Windows event logs you can use PowerShell command *Write-EventLog*. For testing purposes, we have created the script on Figure 1, that generates 200 000 events in the *Application* log category. The message of the event log is "*This is a test event nr X generated at HH:mm:ss.fff"*. *X* is the number of the specific event, *HH* are hours, *mm* are minutes, *ss* are second, and *fff* are milliseconds.

Since Microsoft by default allows you to only run signed scripts, you can use the following command: *Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass* to bypass the execution policy for the current session of PowerShell.

In our generated event we are using source with the name *Test Source* with event ID 1. In order to add this new source, we used the following command *New-EventLog -LogName Application -Source "Test Source"*.

```
#how many events to send
$max = 200000

#Sending a message to user that script has started
Write-Host "Starting..."

#generate $max amount of events
for($i = 1;$i -le $max;$i++ ){

    #time of event
    $time = Get-Date -Format "HH:mm:ss.fff"

    #generating event
    Write-EventLog -LogName Application -Source "Test Source" -EntryType Information `
                -EventID 1 -Message "This is a test event nr $i generated at $time"

    #send message every 10%
    if($i % ($max/10) -eq 0){
        Write-Host "$i sent"
    }
}

#Sending a message to user that script has ended
Write-Host "Done generating $max events"
```

Figure 1. PowerShell script to generate events.

This script can also be accessed from our Github[1] page, that we made for this thesis.

In addition to generating 200 000 events, the script also notifies the user about the start of generation, end of generation, each time 10% from the total amount of messages has been generated.

The tools have been configured to only forward *Application* event logs to the collection server.

Table 3 depicts how much average processor time and RAM each tool uses to send 200 000 events. Processor time shows the percentage of time the process used the processor (the percentage is given for all CPUs) [41]. In a multi-core environment Windows performance monitor show the processor utilization for all cores. For one core it displays utilization in the range of 0-100%. If the machine has for example two cores it

[1] https://github.com/siimsarv/eventlogs

can show up to 200% utilization [42]. So if the Windows performance monitor shows the CPU usage under 100%, it means that the process has utilized only one of the CPU cores.

Table 3. Processor time and RAM usage

Tool	**Processor time**	**RAM**
NXLog	22.6%	2.3MB
Winlogbeat	46.1%	29.4MB
Solarwinds	80.7%	14.1MB

From Table 3 we can see, that NXLog uses the least amount of processor time and RAM. Solarwinds Event Log Forwarder uses the largest amount of processor time and Winlogbeats uses the largest amount of RAM.

In Table 4 we can see, how much time it took to generate 200 000 events and how much time did it take for the events to arrive in our collection server and if there is any delay in the arrival. We ran the test 10 times. All the numbers in Table 4 are in seconds.

Table 4. Message times

		Event generation time	**Event sending time**	**Difference**	**First message delay**	**Last message delay**
NXLog	Minimum	108	108	0	0	0
	Maximum	148	148	0	0	0
	Average	128.8	128.8	0	0	0
Winlogbeat	Minimum	166	166	0	0	0
	Maximum	238	238	0	0	0
	Average	192.1	192.1	0	0	0
Solarwinds	Minimum	147	282	134	2	136
	Maximum	208	352	153	9	158
	Average	159.5	301.7	142.2	5.2	147.4

From Table 4 we can see that for NXLog and Winlogbeats it took the same amount of time to generate the logs as it took for them to arrive in the collection server. For Solarwinds it took in average 142.2 seconds more for the events to arrive in the collection

server then it did for them to generate. This was also observable when doing the test. We could see that in case of NXLog and Winlogbeat the events arrived as fast as they were generated, while for Solarwinds they arrived a little bit later. We can also see from Table 4 that for NXLog the event generation took less time than it did for Winlogbeat and Solarwinds. This is caused by Windows background operation.

3.1.3 Tool selection

From NXlog, Winlogbeat and Solwinds the one that is the best for our needs is NXLog. While the other two also match our requirements, NXlog is the best for us for the following reasons:

- NXLog allows us to forward Windows event logs directly in syslog format, without the need of additional systems to be setup.
- It is possible to configure it remotely. While it does not support this officially it is possible with the aid of UEM (Unified endpoint management) tools.
- It allows a high level of filtering.
- It is lightweight and does not use a lot of resources.

3.1.4 What logs should be collected?

In this section we will be creating a list of event log ID's that should be collected from Windows systems. We will also determine the event log sources for those events. This will help us once we will configure the event log collection. We will do this by combining recommendations from several sources and joining them together into one list. Later we will also add malware related events we identify in section 3.2.2 to the list.

While you could just take one of the tools, that were highlighted in section 3.1.1 and start sending all logs to a centralised system, it is not the best available option. It would work, but you would receive a lot of logs that you do not know what to do with. You could limit those by only forwarding critical or error level events. This would certainly lower the amount of events you would be collecting, but you could miss some important indicators of compromise. For example, if event logs related to Windows Firewall rules are being added, changed or deleted, the events are categorised as informational level by MS. Same goes for events that show the clearing of Windows event logs [7].

Perhaps the hardest part of log collection is determining what events should and should not be collected in order to get the most useful information from thousands of events [27].

To help with this, Microsoft has provided its own recommendations of events, that should be monitored. There are also recommendations from NSA and other sources, that provide their own event logs to monitor. While those different lists often have a several events in common, they also always have unique events, that are not mentioned in other lists. Even lists, that are provided by Microsoft itself, have differences between them. For example, we have got two lists from Microsoft, each of them have 386 events [5] and 379 events [6]. After joining those lists and removing the duplicates, we are left with 391 unique events. The events listed in those sources are mainly in the application, system and security event categories. In reality, there are many other useful categories that should be monitored.

In the paper, published by NSA [7], there are 102 events, that are related to different malicious activities and can be discovered while monitoring those events. From those 102 events only 19 are in the two lists provided by Microsoft. After adding them to existing list of 391 events, that we have got from MS, we now have 474 unique events. While some of the events listed by NSA are from the three main categories (application, system and security), a lot of them are from other log sources.

Malware Archaeology created a "cheat sheet" with sequences of events, that happen if system is being attacked. They use information gathered by MITRE ATT&CK on methods used to compromise systems [8]. To be able to detect these sequences, firstly we would have to collect them. They provide in total 28 individual events and ranges of events related to Task scheduler (event IDs 100-200), PowerShell (event IDs 200-500) and Whitelist failures (event IDs 8000-8027). From those 28 events, 16 was on our already combined list, that leaves us with the total of 486 events to monitor and the three ranges of events, provided in the "cheat sheet".

The same events ID's could appear in several different event log sources. For example Event ID 8000 can be related to *Microsoft-Windows-WLAN-AutoConfig/Operational* logs and *Microsoft-Windows-SoftwareRestrictionPolicies*. To ensure, that collected logs are correct, you could look at logs from all sources and filter only by event ID. However, this would also forward events that we don't need. To prevent this we need to find out event

sources. The NSA list has specified its Event sources, but the other lists do not provide this information. There are several options for identifying the source of the event log. We used different event log knowledge bases to look for the events under question. For example a knowledgebase provided by EventTracker[1].

Event log sources from our unified list are as follows:

- Application,
- Security,
- Setup,
- System,
- Microsoft-Windows-Application-Experience/Program-Inventory,
- Microsoft-Windows-AppLocker/EXE and DLL,
- Microsoft-Windows-AppLocker/MSI and Script,
- Microsoft-Windows-CodeIntegrity/Operational,
- Microsoft-Windows-Kernel-PnP/Device Configuration,
- Microsoft-Windows-NetworkProfile/Operational,
- Microsoft-Windows-PrintService/Operational,
- Microsoft-Windows-PowerShell/Admin,
- Microsoft-Windows-PowerShell/Operational,
- Microsoft-Windows-SoftwareRestrictionPolicies,
- Microsoft-Windows-TaskScheduler/Operational,
- Microsoft-Windows-TerminalServices-LocalSessionManager/Operational,
- Microsoft-Windows-TerminalServices-RemoteConnectionManager/Operational,
- Microsoft-Windows-USBUSBHUB3-Analytic,
- Microsoft-Windows-Windows Defender/Operational,
- Microsoft-Windows-Windows Firewall With Advanced Security/Firewall,
- Microsoft-Windows-WindowsUpdateClient/Operational,
- Microsoft-Windows-WLAN-AutoConfig/Operational.

[1] kb.eventtracker.com

3.2 Malware and event logs.

In order to analyse malware and find out, what events are related to it, we need to run the malware and examine Windows event logs before and after running it. This presents a problem. Running the malware will infect our own systems and may cause irreversible damage. To prevent that, we need a way to run the examined malware safely without putting our production environment at risk.

To analyse malware safely there are some options:

- Physical machine on an isolated network, that is not connected to any outside machine or the Internet – Advantage of this option is that some malware can detect that it is on a virtual network and will act differently. Disadvantage of this option is the complexity of managing the physical environment and additional cost of dedicated hardware [28].
- Virtual machine – Advantage is that you are running the malware on a virtual system and it is harder for the malware to escape the environment and cause damage to the physical machine and other machines on the network. Another advantage is that you can create snapshots of the system, in order to turn back to a previous state. Disadvantage is that some malware can detect that it is running on a virtual system and may act differently. Some may even attempt to escape and cause damage to the physical system that is running the virtual environment [28].

In process of testing malware, it is important to think about how the test machines will connect to each other and to the internet (if needed). For a Virtualized environment there are several options:

- Bridged network – with this option the VM is connected directly to the network by using host machines network card. The VM will have its own IP and will appear in the network as any other connected to the network machine [29].
- NAT (Network Address Translation) networking – with this option the VM and host will share an IP and all the traffic will seem like it's coming from the host. The host will be the only one communicating directly in the network and handle traffic on behalf of the VM. In this case, the VM is not directly connected to the network as in case with bridged network. Instead a virtual network card on the

host will be created and the host will use it when there is traffic that is intended to the VM [29].

- Host-Only networking – with this option the VMs do not get any connection to outside network. They can only communicate with other VM's on the same Host-Only network and the host trough the hypervisor [29].

For our needs it is best to use a virtual environment. The lower setup cost and the ability to use snapshots to revert back to particular points of time are the options that we will need. There are several options for free virtualization software like: Oracle VirtualBox, VMWare player, Microsoft Hyper-V etc. We will be using Oracle VirtualBox, since it is easiest to install and configure.

To isolate the virtual system from rest of the network we will be setting up a Host-Only network between the different VMs. This will allow the virtual machines to communicate with each other, but not the with other computer on the local network. The host can communicate with the VMs only through the virtualization software itself [29].

Since we are interested in events that the malware generates in Windows environment, we will not be doing a deep analysis of the malware. We will look at the events that are generated. We will do this with combination of Windows Event Viewer, Reliability monitor and FullEventLogView.

We will be running our Windows 10 VMs on a Linux Mint host with Oracle Virtualbox virtualization software. Test bench information:

- Linux Mint 20 Cinnamon version 4.6.7
- Linux Kernel 5.4.0-552-generic
- Intel Core i5-4670K @ 3.40GHz x 4
- 16GB of RAM
- Hard Drives – 2 x 1 TB HDD and 1 x 256 GB SSD
- Intel Xeon E3-1200 v3/4th Gen Integrated Graphics Controller and Nvidia GeForce GTX 980 Ti
- Oracle VirtualBox 6.1.10_Ubuntu r138449
- FullEventLogView version 1.57
- 7-Zip version 19.00

Microsoft provides virtual machine versions of Windows 10 Enterprise for free with a trial time. This is meant to be used by developers and comes with some development tools already installed, but this is still suitable for our purposes [43].

We will simply import the virtual machine into our Oracle Virtualbox. We will be setting it to 2 CPU and 4096 MB of RAM just to make the machine a little bit faster. We will also set it to have both NAT and host-only network adapters. We will be using the NAT adapter for the initial setup of the system to install software from the Internet. Once the VM is correctly setup, we will disable the NAT adapter.

3.2.1 Setting up malware test bench.

Our test machine already has a Windows 10 system running on it and we do not want to potentially infect it with any malware. We will install a Linux Mint system as our test environment to host our virtual machines. For the test environment we installed Linux Mint 20 to our workbench. In the Linux we updated the system and made sure there were no driver mismatches and installed VirtualBox. In the VirtualBox we created Windows 10 machines with 4096 RAM and 2 CPUs. In the Windows 10 virtual machine we uninstalled Visual Studio and installed FullEventLogviewer and 7-zip. We also created a runnable commandline file to clear event logs. All events can be cleared with the following command: *wevtutil el | Foreach-Object {wevtutil cl "$_"}* .

3.2.2 Analysing malware

We are basing our malware selection off CheckPoints Global Threath Index. CheckPoint Software Technologies is a security products developer. They develop solutions for network security, cloud security, mobile security etc [44]. They also release a by-monthly report of most popular malware. We will be looking at their reports on the months of January [45], February [46], March [47], April [48], May [49], June [50], July [51], August [52], September [53] and October [54] of 2020. We will be using MalwareBazaar Database to download the selected malware [55]. MalwareBazaar is a publicly available database of malware samples. It is free to use and offers unlimited amount of uploads and downloads of malware samples.

Malware we are going to look at are:

- Agent Tesla – appears in 8 monthly reports,
- Dridex – appears in 7 monthly reports,
- Trickbot – appears in 7 monthly reports,
- Valak – appears in only September report. While it appears only in 1 report it is relatively new malware,
- Frombook – appears in 8 monthly reports
- XMRig – appears in 10 monthly reports,
- Glupteba – appears in 5 monthly reports.

Malware Selection is based on how common it has been this year or how recent it is and if we could find a sample of the malware.

For each tested malware we will use a separate clone of the machine we created in section 3.2.1. We will download a compressed password-protected file to the VM. After that, we will disable the NAT network adapter and take a snapshot of the machine. Snapshot is taken of a powered off VM. After the snapshot had been taken, we will turn on the VM and start FullEventLogView to observe Windows events generated. Then, we will infect the VM with the selected malware. Firstly, we will try to infect the machine with Windows Defender running- just in case the malware has the ability to turn it off. If Windows Defender will prevent the malware from running, we will run the malware again with Windows Defender disabled. By disabling the Defender we will simulate a scenario, when a malware is not yet discoverable by Windows Defender or the Defender has been turned off by the user for some reason. From our test we sometimes detected that the malware was able to trigger some events related to its activity before Windows Defender discovered the malware itself.

To find out what kind of events the test VM generates on its own, we had left an uninfected machine with Windows Defender and NAT adapter disabled running for 4 hours and recorded the generated events.

For the first tests with malware we did not change any logging settings in Windows. We did this in order to see, if it is possible to detect any triggered events with Windows with default logging settings.

To test the selected malware, we let them run in the test environment for several hours while we were monitoring the Event logs in real-time and comparing generated events to the events, that we recorded in uninfected machine.

Agent Tesla is a RAT (Remote Access Trojan) that is designed to steal credentials and sensitive information. It also has the capability to log keystrokes and it collects information about the system it has infected [56].

In our first test with Agent Tesla, Windows Defender was able to successfully detect that a Trojan was run and block it. Then, we reset our machine to previous state (before running the malware) and disabled Windows Defender.

For the tests with default Windows auditing settings we discovered two events that were triggered:

1. Event ID 4797 – Security – An attempt was made to query the existence of blank password for accounts: Adminsitrator, DefaultAccount, Guest, WDAGUtilityAccount
2. Event ID 5381 – Security – Vault credentials were read.

While the two detected events were not found in our uninfected machine, these events can also be seen under normal windows activities. So it is not certain that they were generated by the malware.

Dridex is a banking Trojan, designed to steal banking credentials. It uses form-grapping, clickshot taking and site injection to steal data. It can also change the content of webpages the user is viewing [57].

For the first tests with Dridex we did not find any Windows events that were triggered by the malware.

Trickbot is another banking Trojan designed to steal users bank credentials. It uses redirection to change the targeted webpage that the user is trying to connect to. It is also able to steal users browser history and interject payment process to steal money. It also utilises the EternalBlue exploit to spread through the network [58].

For the first test with default Windows logging settings we were not able to find any events triggered by Trickbot.

Valak was originally classified as malware loader, but after its first appearance in 2019, it has had a lot of different versions. Now it can be used on its own to steal information [59].

While all other malware examples that we found had an executable *.exe* version available, Valak had only MS Word file and a *.dll.* The malware uses *regsvr32.exe* to run the *.dll* file [59], we will do the same.

While running just the *.dll* file, we were not able to detect any suspicious activity in the event logs.

Formbook is an information stealing malware, that has been offered as a service. Its information stealing ability is not the best in the market but because it is easy to use it is still very popular malware [60].

No events related to Formbook discovered in Windows 10 workstation with default auditing configuration.

XMRig is a crypto mining software often installed on target machine after infecting it with other malware [61].

We did not find any events related to XMRig in a Windows 10 workstation with default auditing configuration.

Glupteba is used to install other malware, but also has information stealing functionality [62].

No events related to Glupteba discovered.

Microsoft has its own *Audit Policy Recommendations* [63]. Since we only found two events from our tests with default Windows logging setting, we will now apply settings, that are recommended by Microsoft, and will test the same malware again. You can access Audit Policy setting by navigating to *Administrative tools then Local Security Policy* and afterwards to *Advanced Audit Policy Configuration*. We will configure settings to recommended by Microsoft as baseline and test the malware again. Before testing

malware, we will let the machine run without infection in order to get the normal events. We will test the same malware again and look for new event ID's that are generated by malware.

You can set to log both- success and failure- of certain actions in Windows. Baseline audit policy settings recommended by Microsoft that we applied [63]:

- Credential Validation – success,
- Computer Account Management – success,
- Other Account Management Events – success,
- Security Group Management – success,
- User Account Management – success,
- Process Creation – success,
- Logoff – success,
- Logon – success and failure,
- Special Logon – success,
- Audit Policy Change – success and failure,
- Authentication Policy Change – success,
- IPsec Driver – success and failure,
- Security State Change – success and failure,
- Security State Extension – success and failure,
- System Integrity – success and failure.

After making these changes we were able to observe a lot more of events for tested malware samples.

Events triggered by Agent Tesla:

- Event ID 4688 – Security – A New process was created. – Agent Tesla starts C:\Windwos\SysWOW64\schtasks.exe with new process ID 0xf38.
- Event ID 4688 – Security – A new process has been created. – schtasks.exe with process ID 0xf38 starts C:\Windows\System32\conhost.exe with new process ID 0x1504.
- Event ID 4688 – Security – A new process has been created. – Agent Tesla starts C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegSvcs.exe with new process ID 0x1cc8.

- Event ID 4688 – Security – A new process has been created. – Agent Tesla starts itself with new process ID 0x159c.

Events triggered by Dridex:

- Only event found was us launching the malware.
- Even though Windwos Defender's real-time protection was disabled it found the malware 25 minutes after it was launched.
- Event ID 1116 – Microsoft-Windows-Defender/Operational – Microsoft Defender Antivirus has detected malware or other potential unwanted software.

Even though we did not find any events related directly to the malware, we still saw event ID 1116. This event was triggered because Windows Defender discovered the malware. This event ID was not in any of the lists we looked at in section 3.1.2.

Events triggered by Trickbot:

- Event ID 4688 – Security – A new process has been created. – Trickbot starts with new process ID 0xa30.
- Event ID 4688 – Security – A new process has been created. – C:\Windows\SysWOW64\dllhost.exe with process id 0xf1c starts new Trickbot process with process ID 0xf20.
- Event ID 4688 – Security – A new process has been created. – Trickbot starts C:\Windows\System\wermgr.exe with process ID 0x78c.
- Event ID 4688 – Security – A new process has been created. – C:\Windows\System\svchost.exe with process ID 0x43c starts Trickbot from C:\ProgramData\Microsoft\Windows\Start Menu\Programs\WinPwrSvs\trickbot.exe (not the original Trickbot location were we started it) with process ID 0x1830.
- Event ID 4688 – Security – A new process has been created. – Trickbot with process ID 0x1830 starts C:\Windows\System32\wermgr.exe with process ID 0x1624.
- Event ID 4688 – Security – A new process has been created. – C:\Windows\System\svchost.exe with process ID 0x43c starts Trickbot from C:\ProgramData\Microsoft\Windows\Start Menu\Programs\WinPwrSvs\trickbot.exe with process ID 0x578.

- Event ID 4688 – Security – A new process has been created. – Trickbot with process ID 0x578 starts C:\Windows\System32\wermgr.exe with process ID 0x16d4.
- Last 4 events keep repeating it self. Svchost.exe starts Trickbot and Trickbot starts wermgr.exe.

Events triggered by Valak:

- Event ID 4688 – Security – A new process has been created. – We start C:\Windwos\System32\regsrv32.exe with Valak.dll as target. New process ID 0x1708
- Event ID 4688 – Security – A new process has been created. – C:\Windows\System32\regsrv32.exe with process ID 0x1708 start new process C:\Windwos\SysWOW64\regsvr32.exe with new process ID 0x1058
- Event ID 4688 – Security – A new process has been created. – C:\Windwos\SysWOW64\regsrv32.exe with process ID 0x1058 starts C:\Windows\SysWOW64\wscript.exe with new process ID 0x19a0.

Events triggered by Formbook:

- Event ID 4688 – Security – A new process has been created. – Formbook with process ID 0x3a0 starts C:\Windows\SysWOW64\schtask.exe.
- Event ID 4688 – Security – A new process has been created. – schtask.exe with process ID 0x3a0 opens C:\Windows\System32\conhost.exe with new process ID 0x1e8c.
- Event ID 4688 – Security – A new process has been created. – Formbook with process ID 0x16d0 opens Formbook with new process ID 0x3b4.

Events triggered by XMRig:

- Event ID 4688 – Security – A new process has been created. – XMRig with process ID 0x15e0 starts itself with new process ID 0x1764.
- Event ID 4688 – Security – A new process has been created. – XMRig with process ID 0x1764 starts C:\Windows\notepad.exe with new process ID 0x668.

- Event ID 4688 – Security – A new process has been created. – XMRig with process ID 0x1764 starts C:\Windows\SysWOW64\cmd.exe with new Process ID 0x654.
- Event ID 4688 – Security – A new process has been created. – C:\Windows\SysWOW65\cmd.exe with process if 0x654 starts C:\Windows\System32\conhost.exe with new process ID 0xfe4.
- Event ID 4688 – Security – A new process has been created. – C:\Windows\SysWOW65\cmd.exe with process if 0x654 starts C:\Windows\System32\wscript.exe with new process ID 0xfe0.
- Event ID 4688 – Security – A new process has been created. – XMRig with process ID 0x1764 starts C:\Windows\notepad.exe with new process ID 0x13fc. This events repeats several times. XMRig opens several new processes of notepad.exe.

Events triggered by Glupteba:

- Event ID 4798 – Security – A user's local group memvbership was enumerated by Glupteba with process id 0x16c4.
- Event ID 4688 – Security – A new process has been created. – Glupteba with process ID 0x16c4 starts c:\Windows\System32\cmd.exe with new process ID 0x1d68
- Event ID 4688 – Security – A new process has been created. – C:\Windows\System32\cmd.exe with process ID 0x1d68 starts C:\Windows\System32\conhost.exe with new process ID 0x1e68.
- Event ID 4688 – Security – A new process has been created. – C:\Windows\System32\cmd.exe starts with process ID 0x1d68 starts new process C:\Windows\Sustem32\fodhelper.exe with new process ID 0x1454. This event occurs several times.
- Event ID 4688 – Security – A new process has been created. – C:\Windows\System32\fodhelper.exe with process ID 0x1454 starts Glupteba with new process ID 0x1cf4.
- Event ID 4688 – Security – A new process has been created. – Glupteba with process ID 0x16c4 starts C:\Windows\SysWOW64\WerFault.exe with new process ID 0x1ce0. This event occurs several times.

- Event ID 4688 – Security – A new process has been created. – Glupteba starts itself with new process ID 0x1cf4.

After configuring the VMs according to Microsoft baseline recommendations, we can already see a lot more information than we did before. We were able to detect processes, that are created by the malware, and follow them for some time. The setting that allowed us to achieve that was enabling logging for *Process Creation* successes.

We will now test the same malware with stronger audit policy setting recommended by Microsoft [63]. We will first configure the audit settings and let an uninfected VM with NAT and Windows defender disabled run to get the events that would be normal for our machine. We will then again test the malware and compare with our uninfected events to see what are triggered by malware.

Strong audit policy settings recommended by Microsoft that we applied [63]:

- Credential Validation – success and failure,
- Kerberos Authentication Service – success and failure,
- Kerberos Service Ticket Operations – success and failure,
- Other Account Logon Events – success and failure,
- Computer Account Management – success and failure,
- Other Account Management Events – success and failure,
- Security Group Management – success and failure,
- User Account Management – success and failure,
- DPAPI(Data Protection Application Programming Interface) Activity – success and failure,
- Process Creation – success and failure,
- Account Lockout – success and failure,
- Logoff – success,
- Logon – success and failure,
- Special Logon – success and failure,
- Audit Policy Change – success and failure,
- Authentication Policy Change – success and failure,
- MPSSVC (Microsoft Protection Service) Rule-Level Policy Change – success,
- IPsec Driver – success and failure,

- Security State Change – success and failure,
- Security State Extension – success and failure,
- System Integrity – success and failure.

After we had configured the above setting on our VM, we ran the tests again and we were not able to detect any additional events, that were triggered by the selected malware. We were able to detect the same events, that were triggered in the previous test with baseline audit policy settings.

As we can see, the most useful audit policy setting for us was *Process Creation* setting. This allows us to follow the malware and see what processes it triggered. After *Process Creation* setting is set to log successes, it will trigger an event with ID 4688. This event logs creation of a process in Windows and it also shows what program started this.

From the tests that we have done so far, we can see, that some of the malware started *conhost.exe* and *schtasks.exe*, but we could not see, what commands were used with those commands. *Conhost.exe* is a Command Prompt and *schtasks.exe* is Scheduled Tasks command line utility.

For our next tests we will be enabling Command Line and PowerShell logging. We will also enable Scheduled Tasks logging. To do this we can use command line to run the following commands [64].

Command to enable Task Scheduler logging is displayed on Figure 2:

```
wevtutil sl "Microsoft-Windows-TaskScheduler/Operational" /e:true
```

Figure 2. Command to enable Scheduled tasks logging

Enabling command line logging makes *Process Creation* event with ID 4688 more detailed by adding *process command line* to the log. To enable Command Line we can use the command in Figure 3. Note that the command is a single line command.

```
reg add "hklm\software\microsoft\windows\currentversion\policies\system\audit"
/v ProcessCreationIncludeCmdLine_Enabled /t REG_DWORD /d 1
```

Figure 3. Command to enable Command Line logging

To enable PowerShell logging we can use the commands in Figure 4. Note that all the commands are single line commands.

```
reg add "hklm\Software\Policies\Microsoft\Windows\PowerShell\ModuleLogging"
/v EnableModuleLogging /t REG_DWORD /d 1

reg add "hklm\Software\Policies\Microsoft\Windows\PowerShell\ScriptBlockLogging"
/v EnableScriptBlockLogging /t REG_DWORD /d 1

reg add "hklm\Software\Policies\Microsoft\Windows\PowerShell\Transcription"
/v EnableInvocationHeader /t REG_DWORD /d 1

reg add "hklm\Software\Policies\Microsoft\Windows\PowerShell\Transcription"
/v EnableTranscripting /t REG_DWORD /d 1

reg add "hklm\Software\Policies\Microsoft\Windows\PowerShell\Transcription"
/v OutputDirectory /t REG_SZ /d D:\PS_Transcripts
```

Figure 4. Commands to enable PowerShell logging.

After we have implemented these logging settings we will test all the malware again to see, if we discover any new evidence. We will be recording new events or events that have new information.

Events triggered by Agent Tesla:

- Event ID 4688 – Security – A new process has been created. – Agent Tesla starts new process C:\Windows\System32\schtasks.exe /Create /TN Updates\YpPDUr /XML C:\Users\User\AppData\Local\Temp\tmpF091.tmp with process ID 0xdcc.
- Event ID 4688 – Security – A new process has been created. – schtasks.exe with process ID 0xdcc starts C:\Windows\System32\conhost.exe 0xffffffff –ForceV1 with process ID 0x804.
- Event ID 106 – Microsoft-Windows-TaskScheduler/Operational – User "WINDEV2010WVAL\User" registered Task Scheduler task "\Updates\YpPDUr".
- Event ID 4688 – Security – A new process has been created. – Agent Tesla start C:\Windwos\Microsoft.NET\Framework\v4.0.30319\RegSvcs.exe with process if 0x1534 with argument "{path}". The *path* argument value is not known.

We can see, that adding command line logging to Process creation has given us more information. We see, what commands were used to start *schtasks.exe* and see what task was created.

Events triggered by Dridex:

- No new events discovered.

Events triggered by Trickbot:

- Event ID 4688 – Security – A new process has been created. – C:\Windwos\System32\svchost.exe starts new process C:\Windows\SysWOW64\DllHost.exe /Processid:{3E5FC7F9-9A51-4367-9063-A120244FBEC7} with process ID 0x1730.
- Event ID 106 – Microsoft-Windows-TaskScheduler/Operational – User "SYSTEM" registers Task Scheduler task "\Windows Power Saves".
- Event ID 140 – Microsoft-Windows-TaskScheduler/Operational – User "SYSTEM" updated Task Scheduler task "\Windows Power Saves".
- Event ID 129 – Microsoft-Windows-TaskScheduler/Operational – Task Scheduler launches task "\Windows Power Saves" , instance "C:\ProgramData\Microsoft\Windows\Start Menu\Programs\WinPwrSvs\trickbot.exe" with process ID7392'.
- Event ID 100 – Microsoft-Windows-TaskScheduler/Operational – Task Scheduler started "{3d548fef-1f94-4105-acd5-df36bb461ddd}" instance of the "\Windows Power Saves" task for user "NT AUTHORITY\SYSTEM".
- Event ID 200 – Microsoft-Windows-TaskScheduler/Operational – Task Scheduler launched action "C:\ProgramData\Microsoft\Windows\Start Menu\Programs\WinPwrSvs.exe" in instance "{3d548fef-1f94-4105-acd5-df36bb461ddd}" of task "\Windows Power Saves".
- Event ID 201 – Microsoft-Windows-TaskScheduler/Operational – Task Scheduler successfully completed task "\Windwos Power Saves", instance "{3d548fef-1f94-4105-acd5-df36bb461ddd}" , action "C:\ProgramData\Microsoft\Windows\Start Menu\Programs\WinPwrSvs\trickbot.exe" with return code 0.
- This activity repeats itself.

Events triggered by Valak:

- Event ID 4688 – Security – A new Process has been created. – C:-\Windwos\SysWOW64\regsvr32.exe starts new process with following parameters C:\Windwos\SysWOW64\wscript.exe //E:jscript "C:\Public\IdfxcyvPN.N_Dya" with process ID 0x1118.

Events triggered by Formbook:

- Event ID 4688 – Security – A new Process has been created. – Formbook starts C:\Windows\SysWOW64\schtasks.exe with commands */Create /TN "Updates\<random name> /XML <location of .tmp file>".*
- Event ID 106 – Microsoft-Windows-TaskScheduler – A task with the same name as in previous event is created.

This malware acts similarly to Agent Tesla. It creates scheduled task with similar name pattern. Agent tesla also starts *regsvcs.exe*.

Events triggered by XMRig:

- Event ID 4688 – Security – A new Process has been created. – XMRig starts cmd.exe wist command line *"/C WScript C:\ProgramData\<random name>\<filename>.vbs"*
- Event ID 4688 – Security – A new Process has been created. – cmd.exe starts C:\Windows\SysWOW64\wscript.exe with command line *"WScript C:\ProgramData\<random name>\<filename>.vbs"*
- Event ID 4688 – Security – A new Process has been created. – XMRig starts notepad several timed with command line *"-c C:\ProgramData\<random name>\<filename>"*

Events triggered by Glupteba:

- Event ID 4688 – Security – A new Process has been created. – Glupteba starts cmd.exe with command line *"C:\Windows\Sysnative\cmd.exe /C fodhelper"*
- Event ID 4688 – Security – A new Process has been created. – GLupteba starts C:\Windwos\SysWOW64\WerFault.exe this parameters *"-u –p <number> -s <number>"*

From tests results with Command Line , PowerShell and Task Scheduler monitoring we can see that we got a lot more information about the malware activities. We also managed to discover some new Windows event ID that were not in any of the lists we looked at chapter 3.1.2. Those events are 100, 104,129, 140, 200 and 201. Event source for those events is *Microsoft-Windows-TaskScheduler/Operational*.

For Dridex we were also able to get a working Microsoft Word .doc version sample. This Word document would be sent to a target in form of some important document that has to be opened. After opening the document, user is asked to enable editing of the document in order to view it. After opening the document, we were able to identify following events from Windows events logs:

- Event ID 600 – Windows Powershell – Powershell is started with *–w hidden –ENCOD JAAwADYA….* This is continues as a long encoded message.
- This was done 6 times for providers: Varriable, Alias, Environment, Function and Filesystem. Lastly Engine state was set from *None* to *Available.*
- Event ID 4104 – Microsoft-Windows-PowerShell/Operational – Creat Scriptblock text … - Obfuscated peace of code was executed.
- This event was done 3 times with different obfuscated code.

During the testing we also recorded what kind of events are generated, if Windows Defender is turned off, and when Windows Defender successfully discovers malware. Following events were recorded:

- Event ID 15 – SecurityCenter – Updated Windows Defender status successfully to SECUTIY_PRODUVT_STATE_SNOOZED.
- Event ID 1116 – Microsoft-Windows-Windows_Defender – Microsoft Defender Antivirus has detected malware or other potential unwanted software.

Before every test we cleared event logs with a command mentioned before. This also generates events into the event log. While these events were not identified in our Windwos machines, we did see them in our centralised collection system. Following events were recorded:

- Event ID 1512 – Microsoft-Windows-Eventlog – The *name of the event log* file was cleared.

3.3 Configuring centralized collection

In this section we will give recommendation for configuring the workstation and explain the *nxlog.conf* file we created to send all the events that we have gathered in section 3.1.4 and 3.2.2. The full NXLog configuration can be seen in Appendix 1 and also Github[1] page that is made for this thesis.

For Windows workstation we would recommend activating the baseline setting from Microsoft [63] and adding command line, PowerShell and task scheduler logging. From the baseline setting *Audit Process Creation* events were the most useful for us and provided us with the most information. Location of files and command run was mostly gathered from the command line and task scheduler logs.

We also created a NXLog configuration file to collect recommended events.

First we defined variables called *EventsToCollect* and *MalwareEvents*. This events contain all the events IDs we gathered from sections 3.1.4 and 3.2.2. *EventsToCollect* has all the events we combined in section 3.1.4 and *MalwareEvents* has all events that were not in *EventsToCollect* and we discovered testing malware in section 3.2.2.

We created three *input* sections called *eventlog*, *eventlogSC* and *eventlogPWR*. The first one has all the events gathered from malware and section 3.1.4. It does not have the range of events for TaskScheduler(Event IDs 100-200) and PowerShell(Event IDs 200 - 500). *Eventlog* also has all the eventlog sources we discovered in section 3.1.4.

While configuring event source paths, you need to make sure, that the path exist in your system and that the path is called the same internally as it is shown by Microsoft. The event source for *Kernel Device PnP* configuration in Windows Event viewer is *Microsoft-Windows-Kernel-PnP/Device Configuration*, but in reality the location path internally is *Microsoft-Windows-Kernel-PnP/Configuration*.

[1] https://github.com/siimsarv/eventlogs

At the end of *eventlog* input section there is *if statement*, that is used to check, if the event ID that NXLog is currently processes does not exist in *EventToCollect* and *MalwareEvents* cariable. If they do not exist, the processed event is dropped. After the *if statement* we add the processed event's event ID to the event message.

For input sections *eventlogSC* and *eventlogPWR* we are only are only looking at three event sources. Task Scheduler event source for *eventlogSC* and PowerShell event source for *eventlogPWR*. For Task Scheduler and PowerShell we have a range of event IDs that interest us. 100 to 200 for Task Scheduler and 200 to 500 for PowerShell. We check for the event ID with an *if statement* in the end of the input sections.

3.4 Automatic detection of security incidents

For our purposes the best option is to use SEC for automated detection of security incidents. As for other four solutions from section 2.3:

- NXLog *pm_corr* is not free solution.
- ESPER is not a readily available tool but rather a library which requires the development of an event correlation solution around it.
- ElastiSearch and Splunk are heavyweight solutions which have to be deployed on a separate infrastructure.

SEC is lightweight option that does not use a lot of resources. It is open-source and free to use. It is also simple to use and does not require us to learn a completely new programming language. SEC does event matching with the use of regular expressions.

In chapter 3.2 we looked at several different malware and what kind of events they trigger, we also noted, what kind of events are triggered by Windows Defender if it is disabled or it discovers malware. We also looked at what kind of events are triggered if Windows event logs are cleared. Now we will create SEC rules to detect these events automatically. For this we will be analysing the events generated and determine, what information can be used to create regular expressions, that would match the events. In the second half of this chapter we will also create rules that will discover common attack techniques used by attacker. For this we will analyse Windows ATT&CK cheat sheet [64] for common

tactics used. All SEC rules that we created can be found in Appendix 2 and 3 and also in Github[1].

Events, we will be looking at, are:

- Windows Defender turned off,
- Windows Defender discovered malware,
- Windows Event logs cleared,
- potentially malicious PowerShell commands,
- events triggered by malware,
- events triggered when system is under attack.

Security Centre regularly generates events that show the status of Anti-Malware software. These events come from event source *SecurityCenter* with the event ID 15. The event message contains defender status *ON* if it is turned on and *SNOOZED* if it is turned off. For the SEC rule that checks for Windows Defender status we created a regular expression that checks for the source *SecurityCenter* and that the event message contains *SECURITY_PRODUCT_STATE_SNOOZED.* If an event that has these elements is matched we send a mail using *mailx* to desired address and also log it in another log.

When we use SEC to send message to the user or logging an event to a separate log, we also always include the hostname of the workstation, where the incident was discovered on.

When Windows Defender discovers malware, it generates an event with the event source *Microsoft-Windows-Windows_Defender*. The event message contains *has detected malware* and the name, path and process of the malware. To discover this with SEC, we created regular expresion that matches the source and the message. We also retrieve the name of the malware, path and process. We then send this information to the desired address and also log it.

Clearing an event log in windows generates an event itself. The event source of these messages is *Microsoft-Windows-Eventlog*. The message itself contains the name of the log that was cleared. If a malicious actor would also clear Windows Eventlog log, then

[1] https://github.com/siimsarv/eventlogs

these messages would be also gone. Leaving the user with no indication that any events were cleared, other than the fact that there are no logs. However, if you would forward logs to a log collection system, then these logs would still be available. To match these events with SEC, we are checking for the event source and we also retrieve the event log name that was cleared. We created two SEC rules for event log clearing. The first one sends a mail each time a log is cleared, the second one sends a mail every 60 seconds after the first log is cleared. The mail contains all logs that have been cleared during the 60 seconds. It will stop sending mails if there are no more events after the last 60 seconds is over. We achieve this by using SECs *context* feature.

From our experiments with malware, that used Microsoft Word to infect the target, we discovered that it used PowerShell to run encoded commands in a hidden window. After running the encoded PowerShell command we also noticed an event for scriptblock creation. The event related to the hidden and encoded PowerShell is from *Microsoft-Security-Auditing* event source. The *–w hidden* and *ENCOD* can be seen in the *Process Command Line* section of the message. This section also contains the encoded code. The SEC rule, that we created in order to detect this event, looks for the event source and that the message contains "*A new process has been created"* and that the new process name is either *cmd.exe* or *powershell.exe*. We also retrieve the process that started this new process, and send it to the user.

Scriptblock creation events are from *Microsoft-Windows-PowerShell* event source and in the event message have *Creating Scriptblock* and then the script that was used to create the scriptblock. To detect this with SEC we are looking for events that have the correct evnet source and message. We then send this information to the user.

To discover Agent Tesla we need to look at several different events that happen in a specific order. These events are:

1. Some process starts *schtasks.exe* with a command to create a new task with a name that follows this pattern *\Updates\<name>*. The <name> field contains of random letters and numbers.
2. That same *schtask.exe* that was started in step 1 then starts *conhost.exe*.
3. A scheduled task with the name used in step 1 is created.

The first and second event are from the same events source of *Microsoft-Security-Auditing* and the last message is from *Microsoft-Windows-TaskScheduler*. From the first event we are retrieving the created process ID and the task name. In order to pass this information to the rules to detect step 2 and 3 we are using SEC *context* feature. With the first rule, we are creating a context with the hostname, the process ID and another context with the hostname and task name. In the second rule we are comparing the creator process ID with the one that was discovered in the first step. In the third rule we are comparing the task names. Once the third step has successfully identified the correct event it sends a mail to the desired location.

To discover Trickbot we have to look at 5 events that happen in a specific order. These events are:

1. *Svchost.exe* starts *dllhost* with process command line *DllHost.exe*
2. *Dllhost.exe* starts the trickbot file.
3. Trickbot start *wermgr.exe*
4. A task is created with the name of "Windows Power Saves"
5. A task named "Windows Power Saves" starts trickbot from a new location.

The steps 1, 2 and 3 all have the same event source of *Microsoft-Security-Auditing* and the steps 4 and 5 have event source of *Microsoft-Windows-TaskScheduler*. We are again using *context* to pass the required information to the next rule and to ensure that these actions happen in a specific order. For the first event we are checking if the event source is correct and that the creator process is *svchost* and new process is *dllhost*, we are then retrieving the new process ID. For the second rule we are checking, if the creator process id *dllhost* and the process ID is the same as in previous step. We then collect the new process name and ID. For the third step we are looking if the process from step 2 has started *wermgr.exe*, if we detect this we are sending a mail to the specified email with the original location of the malware. The malware location is the same as new process in step 2 and creator process in step 3. We then look for *TaskSheheduler* event that show us the creation of a task named "Windows Power Saves". In the final step we look for a task with the same name that starts a process. This process is the new location for Trickbot. We send this information also to the target email.

To discover Valak, we are looking for events with event source *Microsoft-Windows-Security-Auditing* and if *regsvr32.exe* is starting *wscript.exe*.

To discover XMRig, we searched for processes that start itself and then start *notepad.exe* and *cmd.exe*. When a process starts itself, we record the new process ID and look for events where that same event ID starts notepad and cmd. We also record the command lines of the notepad and cmd events.

To discover Glupteba we are first looking for processes that start *cmd.exe* with command line parameter *fodhelper* and capture the process ID. Then we look for *cmd.exe* processes with the same ID, that start *fodhelper.exe* and capture the new process ID. After that, we look for events were *fodhelper* with the same ID starts something else. *Fodhelper* starts the original malware file again. We capture the new process ID and look for the original malware file starting new process *WerFault.exe*. We then send a notification email to the user with the location of the original malware.

The *cheat sheet* made by Malware Archaeology [8] covers different techniques that attackers use to compromise systems. This *cheat sheet* is based on MITRE ATT&CK [65] research. Malware Archaeology as has combined the research into an easy to follow table that show the sequence of events for different techniques. These techniques include: reconnaissance, collection, lateral movement, credential access, execution etc.

The *cheat sheet* is divided by tactics and techniques. For each technique there are 1 to 7 data sources that show what kind of information should be examined to discover this technique. It does also show if the coverage of a specific technique is good, not complete or none at all. The data sources that have a Windows event related to them also show an event ID of the related event.

We are going to look at the techniques that have good coverage and that all the data sources have event related to them. From those techniques we select the ones that have at least three data sources or have unique data sources. We do this to prevent possible false positives. For example, consider technique that has only two data sources process execution and process command line. In that case, it is not possible to ensure that we have discovered the specific technique, since these kind of events happen often in a Windows environment and are also often the first two data sources of other techniques.

Techniques that match our criteria are:

- Collection – data from local system,
- Collection – data from network shared drive,
- Collection – data from removable media,
- Collection – data Staged
- Credential access – Brute force,
- Defence evasion – deobfuscate code,
- Defence evasion – network share connection removal,
- Discovery – system network configuration discovery,
- Discovery – system ownership discovery,
- Execution – PowerShell
- Execution – service execution,
- Execution/ lateral movement – windows remote management,
- Exfiltration – automated exfiltration,
- Lateral movement – application deployment software,
- Persistence – modify existing service.

Most of the techniques, that we chose to be automated, start with data sources: *process execution* and *process command line*. These data sources are both related to event ID 4688. In some techniques *process execution* is the first data source and *process command line* is the second, in others it is the other way around. Event with the ID 4688 is regular process creation event and it appears often in Windows environment. In order to make sure that the two events are related to each other, we have to make sure that the first events *new process ID* matches the second events creator process ID. We also need to make sure that *command line* events include *cmd.exe* as creator process and that *process execution* events do not have *cmd.exe* as creator process. Once two events matching these rules have been discovered by SEC we create a context name *exec_cmd* or *cmd_exec*.

To discover specific techniques we check the existence of the correct context to match the first two data sources and for the other data sources we just check, if the correct event ID exists in a specific amount of time. For example, to discover collection of data from local system we first check for the presence of *exec_cmd* context. If this is present, we check for *PowerShell* events with the event IDs of 200 to 500 or 4100 to 4104. If an event appears, that matches those PowerShell events, we wait for another event with the ID of

4663 and after that we look for an event with ID of 5861. If this sequence of events appears, we send an email to the desired location to notify the user.

The rules that we created to detect the techniques described in the *cheat sheet* are included in Appendix 3 and in Github1.

While we performed test to ensure, if the rules that we created for SEC worked correctly, we modified the same testing environment we used to analyse malware. We configured it to forward its events to a Rsyslog and Solaris based centralised collection system. We used NXLog with configuration we made in section 3.3 to forward the events from Windows. SEC was configured on the centralised collection system. In order to test, if the rules to discover malware worked correctly, we activated each malware separately in the Windows environment and observed SEC logs. We were able to confirm, that all the rules we created for malware detection, worked correctly. To test the rules we created based on the *cheat sheet*, we modified the PowerShell script that we used to generate specific events. For each technique, we modified the script to generate events, according to the sequence specified in the *cheat sheet*. We were able to successfully discover all the techniques that we created rules for.

3.5 Testing on production environment and results

In order to test how the proposed solution performs in a production environment with real data, we implemented our solution to an already existing rsyslog based event collection system. For the initial test we configured 10 Windows 10 workstations to forward their events to the collection system. We run this test for 6 days, during this time those 10 workstations sent 953044 events. The test run for a total of 521971 seconds which makes it an average of 1.9 events per second.

We also monitored how much processor time was used by NXLog, in the workstations and by SEC in the collection server. The workstations had a 4 core Intel processor and NXLog used an average of 1.5% of one of the CPU cores. The server with SEC had 8 virtual CPUs and SEC used an average of 0.52% of one CPU out of 8. These result show that collection of events and monitoring requires a modest amount of CPU resources, and

[1] https://github.com/siimsarv/eventlogs

the solution can process event log data from a much larger number of workstations. We plan to extend our solution to 300 workstations in the institution where the experiment was conducted.

From the 10 workstations that we used in the test 9 of them had their auditing settings set according to our recommendations and they produces on average 14 000 events per day. The 1 workstation what was not set according to our recommendations was set to send auditing information about everything that happens in the machine. This workstation sent on average 273 000 events per day. This was on average about 300 MB of storage per day compared to an average of 10 MB per day on the other machines. These findings illustrate that if the collection of events relevant to security is not properly configured on Windows workstations, the amount of event log data arriving to central log collection server can increase significantly and easily consume 30 times more disk space. Therefore, it is essential to have the right event collection policies implemented on workstations, so that events that are irrelevant from the security perspective would not needlessly waste the resources of log collection and security monitoring systems.

4 Summary

Collecting and analysing of Windows event logs is often neglected. It may be caused by the huge amount of events, that Windows can generate on a daily basis or because users do not know, what to do with the collected logs. There are several tools available that allow the user to collect and analyse Windows events, but more often than not these tools are not free or the required functionality is not available in the free version.

The goal of this thesis was to develop a solution to collect and automatically analyse Windows event logs to find security incidents. The tools and the required functionality must be available for free. These tools should also be lightweight, easy to use and compatible with existing centralised log collection solutions. Another goal is to identify what events should be collected from Windows system.

We discovered over 500 unique events from 22 different event sources that should be collected in a Windows 10 environment. We did this by combining recommendations from 4 different sources. These sources included recommendations from Microsoft, the NSA and a list of different techniques used by attackers collected by Malware Archology and MITRE ATT&CK. More events were discovered when analysing 7 different malwares. Malware selection was based on their popularity in this year. When analysing malware we looked at what kind of events they trigger in a Windows environment. We also identified what Windows auditing setting are the most useful to discover security incidents.

In order to identify the best tool to use for Windows event collection we looked at 9 different tools that claim to have that functionality. From those 9 look we take a closer look at 3 that match all our requirements. We look at the functionality, ease of use and performance of these tools. From these tools we identified NXLog to be the best for forwarding Windows events logs. We also develop a configuration file to forward all the events we have discovered that should be forwarded.

We used event correlation to automatically detect security incidents from the collected events. There are several tools available that allow you to use event correlation. We chose

SEC, because it is free, lightweight and easy to use. We created event correlation rules, based on the information we gathered when analysing malware and based on the techniques used by attackers. In total, we created 36 event correlation rules for SEC. These rules were able to successfully detect the malware we analysed and the sequences of events that attacks generate.

We tested the solution in a production environment by configuring 10 Windows 10 machines to send their events to a centralised log collection system. The test result show that the proposed solution uses modest amount of resources and can be scaled to process information for much larger amount of workstations.

In conclusion, we were able to identify, what Windows events should be collected, and developed a solution to collect and analyse Windows events to discover security incidents using free tools.

References

[1] StatsCunter, “Desktop Operating System Market Share Worldwide Jan 2009 - oct 2020,” [Online]. Available: https://gs.statcounter.com/os-market-share/desktop/worldwide/#monthly-200901-202010. [Accessed 24 11 2020].

[2] Statscounter, [Online]. Available: https://gs.statcounter.com/windows-version-market-share/desktop/worldwide/#monthly-200901-202011. [Accessed 24 11 2020].

[3] OWASP, [Online]. Available: https://owasp.org/www-project-top-ten/#. [Accessed 24 11 2020].

[4] R. Anthony, “Detecting Security Incidents Using Windows Workstation Event Logs,” SANS Institute, 2013.

[5] Microsoft, “Description of security events in Windows 7 and in Windows Server 2008 R2,” Microsoft, [Online]. Available: https://support.microsoft.com/en-us/help/977519/description-of-security-events-in-windows-7-and-in-windows-server-2008. [Accessed 4 10 2020].

[6] Microsoft, “Events to Monitor,” Microsoft, 30 7 2018. [Online]. Available: https://docs.microsoft.com/en-gb/windows-server/identity/ad-ds/plan/appendix-l--events-to-monitor?redirectedfrom=MSDN. [Accessed 04 10 2020].

[7] National Security Agancy, “Spotting the Adversary with Windows Event Log Monitoring,” National Security Agancy, 2013.

[8] MalwareArchaeology, “WINDOWS ATT&CK LOGGING CHEAT SHEET,” MalwareArchaeology, 2018.

[9] L. P. T. G. S. P. L. K. T. M. María del Carmen Prudente Tixteco, “Intrusion Detection Using Indicators of Compromise Based on Best Practices and Windows Event Logs,” in *The Eleventh International Conference on Internet Monitoring and Protection*, 2016.

[10] T. A. Arpan Man Sainju, “An Experimental Analysis of Windows Log Events Triggered by Malware,” in *SouthEast Conference*, Kennesaw, 2017.

[11] M. D. Mullinix, *An Analysis of Microsoft Event logs,* Utica: Utica College, 2013.

[12] J. Baráth, “Optimizing Windows 10 logging to detect network security threats,” in *2017 Communication and Information Technologies (KIT)*, 2017.

[13] B. B. M. K. Risto Vaarandi, “An Unsupervised Framework for Detecting Anomalous Messages from Syslog Log Files,” in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, 2018.

[14] S. Petai, *Detecting Anomalies in System Logs,* Tallinn: Tallinn University of Technology, 2014.

[15] M. Gerges, *Log monitoring and event correlation on Microsoft® Windows™ using Simple Event Correlator,* Tallinn: Tallinn University of Technology, 2016.

[16] NXLog Ltd., NXLog User Guide, 2020.

[17] Elasticsearch , “Winlogbeat Reference,” Elasticsearch , [Online]. Available: https://www.elastic.co/guide/en/beats/winlogbeat/current/index.html. [Accessed 20 10 2020].

[18] Elasticsearch, “What are Beats?,” Elasticsearch, [Online]. Available: https://www.elastic.co/guide/en/beats/libbeat/7.9/beats-reference.html. [Accessed 20 10 2020].

[19] Rsyslog, “Rsyslog,” [Online]. Available: https://www.rsyslog.com/doc/v8-stable/. [Accessed 12 10 2020].

[20] Snare Solutions, [Online]. Available: https://www.snaresolutions.com/portfolio-item/open-source-vs-enterprise/. [Accessed 23 10 2020].

[21] One Identity, syslog-ng Premium Edition 7.0.21 Administration Guide, California, 2020.

[22] One identity, syslog-ng Open Source Edition 3.26 Administration Guide, California, 2020.

[23] One identity, syslog-ng Premium Edition 7.0.21 Windows Event Collector Administration Guide, California, 2020.

[24] Graylog, “Graylog,” [Online]. Available: https://docs.graylog.org/en/3.3/. [Accessed 20 10 2020].

[25] “Eventlog-to-syslog,” [Online]. Available: https://code.google.com/archive/p/eventlog-to-syslog/. [Accessed 12 10 2020].

[26] Purdue University, “Eventlog to syslog utility,” [Online]. Available: https://engineering.purdue.edu/ECN/Resources/Documents/UNIX/evtsys. [Accessed 12 10 2020].

[27] Microsoft, “Use Windows Event Forwarding to help with intrusion detection,” [Online]. Available: https://docs.microsoft.com/en-us/windows/security/threat-protection/use-windows-event-forwarding-to-assist-in-intrusion-detection. [Accessed 20 10 2020].

[28] A. H. Michael Sikorski, Practical Malware Analysis, San Fransico: No starch press, 2012.

[29] T. Robinson, Building Virtual Machine Labs.

[30] NirSoft, “FullEventLogView v1.57 - Event Log Viewer for Windows 10 / 8 / 7 / Vista,” NirSoft, [Online]. Available: http://www.nirsoft.net/utils/full_event_log_view.html. [Accessed 5 11 2020].

[31] M. W. Gabriel Jakobson, “Real-time telecommunication network management: extending event correlation with temporal constraints.,” Springer, Boston, 1995.

[32] R. Vaarandi, “Simple Event Correlator for real-time security log monitoring,” *Hakin9*, pp. 28-39, 2006.

[33] R. Vaarandi, “SEC – a Lightweight Event Correlation Tool,” in *IEEE Workshop on IP Operations and Management*, 2002.

[34] EsperTEch, “Esper,” [Online]. Available: https://www.espertech.com/esper/. [Accessed 23 11 2020].

[35] Elastic, [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/current/eql.html. [Accessed 23 11 2020].

[36] D. Harris, “How Splunk Is Riding IT Search Toward an IPO,” Gigaom, 17 12 2010. [Online]. Available: https://gigaom.com/2010/12/17/how-splunk-is-riding-it-search-toward-an-ipo/. [Accessed 23 11 2020].

[37] Splunk, “Event Correlation”.

[38] Splunk, Splunk Enterprice Admin Manual 8.1.0, 2020.

[39] M. S. Karen Kent, Guide to Computer Security Log Managment, Gaithesburg: National Intitute of Standards and Technology, 2006.

[40] R. A. Grimes, “Why you need centralized logging and event log management,” CSO, 12 6 2018. [Online]. Available: https://www.csoonline.com/article/3280123/why-you-need-centralized-logging-and-event-log-management.html. [Accessed 28 10 2020].

[41] B. J. Hemanth Tarra, “Understanding Processor (% Processor Time) and Process (%Processor Time),” Microsoft, 2012 08 13. [Online]. Available: https://social.technet.microsoft.com/wiki/contents/articles/12984.understanding-processor-processor-time-and-process-processor-time.aspx. [Accessed 20 11 2020].

[42] Adrem software, “Correct Monitoring of Windows Processes on multi-core machines,” [Online]. Available: https://www.adremsoft.com/blog/view/blog/6703603919139/correct-monitoring-of-windows-processes-on-multicore-machines. [Accessed 19 12 2020].

[43] Microsoft, “Get a Windows 10 development environment,” [Online]. Available: https://developer.microsoft.com/en-us/windows/downloads/virtual-machines/. [Accessed 30 10 2020].

[44] CheckPoint, [Online]. Available: https://www.checkpoint.com/. [Accessed 6 11 2020].

[45] Checkboint, “January 2020’s Most Wanted Malware,” 13 02 2020. [Online]. Available: https://blog.checkpoint.com/2020/02/13/january-2020s-most-wanted-malware-coronavirus-themed-spam-spreads-malicious-emotet-malware/. [Accessed 20 11 2020].

[46] Checkpoint, “February 2020’s Most Wanted Malware,” 11 03 2020. [Online]. Available: https://blog.checkpoint.com/2020/03/11/february-2020s-most-wanted-malware-increase-in-exploits-spreading-the-mirai-botnet-to-iot-devices/. [Accessed 20 11 2020].

[47] Checkpoint, “March 2020’s Most Wanted Malware,” 9 4 2020. [Online]. Available: https://blog.checkpoint.com/2020/04/09/march-2020s-most-wanted-malware-dridex-banking-trojan-ranks-on-top-malware-list-for-first-time/. [Accessed 20 11 2020].

[48] Checkpoint, “April 2020’s Most Wanted Malware,” 11 5 2020. [Online]. Available: https://blog.checkpoint.com/2020/05/11/april-2020s-most-wanted-malware-agent-tesla-remote-access-trojan-spreading-widely-in-covid-19-related-spam-campaigns/. [Accessed 20 11 2020].

[49] Checkpoint, “May’s Most Wanted Malware,” 15 6 2020. [Online]. Available: https://blog.checkpoint.com/2020/06/15/mays-most-wanted-malware-ursnif-banking-trojan-ranks-on-top-10-malware-list-for-first-time-over-doubling-its-impact-on-organizations/. [Accessed 20 11 2020].

[50] Checkpoint, “June‘s Most Wanted Malware,” 10 7 2020. [Online]. Available: https://blog.checkpoint.com/2020/07/10/junes-most-wanted-malware-notorious-

phorpiex-botnet-rises-again-doubling-its-global-impact-on-organizations/. [Accessed 20 11 2020].

[51] Checkpoint, “July‘s Most Wanted Malware,” 7 8 2020. [Online]. Available: https://blog.checkpoint.com/2020/08/07/julys-most-wanted-malware-emotet-strikes-again-after-five-month-absence/. [Accessed 20 11 2020].

[52] Checkpoint, “August 2020’s Most Wanted Malware,” 9 9 2020. [Online]. Available: https://blog.checkpoint.com/2020/09/09/august-2020s-most-wanted-malware-evolved-qbot-trojan-ranks-on-top-malware-list-for-first-time/. [Accessed 20 11 2020].

[53] CheckPoint, “September 2020's Most Wanted Malware,” 7 10 2020. [Online]. Available: https://blog.checkpoint.com/2020/10/07/september-2020s-most-wanted-malware-new-info-stealing-valak-variant-enters-top-10-malware-list-for-first-time/. [Accessed 6 11 2020].

[54] CheckPoint, “October 2020's Most Wanted Malware,” 6 11 2020. [Online]. Available: https://blog.checkpoint.com/2020/11/06/october-2020s-most-wanted-malware-trickbot-and-emotet-trojans-are-driving-spike-in-ransomware-attacks/. [Accessed 6 11 2020].

[55] Abuse, “MalwareBazaar,” [Online]. Available: https://bazaar.abuse.ch. [Accessed 06 11 2020].

[56] C. Moore, “Malware Analysis: What is Agent Tesla and How Can You Protect Your Enterprise From It?,” Reliaquest, 30 7 2020. [Online]. Available: https://www.reliaquest.com/blog/malware-analysis-what-is-agent-tesla-and-how-can-you-protect-your-enterprise-from-it/. [Accessed 7 11 2020].

[57] Any Run, “Dridex,” [Online]. Available: https://any.run/malware-trends/dridex. [Accessed 7 11 2020].

[58] Any Run, “Tricbot,” [Online]. Available: https://any.run/malware-trends/trickbot. [Accessed 7 10 2020].

[59] L. R. A. D. Eli Salem, “Valak: More than Meets the Eye,” Cybereason, 28 5 2020. [Online]. Available: https://www.cybereason.com/blog/valak-more-than-meets-the-eye. [Accessed 10 11 2020].

[60] Any Run, “Formbook,” [Online]. Available: https://any.run/malware-trends/formbook. [Accessed 02 12 2020].

[61] A. Kuzmenko, “On the trail of the XMRig miner,” Kaspersky, [Online]. Available: https://securelist.com/miner-xmrig/99151/. [Accessed 2 12 2020].

[62] Any Run, “Glupteba,” [Online]. Available: https://any.run/malware-trends/glupteba. [Accessed 2 12 2020].

[63] Microsoft, “Audit Policy Recommendations,” 31 5 2017. [Online]. Available: https://docs.microsoft.com/en-gb/windows-server/identity/ad-ds/plan/security-best-practices/audit-policy-recommendations. [Accessed 9 11 2020].

[64] Malware Archaeology, “Logging,” [Online]. Available: https://www.malwarearchaeology.com/logging/. [Accessed 12 11 2020].

[65] MITRE ATT&CK, “ATT&CK Matrix for Enterprise,” [Online]. Available: https://attack.mitre.org/. [Accessed 6 12 2020].

[66] IBM, WinCollect User Guide V7.2.3, 2016.

[67] Any Run, “Agent Tesla,” [Online]. Available: https://any.run/malware-trends/agenttesla. [Accessed 7 11 2020].

[68] Any Run, “Emotet,” [Online]. Available: https://any.run/malware-trends/emotet. [Accessed 7 11 2020].

[69] R. Antony, *Detecting Security Incidents,* SANS Institute, 2013.

Appendix 1 – NXLog configuration

```
Panic Soft
#NoFreeOnExit TRUE


define ROOT     C:\Program Files (x86)\nxlog
define CERTDIR  %ROOT%\cert
define CONFDIR  %ROOT%\conf
define LOGDIR   %ROOT%\data
define LOGFILE  %LOGDIR%\nxlog.log
LogFile %LOGFILE%

Moduledir %ROOT%\modules
CacheDir  %ROOT%\data
Pidfile   %ROOT%\data\nxlog.pid
SpoolDir  %ROOT%\data

#Events to collect
define EventsToCollect     1, 2, 6, 15, 19, 20, 21, 23, 24, 25, 31, 34, 35,
41, \
                           43, 104, 219, 307, 400, 410, 441, 800, 865, 866, \
                           867, 868, 882, 903, 904, 905, 906, 907, 908, 1000,
\
                           1001, 1002, 1005, 1006, 1008, 1009, 1010, 1022, \
                           1033, 1102, 1125, 1127, 1129, 2001, 2003, 2004, \
                           2005, 2006, 2009, 2033, 3001, 3002, 3003, 3004, \
                           3010, 3023, 4100, 4101, 4102, 4103, 4104, 4608, \
                           4609, 4610, 4611, 4612, 4614, 4615, 4616, 4618, \
                           4621, 4622, 4624, 4625, 4634, 4646, 4647, 4648, \
                           4649, 4650, 4651, 4652, 4653, 4654, 4655, 4656, \
                           4657, 4658, 4659, 4660, 4661, 4662, 4663, 4664, \
                           4665, 4666, 4667, 4668, 4670, 4671, 4672, 4673, \
                           4674, 4675, 4688, 4689, 4690, 4691, 4692, 4693, \
                           4694, 4695, 4696, 4697, 4698, 4699, 4700, 4701, \
                           4702, 4704, 4705, 4706, 4707, 4709, 4710, 4711, \
                           4712, 4713, 4714, 4715, 4716, 4717, 4718, 4719, \
                           4720, 4722, 4723, 4724, 4725, 4726, 4727, 4728, \
                           4729, 4730, 4731, 4732, 4733, 4734, 4735, 4737, \
                           4738, 4739, 4740, 4741, 4742, 4743, 4744, 4745, \
                           4746, 4747, 4748, 4749, 4750, 4751, 4752, 4753, \
                           4754, 4755, 4756, 4757, 4758, 4759, 4760, 4761, \
                           4762, 4764, 4765, 4766, 4767, 4768, 4769, 4770, \
                           4771, 4772, 4773, 4774, 4775, 4776, 4777, 4778, \
                           4779, 4780, 4781, 4782, 4783, 4784, 4785, 4786, \
```

```
                            4787, 4788, 4789, 4790, 4791, 4792, 4793, 4794, \
                            4800, 4801, 4802, 4803, 4816, 4817, 4864, 4865, \
                            4866, 4867, 4868, 4869, 4870, 4871, 4872, 4873, \
                            4874, 4875, 4876, 4877, 4878, 4879, 4880, 4881, \
                            4882, 4883, 4884, 4885, 4886, 4887, 4888, 4889, \
                            4890, 4891, 4892, 4893, 4894, 4895, 4896, 4897, \
                            4898, 4899, 4900, 4902, 4904, 4905, 4906, 4907, \
                            4908, 4909, 4910, 4912, 4928, 4929, 4930, 4931, \
                            4932, 4933, 4934, 4935, 4936, 4937, 4944, 4945, \
                            4946, 4947, 4948, 4949, 4950, 4951, 4952, 4953, \
                            4954, 4956, 4957, 4958, 4960, 4961, 4962, 4963, \
                            4964, 4965, 4976, 4977, 4978, 4979, 4980, 4981, \
                            4982, 4983, 4984, 4985, 5008, 5024, 5025, 5027, \
                            5028, 5029, 5030, 5031, 5032, 5033, 5034, 5035, \
                            5037, 5038, 5039, 5040, 5041, 5042, 5043, 5044, \
                            5045, 5046, 5047, 5048, 5049, 5050, 5051, 5056, \
                            5057, 5058, 5059, 5060, 5061, 5062, 5063, 5064, \
                            5065, 5066, 5067, 5068, 5069, 5070, 5120, 5121, \
                            5122, 5123, 5124, 5125, 5126, 5127, 5136, 5137, \
                            5138, 5139, 5140, 5141, 5142, 5143, 5144, 5145, \
                            5148, 5149, 5150, 5151, 5152, 5153, 5154, 5155, \
                            5156, 5157, 5158, 5159, 5168, 5376, 5377, 5378, \
                            5440, 5441, 5442, 5443, 5444, 5446, 5447, 5448, \
                            5449, 5450, 5451, 5452, 5453, 5456, 5457, 5458, \
                            5459, 5460, 5461, 5462, 5463, 5464, 5465, 5466, \
                            5467, 5468, 5471, 5472, 5473, 5474, 5477, 5478, \
                            5479, 5480, 5483, 5484, 5485, 5632, 5633, 5712, \
                            5861, 5888, 5889, 5890, 6008, 6144, 6145, 6272, \
                            6273, 6274, 6275, 6276, 6277, 6278, 6279, 6280, \
                            6281, 6400, 6401, 6403, 6404, 6405, 6406, 6407, \
                            7022, 7023, 7024, 7026, 7031, 7032, 7034, 7040, \
                            7045, 8000, 8001, 8002, 8003, 8004, 8006, 8007, \
                            8011, 10000, 10001, 11000, 11001, 11002, 11004, \
                            11005, 11006, 11010, 12011, 12012, 12013, 24577, \
                            24578, 24579, 24580, 24581, 24582, 24583, 24584, \
                            24586, 24588, 24592, 24593, 24594, 24595, 24621
#Malware related events
define MalwereEvents       100, 104, 129, 140, 200, 201, 1116

<Extension syslog>
    Module          xm_syslog
</Extension>

<Input eventlog>
    Module          im_msvistalog
    <QueryXML>
        <QueryList>
            <Query Id='0'>
                <Select Path='Application'>*</Select>
                <Select Path='Security'>*</Select>
                <Select Path='Setup'>*</Select>
```

```
                <Select Path='System'>*</Select>
                <Select Path='Microsoft-Windows-Application-
Experience/Program-Inventory'>*</Select>
                <Select Path='Microsoft-Windows-AppLocker/EXE and
DLL'>*</Select>
                <Select Path='Microsoft-Windows-AppLocker/MSI and
Script'>*</Select>
                <Select Path='Microsoft-Windows-
CodeIntegrity/Operational'>*</Select>
                <Select Path='Microsoft-Windows-Kernel-
PnP/Configuration'>*</Select>
                <Select Path='Microsoft-Windows-
NetworkProfile/Operational'>*</Select>
                <Select Path='Microsoft-Windows-
PrintService/Operational'>*</Select>
                <Select Path='Microsoft-Windows-PowerShell/Admin'>*</Select>
                <Select Path='Microsoft-Windows-
PowerShell/Operational'>*</Select>
                <Select Path='Microsoft-Windows-
TaskScheduler/Operational'>*</Select>
                <Select Path='Microsoft-Windows-TerminalServices-
LocalSessionManager/Operational'>*</Select>
                <Select Path='Microsoft-Windows-TerminalServices-
RemoteConnectionManager/Operational'>*</Select>
                <Select Path='Microsoft-Windows-Windows
Defender/Operational'>*</Select>
                <Select Path='Microsoft-Windows-Windows Firewall With
Advanced Security/Firewall'>*</Select>
                <Select Path='Microsoft-Windows-
WindowsUpdateClient/Operational'>*</Select>
                <Select Path='Microsoft-Windows-WLAN-
AutoConfig/Operational'>*</Select>
            </Query>
        </QueryList>
    </QueryXML>
    exec    if ($EventID NOT IN (%EventsToCollect%)) and \
                ($EventID NOT IN (%MalwereEvents%)) drop();
    exec $Message = 'EventID:[' + $EventID + '] ' + $Message ;
</Input>


#100 - 200 TaskScheduler
<Input eventlogSC>
    Module          im_msvistalog
    <QueryXML>
        <QueryList>
            <Query Id='0'>
                <Select Path='Microsoft-Windows-
TaskScheduler/Operational'>*</Select>
            </Query>
        </QueryList>
    </QueryXML>
    exec    if ($EventID < 100) or \
```

```
                ($EventID > 200) drop();
    exec $Message = 'EventID:[' + $EventID + '] ' + $Message ;
</Input>

#200 - 500 PowerShell
<Input eventlogPWR>
    Module          im_msvistalog
    <QueryXML>
        <QueryList>
            <Query Id='0'>
                <Select Path='Microsoft-Windows-PowerShell/Admin'>*</Select>
                <Select Path='Microsoft-Windows-
PowerShell/Operational'>*</Select>
            </Query>
        </QueryList>
    </QueryXML>
    exec    if ($EventID < 200) OR \
                ($EventID > 500) drop();
    exec $Message = 'EventID:[' + $EventID + '] ' + $Message ;
</Input>

00
<Output udp>
    Module          om_udp
    Host            192.168.56.105
    Port            514
    Exec            to_syslog_bsd();
</Output>

<Route eventlog_to_udp>
    Path            eventlog, eventlogSC, eventlogPWR => udp
</Route>

<Extension _charconv>
    Module      xm_charconv
    AutodetectCharsets iso8859-2, utf-8, utf-16, utf-32
</Extension>

<Extension _exec>
    Module      xm_exec
</Extension>

<Extension _fileop>
    Module      xm_fileop

    # Check the size of our log file hourly, rotate if larger than 5MB
    <Schedule>
        Every   1 hour
        Exec    if (file_exists('%LOGFILE%') and \
                   (file_size('%LOGFILE%') >= 5M)) \
                    file_cycle('%LOGFILE%', 8);
```

```
    </Schedule>

    # Rotate our log file every week on Sunday at midnight
    <Schedule>
        When    @weekly
        Exec    if file_exists('%LOGFILE%') file_cycle('%LOGFILE%', 8);
    </Schedule>
</Extension>
```

Appendix 2 – SEC rules 1

```
#Defender Disabled
type=SingleWithSuppress
ptype=RegExp
pattern=^[\d\:\-\+T]+ (\w+) SecurityCenter\[\d+\]: EventID:\[\d+\] .+
SECURITY_PRODUCT_STATE_SNOOZED\.
desc=Windows Defender disabled at $1
action=pipe '%s' /bin/mailx -r noreply@someplace.com -s 'Defender disabled'
root@localhost; write /opt/sec/logs/detected.log %t %s
window=60

#Defender discovered malware
type=Single
ptype=RegExp
pattern=^[\d\:\-\+T]+ (\w+) Microsoft-Windows-Windows_Defender\[\d+\]:
EventID:\[\d+\] .+ has detected malware .+ #011Name: (.+) #011ID: .+
#011Path: (.+) #011Detection Origin: .+ #011Process Name: (.+) #011Security
desc=Windows Defender has discovered malware at $1. Malware name:$2 Path:$3
Process:$4
action=pipe '%s' /bin/mailx -r noreply@someplace.com -s 'Malware discovered'
root@localhost; write /opt/sec/logs/detected.log %t %s

#Event Log file was cleared
#type=Single
#ptype=RegExp
#pattern=^[\d\:\-\+T]+ (\w+) Microsoft-Windows-Eventlog\[\d+\]:
EventID:\[\d+\] The ([\w-\/\s]+) log
#desc=The $2 log has been cleared from $1
#action=pipe '%s' /bin/mailx -r noreply@someplace.com -s 'Event log cleared'
root@localhost; write /opt/sec/logs/detected.log %t %s

#Event Log file was cleared
#all events that are generated in 60 seconds are sent to email
type=Single
ptype=RegExp
pattern=^[\d\:\-\+T]+ (\w+) Microsoft-Windows-Eventlog\[\d+\]:
EventID:\[\d+\] The ([\w\-\/\s]+) log
desc=The $2 log has been cleared from $1
action=exists %iscreated EVENT_CLEARED_$1;if %iscreated (add EVENT_CLEARED_$1
%s;) else (create EVENT_CLEARED_$1 60 (report EVENT_CLEARED_$1 /bin/mailx -r
noreply@someplace.com -s 'Events Cleared' root@localhost); add
EVENT_CLEARED_$1 %s;)

#Potentially malicious command
type=Single
ptype=RegExp
```

```
pattern=^[\d\:\-\+T]+ (\w+) Microsoft-Windows-Security-Auditing\[\d+\]:
EventID:\[\d+\] A new process has been created.+New Process
ID:#011#011([\w\d]+)  #011New Process
Name:#011([\w:\\\.]+(cmd|powershell)\.exe) .+#011Creator Process
ID:#011([\w\d]+)  #011Creator Process Name:#011([\w:\\\.]+).+ -w hidden -
ENCOD
desc=Discovered a potentially malicious $3 process ID $2 started by $6
process ID$5 on $1
action=pipe '%s' /bin/mailx -r noreply@someplace.com -s 'Potentially
malicious command run' root@localhost; write /opt/sec/logs/detected.log %t %s

#Scriptblock created
type=Single
ptype=RegExp
pattern=^[\d\:\-\+T]+ (\w+) Microsoft-Windows-PowerShell\[\d+\]:
EventID:\[\d+\] Creating Scriptblock text
desc=Scriptblock created on $1
action=write /opt/sec/logs/detected.log %t %s

#Agent tesla
#something starts schtasks.exe and creats schtask /TN "Updates\<some name>"
(/TN - taskname)
#that same schtasks.exe starts conhost.exe
#schtask with "Updates\<some name>" created
type=Single
ptype=RegExp
pattern=^[\d\:\-\+T]+ (\w+) Microsoft-Windows-Security-Auditing\[\d+\]:
EventID:\[\d+\] A new process has been created.+New Process
ID:#011#011([\w\d]+)  #011New Process Name:#011[\w:\\\.]+schtasks\.exe.+\/TN
"([\w\d\\]+)"
context=!AT1_$1_$2 && !AT1_$1_$3
desc=AT1_$1_$2 and AT1_$1_$3
action=create AT1_$1_$2 60;create AT1_$1_$3 60;write
/opt/sec/logs/detected.log %t %s

type=Single
ptype=RegExp
pattern=^[\d\:\-\+T]+ (\w+) Microsoft-Windows-Security-Auditing\[\d+\]:
EventID:\[\d+\] A new process has been created.+#011New Process
Name:#011[\w:\\\.]+conhost\.exe.+Creator Process ID:#011([\w\d]+)
#011Creator Process Name:#011[\w:\\\.]+schtasks\.exe
context=AT1_$1_$2 && !AT2_$1
desc=AT2_$1
action=create AT2_$1 60;write /opt/sec/logs/detected.log %t %s

type=Single
ptype=RegExp
pattern=^[\d\:\-\+T]+ (\w+) Microsoft-Windows-TaskScheduler\[\d+\]:
EventID:\[\d+\] .+registered Task Scheduler task "\\([\w\d\\]+)"
context=AT1_$1_$2 && AT2_$1
desc=Potential Agent Tesla on $1 with task name $2
action=pipe '%s' /bin/mailx -r noreply@someplace.com -s 'Potential Agent
Tesla' root@localhost;write /opt/sec/logs/detected.log %t %s
```

```
#Trickbot
#svchost starts dllhost with process command line DllHost.exe
type=Single
ptype=RegExp
pattern=^[\d\:\-\+T]+ (\w+) Microsoft-Windows-Security-Auditing\[\d+\]:
EventID:\[\d+\] A new process has been created.+New Process
ID:#011#011([\w\d]+)  #011New Process
Name:#011[\w:\\\.]+dllhost\.exe.+svchost\.exe.+DllHost\.exe
context=!TB1_$1_$2
desc=TB1_$1_$2
action=create TB1_$1_$2 60;write /opt/sec/logs/detected.log %t %s

#dllhost start trickbot
type=Single
ptype=RegExp
pattern=^[\d\:\-\+T]+ (\w+) Microsoft-Windows-Security-Auditing\[\d+\]:
EventID:\[\d+\] A new process has been created.+New Process
ID:#011#011([\w\d]+)  #011New Process Name:#011([\w:\\\.]+).+Creator Process
ID:#011([\w\d]+)  #011Creator Process Name:#011[\w:\\\.]+dllhost\.exe
context=TB1_$1_$4 && !TB2_$1_$2_$3
desc=TB2_$1_$2_$3
action=create TB2_$1_$2_$3 60;write /opt/sec/logs/detected.log %t %s

#Trickbot starts wermgr.exe
type=Single
ptype=RegExp
pattern=^[\d\:\-\+T]+ (\w+) Microsoft-Windows-Security-Auditing\[\d+\]:
EventID:\[\d+\] A new process has been created.+#011New Process
Name:#011[\w:\\\.]+wermgr\.exe.+Creator Process ID:#011([\w\d]+)  #011Creator
Process Name:#011([\w:\\\.]+)
context=TB2_$1_$2_$3 && !TB3_$1
desc=TB3_$1
action=create TB3_$1 600;pipe 'TB2_$1_$2_$3' /bin/mailx -r
noreply@someplace.com -s 'Potential Trickbot stage 3' root@localhost;write
/opt/sec/logs/detected.log %t %s

#Windwos Power Saves named task started
type=Single
ptype=RegExp
pattern=^[\d\:\-\+T]+ (\w+) Microsoft-Windows-TaskScheduler\[\d+\]:
EventID:\[\d+\].+Windows Power Saves
context=TB3_$1 && !TB4_$1
desc=TB4_$1
action=create TB4_$1 60;write /opt/sec/logs/detected.log %t %s

#Windows Power Saves starts trickbot
type=Single
ptype=RegExp
pattern=^[\d\:\-\+T]+ (\w+) Microsoft-Windows-TaskScheduler\[\d+\]:
EventID:\[\d+\].+action "([\w:\\\.\ ]+).+Windows Power Saves
context=TB4_$1
```

```
desc=Potential Trickbot at $1 task $2
action=pipe '%s' /bin/mailx -r noreply@someplace.com -s 'Potential Trickbot
stage 5' root@localhost;write /opt/sec/logs/detected.log %t %s

#Valak starts wscript.exe
type=Single
ptype=RegExp
pattern=^[\d\:\-\+T]+ (\w+) Microsoft-Windows-Security-Auditing\[\d+\]:
EventID:\[\d+\] A new process has been created.+#011New Process
Name:#011[\w:\\\.]+wscript\.exe.+#011Creator Process
Name:#011[\w:\\\.]+regsvr32\.exe
desc=Potential Valak at $1
action=pipe '%s' /bin/mailx -r noreply@someplace.com -s 'Potential Valak'
root@localhost;write /opt/sec/logs/detected.log %t %s

#XMRig
#XMRig starts itself
#the new XMRig starts notepad with command line -c <file location>
#xmrig starts cmd with ommand line wscript <script location>
type=Single
ptype=RegExp
pattern=[\d\:\-\+T]+ (\w+) Microsoft-Windows-Security-Auditing\[\d+\]:
EventID:\[\d+\] A new process has been create.+#011New Process
ID:#011#011([\w\d]+)  #011New Process Name:#011(.+)  #011Token.+#011Creator
Process Name:#011(.+)  #
context=!XMRig_$1_$2 && =("$3" eq "$4")
desc=XMRig_$1_$2 $3
action=create XMRig_$1_$2 60

type=Single
ptype=RegExp
pattern=[\d\:\-\+T]+ (\w+) Microsoft-Windows-Security-Auditing\[\d+\]:
EventID:\[\d+\] A new process has been create.+#011New Process
Name:#011[\w:\\\.]+notepad\.exe.+#011Creator Process
ID:#011([\w\d]+).+Command Line:#011"(.+)"
context=XMRig_$1_$2 && !XMRig2_$1_$2
desc=XMRig2_$1 $3
action=create XMRig2_$1_$2 60;write /opt/sec/logs/detected.log %t %s


type=Single
ptype=RegExp
pattern=[\d\:\-\+T]+ (\w+) Microsoft-Windows-Security-Auditing\[\d+\]:
EventID:\[\d+\] A new process has been create.+#011New Process
Name:#011[\w:\\\.]+cmd\.exe.+#011Creator Process ID:#011([\w\d]+).+Command
Line:#011(.+)"
context=XMRig2_$1_$2 && !XMRig3_$1
desc=Potential XMRig at $1 $3
action=pipe '%s' /bin/mailx -r noreply@someplace.com -s 'Potential XMRig'
root@localhost;create XMRig3_$1 60;write /opt/sec/logs/detected.log %t %s

#Glupteba
type=Single
```

```
ptype=RegExp
pattern=[\d\:\-\+T]+ (\w+) Microsoft-Windows-Security-Auditing\[\d+\]:
EventID:\[\d+\] A new process has been create.+#011New Process
ID:#011#011([\w\d]+).+\/C fodhelper
context=!Glupteba_$1_$2
desc=Glupteba_$1_$2
action=create Glupteba_$1_$2 60;write /opt/sec/logs/detected.log %t %s

type=Single
ptype=RegExp
pattern=[\d\:\-\+T]+ (\w+) Microsoft-Windows-Security-Auditing\[\d+\]:
EventID:\[\d+\] A new process has been create.+#011New Process
ID:#011#011([\w\d]+)  #011New Process
Name:#011[\w:\\\.]+fodhelper\.exe.+#011Creator Process ID:#011([\w\d]+)
context=Glupteba_$1_$3 && !Glupteba2_$1_$2
desc=Glupteba2_$1_$2
action=create Glupteba2_$1_$2 60;write /opt/sec/logs/detected.log %t %s

type=Single
ptype=RegExp
pattern=[\d\:\-\+T]+ (\w+) Microsoft-Windows-Security-Auditing\[\d+\]:
EventID:\[\d+\] A new process has been create.+#011New Process
ID:#011#011([\w\d]+)  #011New Process Name:#011([\w:\\\.]+).+#011Creator
Process ID:#011([\w\d]+)  #011Creator Process
Name:#011[\w:\\\.]+fodhelper\.exe
context=Glupteba2_$1_$4 && !Glupteba3_$1_$2_$3
desc=Glupteba3_$1_$2_$3
action=create Glupteba3_$1_$2_$3 60;write /opt/sec/logs/detected.log %t %s

type=Single
ptype=RegExp
pattern=[\d\:\-\+T]+ (\w+) Microsoft-Windows-Security-Auditing\[\d+\]:
EventID:\[\d+\] A new process has been create.+#011New Process
Name:#011[\w:\\\.]+WerFault\.exe.+#011Creator Process ID:#011([\w\d]+)
#011Creator Process Name:#011([\w:\\\.]+)
context=Glupteba3_$1_$2_$3
desc=Potential malware Glupteba at $1 file $3
action=pipe '%s' /bin/mailx -r noreply@someplace.com -s 'Potential Glupteba'
root@localhost;write /opt/sec/logs/detected.log %t %s
```

Appendix 3 – SEC rules 2

```
#4688 execution then cmd
type=pair
ptype=RegExp
pattern=[\d\:\-\+T]+ (\w+) .+EventID:\[4688\].+#011New Process
ID:#011#011([\w\d]+).+#011Creator Process ID:#011([\w\d]+)  #011Creator
Process Name:#011[\w:\\\.]+(?<!cmd).exe
context=!cmd_$1_$3
desc=exec_$1_$2
action=create exec_$1_$2 5
ptype2=RegExp
pattern2=[\d\:\-\+T]+ $1 .+EventID:\[4688\].+#011Creator Process ID:#011$2
#011Creator Process Name:#011[\w:\\\.]+cmd
context2=!exec_cmd_%1
desc2=exec_cmd_%1
action2=create exec_cmd_%1 10;write /opt/sec/logs/detected.log %t %s
window=60

#4688 cmd then execution
type=pair
ptype=RegExp
pattern=[\d\:\-\+T]+ (\w+) .+EventID:\[4688\].+#011New Process
ID:#011#011([\w\d]+).+#011Creator Process ID:#011([\w\d]+)  #011Creator
Process Name:#011[\w:\\\.]+cmd
context=!exec_$1_$3
desc=cmd_$1_$2
action=create cmd_$1_$2 5
ptype2=RegExp
pattern2=[\d\:\-\+T]+ $1 .+EventID:\[4688\].+#011Creator Process ID:#011$2
#011Creator Process Name:#011[\w:\\\.]+(?<!cmd).exe
context2=!cmd_exec_%1
desc2=cmd_exec_%1
action2=create cmd_exec_%1 10;write /opt/sec/logs/detected.log %t %s
window=60

#this is currently commented out for testing purposes. Since we generate
events with powershell
#collection - data from local system
#exec_cmd - powershell 100-500, 4100-4104 - 4663 - 5861
type=Single
ptype=RegExp
pattern=[\d\:\-\+T]+ (\w+).+PowerShell.+EventID:\[(\d+)\]
context=exec_cmd_$1 && !col1_ds1_$1 && (=("$2" >= 100 && "$2" <= 500) ||
=("$2" >= 4100 && "$2" <= 4104))
desc=col1_ds1_$1
action=create col1_ds1_$1 5;write /opt/sec/logs/detected.log %t %s

type=Single
ptype=RegExp
pattern=[\d\:\-\+T]+ (\w+).+EventID:\[4663\]
```

```
context=col1_ds1_$1 && !col1_ds2_$1
desc=col1_ds2_$1
action=create col1_ds2_$1 5;write /opt/sec/logs/detected.log %t %s

type=Single
ptype=RegExp
pattern=[\d\:\-\+T]+ (\w+).+EventID:\[5861\]
context=col1_ds2_$1 && !col1_ds3_$1
desc=Potential Collection - data from local system or Potential Discovery -
system net conf at $1. Look for exec_cmd - powershell 100-500, 4100-4104 -
4663 - 586
action=pipe '%s' /bin/mailx -r noreply@someplace.com -s 'Potential Collection
or Discovery' root@localhost;create col1_ds3_$1 5;write
/opt/sec/logs/detected.log %t %s


#collection - data from network shared drive
#cmd_exec - 5140/5145 - 4663
type=Single
ptype=RegExp
pattern=[\d\:\-\+T]+ (\w+).+EventID:\[(\d+)\]
context=cmd_exec_$1 && !col2_ds1_$1 && =("$2" == 5140 || "$2" == 5145) &&
!col3_ds1_$1
desc=col2_ds1_$1
action=create col2_ds1_$1 5;write /opt/sec/logs/detected.log %t %s

type=Single
ptype=RegExp
pattern=[\d\:\-\+T]+ (\w+).+EventID:\[4663\]
context=col2_ds1_$1 && !col2_ds2_$1
desc=Potential Collection - data from network shared drive at $1. Look for
cmd_exec - 5140/5145 - 4663
action=pipe '%s' /bin/mailx -r noreply@someplace.com -s 'Potential
Collection' root@localhost;create col2_ds2_$1 5;write
/opt/sec/logs/detected.log %t %s


#defence evasion - net share connection removal
#cmd_exec - 5140/5145 - 4624
type=Single
ptype=RegExp
pattern=[\d\:\-\+T]+ (\w+).+EventID:\[4624\]
context=col2_ds1_$1 && !def1_ds2_$1
desc=Potential defence evasion - network share connection removal. Look for
cmd_exec - 5140/5145 - 4624
action=pipe '%s' /bin/mailx -r noreply@someplace.com -s 'Potential Defence
evasion' root@localhost;create def1_ds2_$1 5;write /opt/sec/logs/detected.log
%t %s


#collection - data from removeble media
#cmd_exec - 4657 - 4663 - 5140/5145
```

```
type=Single
ptype=RegExp
pattern=[\d\:\-\+T]+ (\w+).+EventID:\[4657\]
context=exec_cmd_$1 && !col3_ds1_$1
desc=col3_ds1_$1
action=create col3_ds1_$1 5;write /opt/sec/logs/detected.log %t %s

type=Single
ptype=RegExp
pattern=[\d\:\-\+T]+ (\w+).+EventID:\[4663\]
context=col3_ds1_$1 && !col3_ds2_$1
desc=col3_ds2_$1
action=create col3_ds2_$1 5;write /opt/sec/logs/detected.log %t %s;pipe
'Potential Execution - powershell at $1. Look for cmd_exec - 4657 - 4663'
/bin/mailx -r noreply@someplace.com -s 'Potential Collection' root@localhost;

type=Single
ptype=RegExp
pattern=[\d\:\-\+T]+ (\w+).+EventID:\[(\d+)\]
context=col3_ds2_$1 && !col3_ds3_$1 && =("$2" == 5140 || "$2" == 5145)
desc=Potential collection - data from removable media. Look for cmd_exec -
4657 - 4663 - 5140/5145
action=pipe '%s' /bin/mailx -r noreply@someplace.com -s 'Potential
Collection' root@localhost;create col3_ds3_$1 5;write
/opt/sec/logs/detected.log %t %s


#Credential access Brute Force
#4625 more then X ammount of times
type=SingleWithThreshold
ptype=RegExp
pattern=[\d\:\-\+T]+ (\w+).+EventID:\[4625\]
desc=Potential Bruteforce at $1. Look fr 4625
action=pipe '%s' /bin/mailx -r noreply@someplace.com -s 'Potential
Bruteforce' root@localhost;write /opt/sec/logs/detected.log %t %s
thresh=5
window=180


#Execution
#service cmd_exec - 4657 - 7054 - 7040
type=Single
ptype=RegExp
pattern=[\d\:\-\+T]+ (\w+).+EventID:\[7054\]
context=col3_ds1_$1 && !exe1_ds2_$1
desc=exe1_ds2_$1
action=create exe1_ds2_$1 5;write /opt/sec/logs/detected.log %t %s

type=Single
ptype=RegExp
pattern=[\d\:\-\+T]+ (\w+).+EventID:\[7040\]
context=exe1_ds2_$1 && !exe1_ds3_$1
```

```
desc=Potential execution - service at $1. Look for cmd_exec - 4657 - 7054 -
7040
action=pipe '%s' /bin/mailx -r noreply@someplace.com -s 'Potential Execution'
root@localhost;create exe1_ds3_$1 5;write /opt/sec/logs/detected.log %t %s
```