Threat-Hunting in Windows Environment Using Host-based Log Data

by

Mohammad Rasool Fatemi

Bachelor of Science in Software Engineering, FUM, 2017

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Computer Science

In the Graduate Academic Unit of Computer Science

THE UNIVERSITY OF NEW BRUNSWICK

December, 2019

Abstract

In a general log-based anomaly detection system, network, devices and host logs are all used together for analysis and the detection of anomalies. However, the ever-increasing volume of logs remains as one of the main challenges that anomaly detection tools face. This thesis proposes a host-based log analysis system that detects anomalies without using network logs to reduce the volume and to show the importance of host-based logs. A parser is implemented to parse and extract features from Sysmon logs and use them to perform detection methods on the data. The valuable information is successfully retained after two extensive volume reduction steps. An anomaly detection system is proposed and performed on different datasets with up to 55,000 events and over a million log messages. The system easily detects the attacks and malicious activities using the preserved logs. The analysis results demonstrate the significance of host-based logs in auditing, security monitoring, and intrusion detection systems.

Dedication

This thesis is dedicated to my beloved parents and fiance for their love and support. Thank you all for believing in me and for helping me to find what I love in my life.

Acknowledgements

I would like to acknowledge my indebtedness and render my warmest thanks to my supervisor, Professor Ali Ghorbani, who made this work possible. His friendly guidance and expert advice have been invaluable throughout all stages of the work.

Table of Contents

List of Tables

List of Figures

Abbreviations

SIEM	Security Information and Event Management
JSON	JavaScript Object Notation
EXE	Executable
ELK	Elasticsearch, Logstash, Kibana
Cmd	Command Prompt
DLL	Dynamic-link library
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VPN	Virtual Private Network
GUDI	Globally Unique Identifier
WMI	Windows Management Instrumentation
DNS	Domain Name System
AWS	Amazon Web Services
URL	Uniform Resource Locator
XML	eXtensible Markup Language
MD5	MD5 message-digest algorithm
SHA-256	Secure Hash Algorithm 256-bit
CSV	Comma-Separated Values
EVTX	Windows XML Event Log
API	Application Programming Interface
GDPR	General Data Protection Regulation
LSASS	Local Security Authority Subsystem Service

Chapter 1

Introduction

1.1 Introduction

One of the results of the continually growing number of devices connected to the Internet is the production of vast amounts of log and other data. This ever-increasing volume of data, especially security logs, should be stored and can be analyzed for different purposes. Whether it be an attack or a malicious activity, with an excellent and in-depth analysis of the right log data, we can find it. Also, event pattern discovery, one of the most critical tasks in security log management and analysis, can be used to build security log file profiles; thus anomalous lines in the log files can be identified. Based on recent studies, despite having notable progress, there are several remaining challenges in log analysis, and there is still a long way to go when it comes to security analysis of big data.

With more than 75% of the share of desktop operating systems [35], Windows is considered the most widely used desktop operating system worldwide, and hence, there is a massive number of logs generated by Windows-operated workstations and laptops every day. Network log analysis has come a long way so far, but system log analysis seems to have been undervalued compared to that. Sysmon could be one possible way to help solve this issue. As a good start, security researchers at Microsoft developed Sysmon to help with Windows the auditing process. This logging tool can help address several of the above issues if properly configured and deployed. There are many security information and event management software (SIEM) applications that can help with the analysis of these logs and other logs from different sources. In this thesis, we aim to show the importance of host-based logs for security analysis. We also list and address related challenges and discuss potential solutions.

This thesis will demonstrate how an enormous number of security logs can be handled and used as the sole source of information for anomaly detection. Here, we propose a set of tools to help detect anomalies in Windows workstations. The goal of the work is to provide a set of simple and easy-to-deploy tools that can be used to track and hunt malicious activities and help with the response after an attack. To achieve this, we focus on host-based logs generated by Sysmon and aim to show their importance and demonstrate that they can be used as an exclusive source of information for anomaly detection. We take a two-step procedure to reduce the size of the logs produced

by Sysmon. First, we define several rules to include relevant information and exclude network logs as well as less informative and noisy events like empty key retraction processes. We create several datasets in two different virtual environments using the two most common versions of Windows. We implement a fast and highly configurable Sysmon parser that can clean and parse Sysmon logs which also extracts features based on its configuration.
The second step in data reduction is done after parsing. Using clustering techniques, we detect outliers to reduce the size of the data extensively. The outliers are analyzed and flagged in the anomaly detection engine. Here logs are analyzed by all engines available in VirusTotal where the malicious ones are normally detected. A human analyst then can check the few remaining logs that are flagged as suspicious or unknown in the previous step. All attacks are detected successfully at the end.

1.2 Summary of Contributions

The main purpose of this thesis is to extract intelligent insights from host-based system logs to detect anomalies. In summary, the following are the contributions of this thesis:

- Design and implementation of a comprehensive highly configurable Sysmon parser for parsing and extracting features from the logs.
- A hierarchical approach to considerably reduce the volume of Windows generated Sysmon logs while preserving actionable intelligence.

- Design and implementation of an anomaly detection engine that successfully detects attacks and malware on different datasets.

- Creation of 15 different Sysmon events datasets with no modification or anonymization.

- Comparative analysis of the proposed method against state-of-the-art SIEM software.

The results of our experiments show that the proposed anomaly detection is capable of finding various malware in the datasets. The analysis results demonstrate the significance of host-based logs in auditing, security monitoring, and intrusion detection systems.

1.3 Thesis Organization

The remainder of this thesis is organized as follows. Log-based anomaly detection systems, log clustering, and log parsing tools are discussed in Chapter 2. After that, challenges in the area are reviewed in the next Chapter. In Chapter 3, we review endpoint logging and its challenges and a brief background on popular SIEMs. We also provide a detailed description of Sysmon as well as some of its capabilities. In Chapter 4, we discuss the configurations, the datasets, and the malware along with the environment used to create them. Subsequently, details of our framework, implementation, and experiments, as well as the deployment of the system and the proposed anomaly detection system's architecture, are described in Chapter 5. The results of the analysis are presented in Chapter 6. The conclusion and potential future work are discussed in Chapter 7.

Chapter 2

Literature Review

Logs can be used to get a glimpse into every change and state of a computer system. There are different types of logs with various content and formats that are used in particular use cases. Along with log analysis applications, the work by [32] reviews some of the advances in log analysis in the recent decade as well as observing the remaining challenges in the area. But in general, we review log analysis and big data from 4 different aspects mainly log-based anomaly detection, big data for anomaly detection, log clustering, and log parsing.

2.1 Log Based Anomaly Detection

Logs can become massive and noisy. But the quantity of information that they contain makes them an essential source of data to address different security problems such as detecting attackers on a network. In [41], authors designed and implemented a system that automatically analyzes the logs and mines useful information from them to enhance the security of an enterprise network. The experiment was extended by [33] who proposed a complementary tool to detect early-stage enterprise network infections. The focus of the paper is on high-risk attacks with a potential financial loss. The idea is followed in other works as well. A signal coming from the analysis result of network audit logs can become a low-cost security solution for an enterprise network which can cooperate with their current antivirus and other security software to provide a more secure network [13]. However, not all log-based anomaly detection systems rely on network logs. Some works even rely on host-based audit logs as the only source of data for threat hunting. One highly valuable resource to take part in a more extensive security audit process is Windows event logs collection.

The users working on workstations are one of the main targets of the cyberwar that is going on today and sometimes they are the first place where the attacks start from [11]. As an example, the threat detection system suggested by [28] uses Sysmon logs as the data source. Threats are classified into different levels based on their characteristics identified through the analysis of the logs. Similarly, host-based logs are used for security enhancement purposes in different applications.

Monitoring audit logs are not limited to small applications such as anomaly detection, but sometimes they are used for prediction. As suggested by [16],

in large distributed systems, analysis of unstructured logs can be used to detect anomalies inside future generated log files. These logs are also used to help identify insider threats [10]. Console logs and mobile device generated events can be considered as other types of host-based logs which are used for security analysis and threat detection.
Most applications and software contain enough information to detect anomalies by extracting unusual patterns from their console logs [12]. On mobile devices, the analysis of system logs can help detect mobile threats such as smishing and backdoor attacks [24]. Lastly, in [39], the authors discussed the security metrics that can be collected with the use of security logs. They reviewed several scenarios for security metrics collection based on different common security log types. Also, a framework was described that can help with the collection and report process and is implemented based on open-source technologies for log management.

2.2 Big Data for Anomaly Detection

Processing of real-time big data is one of the critical aspects of every log-based anomaly detection system. Gathering and preprocessing massive data constantly generated by components of a network is difficult and hence is usually done using big data techniques. In a work by [29], the authors analyzed several tools to detect mobile devices malware. They mostly lacked sufficient user profiling abilities and therefore failed to automate the process of dynamic analysis of big data for malware detection. Large scale testing of their proposed real-time mobile malware detection framework reveals the need for performance optimization which can be addressed using machine learning algorithms optimized for big data. As another example, in [37], the authors used Azure HDInsight and Apache Spark to handle and analyze NetFlow data for anomaly detection.

2.3 Log Clustering and Mining

Several different approaches can be employed to capture insightful data from the logs. One of the main methods is clustering. Starting with [38], the author proposed SLCT, a data clustering algorithm that is capable of mining and extracting interesting patterns from different types of event logs as well as finding the outliers. Most of the shortcomings of the work were then addressed in [40] where the authors propose an approach that fixes wildcard detection issues along with word shifting sensitivity and delimiter noise. Researchers in [22] suggested a generic log clustering framework named LOGDIG. The framework relies on temporal data (time-stamps) and data related explicitly to the system (e.g., data associated with moving objects). The state-machine-based behavior mining searches for the states and aims to find desired event messages and uses meta-data interpretation for input data. The output of the system is a human-friendly behavioral knowledge report.

In one work by [27], authors introduce IPLoM which is an algorithm for the mining of clusters from event logs. They take a hierarchical 3-step partitioning process which they call iterative partitioning mining. Descriptions of the clusters are presented as the results. The algorithm outperforms both [27] and [40] in terms of precision and performance.
Another approach to log mining is log abstraction. Log abstraction is a technique to convert the messages hidden inside the log files to different event types. In a complementary work by [42] the same clustering algorithm namely LogCluster was implemented in C which greatly improved the speed and efficiency. It was originally written in Perl which is significantly slower than C. This implementation took the crown once again for LogCluster algorithm and was also extended to work with stream mining scenarios. Another example of log abstraction is [31] in which authors presented a comprehensive and straightforward approach to log file abstraction that is similar to the SLCT log clustering tool. They have clustered similar frequency words in each line and abstracted it to event types.

2.4 Log Parsing

Automated methods for parsing logs are critical when it comes to knowledge extraction and anomaly detection from any log [9]. The authors of [19] study four different log parsers and parsing packages to provide a better understanding of the advantages of parsers and their impact on log mining. Their insightful findings substantiate the significance of the information that is obtainable from logs. Their work is further extended by [21] who implement an online parser for web service management techniques. Later, it was updated to function in a streaming manner with dynamic parsing rules [20].

In certain scenarios, having a dedicated parser for each type of log could be a better approach. There are tools available that help with the field retrieval process from different logs. Microsoft came up with a SQL based log parsing tool for logs generated by their services, mainly Windows logs. They discussed the necessity of it and why having a parser is essential for mining data from logs [17]. However, by the time of this writing, the tool is more than 14 years old and does not work properly with newer types of logs including Sysmon logs. The alternatives to most of these parsers are commercial SIEMs. They usually have a user interface and can help with log parsing as well as the analysis and mining process. Nonetheless, their parsing system is generally basic and log messages are mostly treated like text values with no specific structure. In most cases, the end users themselves need to define the parser using regular expressions which requires extensive knowledge of logs and is a time-consuming task [15].

2.5 Concluding Remarks

As a very rich source of information for log analysis and intrusion detection and prevention systems, host-based logs have the potential to be used to hunt threat and malicious activities on different machines in an enterprise. Most of existing log-based anomaly detection systems are generally built towards firewall and network logs to perform the analysis and prevent threats. On the other hand, host-based logs are proven to have rich enough information to be used as the sole source of information for anomaly detection. A large number of existing studies in this domain are also more reliant on network logs than host-based logs. This makes available tools and resources to fully take advantage of host-based logs to be limited and lacking. Also, in some areas like log volume reduction, there is so much left to be done. We are motivated to address these shortcomings by utilizing host-based logs to their full potential.

Chapter 3

Endpoint Logging for Anomaly Detection

In this chapter, we discuss our motivation for log-based security solutions and the challenges of endpoint logging for anomaly detection systems in general. We also overview the most common solutions to log-based anomaly detection. Furthermore, we introduce Sysmon, a Windows-based log generation tool introduced by Microsoft and discuss its abilities for customized log generation in a Windows environment.

3.1 Motivation and Challenges

Although significant progress has been made in the area, several challenges make the detection and response process more difficult, and some of them are still fully open. As an example, built-in Windows logging and auditing tools, while useful, are for the most part not informative enough to address these security issues. First off, the information provided by these tools is limited especially for sensitive files like DLLs and for some critical events such as process creation. Moreover, network logging in Windows is too limited, messy, and often obscure. These limitations make it hard to track and capture the attackers behavior [41]. Therefore, several other complementary tools have been developed to address these issues. The following are some of the challenges we face in log analysis-based security solutions.

- **Data Sources:** The first challenge is sources of the logs, which is the process of choosing the right logging tool to capture events. Logging tools have their advantages and disadvantages regarding accuracy, dependability, configurability, and performance impact. The tools of choice must meet some minimum requirements to be useful. The importance of having the right data to analyze and get the correct result is undeniable. Having several different tools and software and hundreds of ways of configuration and deployment make this process time consuming and difficult.

- **Data Generation:** Logging processes and producing security logs need extensive management and effort. Data must be accurate, comprehensive, and dependable. It also must give a good enough representation of what is happening in the system. We need to make sure the data is first large enough, and second, cover different types of activities so that we can make sure we will have a complete package of raw log data to analyze on. The essential data properties include size, content, realism, and accuracy.

- **Data Preprocessing:** Security-relevant logs in a large enterprise can grow by terabytes per day. Handling, cleaning, and processing this massive volume of security log files remains one of the main challenges. Moreover, parsing this data adds another problem. While there are tools that can help with this part, depending on the type of logs, it could be challenging to find the right parser. In some cases, a parser is not provided so we will have to implement our parser to be able to extract meaningful data and features out of the raw logs.

  Analysis Process: The last challenge is providing actionable insights from mining large quantities of security log data and finding anomalous behaviors from it. Statistical techniques could be helpful, but they do not explain what we can do about the malicious pattern that is found and where to go from there. Also, a subjective interpretation of the analysis results, as well as metrics to validate the informativeness of the results, could become complicated.

3.2 Security Log Analysis

Security Information and Event Management agents are one of the most common destinations of any logs including Sysmon logs. They provide insights from the logs and analyze them in real-time. The main idea of having a SIEM is to aggregate similar logs and events from various sources and combine and manage all of them together to extract meaningful insights and detect possible deviations. SIEMs are also used as the security control center and for monitoring and can usually be configured to generate an alert when something suspicious happens. There exist a lot of similar software applications for log analysis but SIEMs are usually the first choice of enterprises for this type of security solution.
The process of collection, archival, analysis and correlation of the log data is used to producing reports using all information gathered from various sources. This is what deploying a Security Information and Event Management (SIEM) software in an organization is intended for. The intersection of the Security Event Management (SEM) and Security Information Management (SIM) technologies is called Security Information and Event Management (SIEM) [5].

3.3 Popular SIEMs

While there are many different SIEM and related software as well as thousands of plugins, a few of them stand out as more sophisticated popular

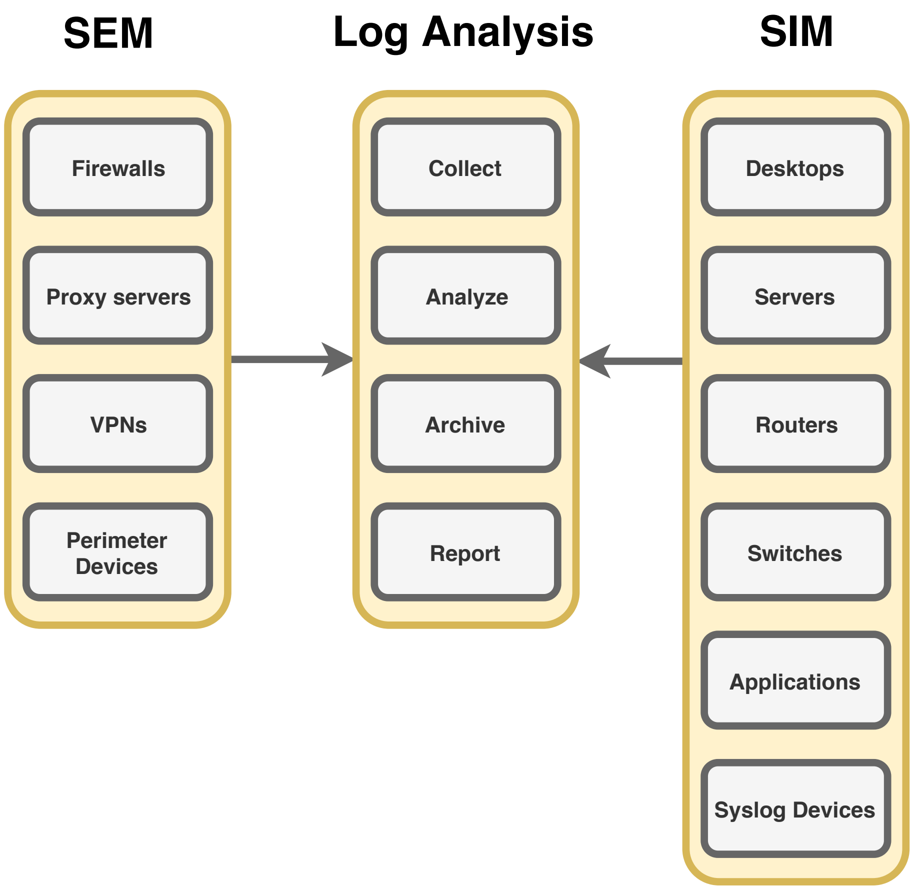


Figure 3.1: The relation between Security Information Management (SIM) and Security Event Management (SEM) [5]

tools commonly used by enterprises. Here is a brief overview of 3 of the more commonly used SIEMs.

3.3.1 Elastic Stack

The Elastic Stack is a set of open-sources software and tools for data analysis. ELK (Elasticsearch, Logstash, Kibana) can gather logs from different distributed sources.

It can index and store events to help the user query and visualize the logs. The architecture of the ELK Stack is depicted in Figure 3.2.

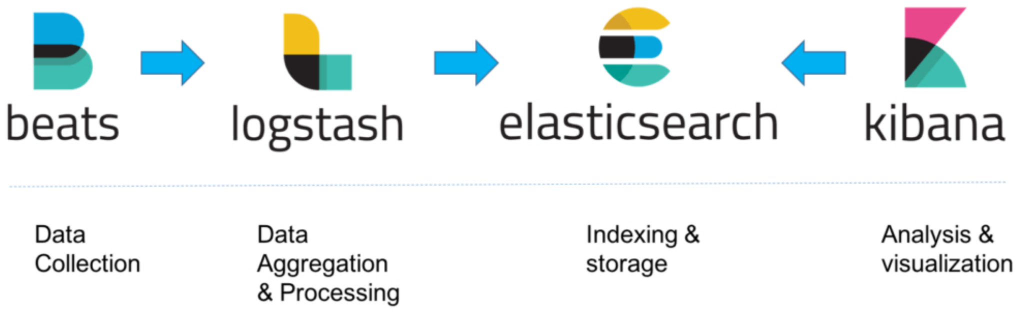
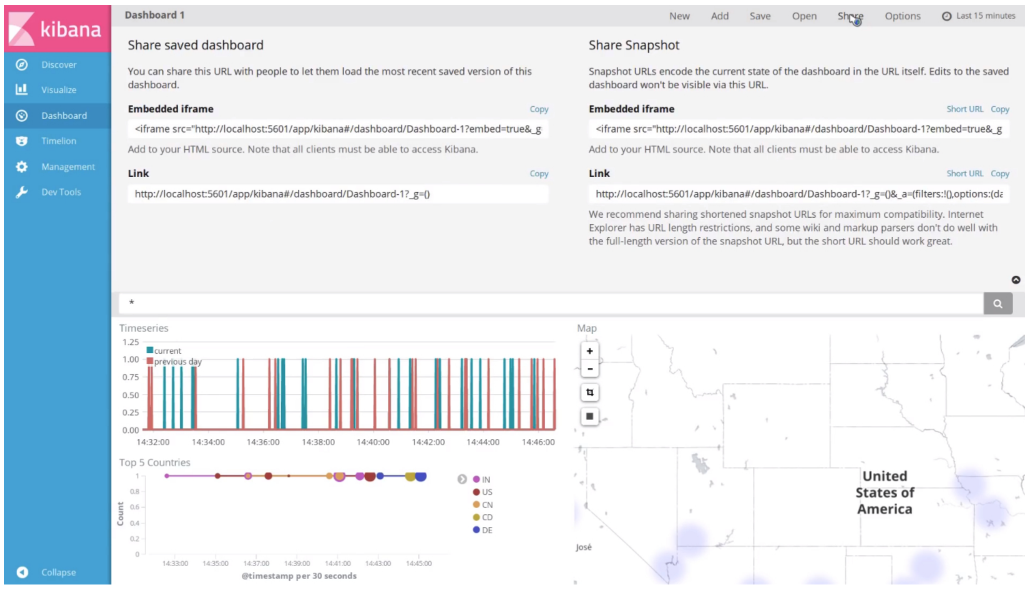


Figure 3.2: ELK Stack architecture [25]

As an open-source data processing pipeline, Logstash is a server-side tool that can gather data from several sources simultaneously. The data are then transformed and sent to the analysis module. In this case, the analyzer is Elasticsearch, which is a search engine that provides meaningful information and queries from the input data that can be used to detect threats and generate alerts. The third tool in this stack is called Kibana. Kibana is the visualization engine in the ELK Stack. It can help visualize and prepare Elasticsearch data for analysis. Figure 3.3 shows a Kibana environment where an analyst can interact with the tools and data.



Share saved dashboard

You can share this URL with people to let them load the most recent saved version of this dashboard.

Embedded iframe

```
<iframe src="http://localhost:5601/app/kibana#/dashboard/Dashboard-1?embed=true&_g
```

Add to your HTML source. Note that all clients must be able to access Kibana.

Link

```
http://localhost:5601/app/kibana#/dashboard/Dashboard-1?_g=()
```

Share Snapshot

Snapshot URLs encode the current state of the dashboard in the URL itself. Edits to the saved dashboard won't be visible via this URL.

Embedded iframe

```
<iframe src="http://localhost:5601/app/kibana#/dashboard/Dashboard-1?embed=true&_g
```

Add to your HTML source. Note that all clients must be able to access Kibana.

Link

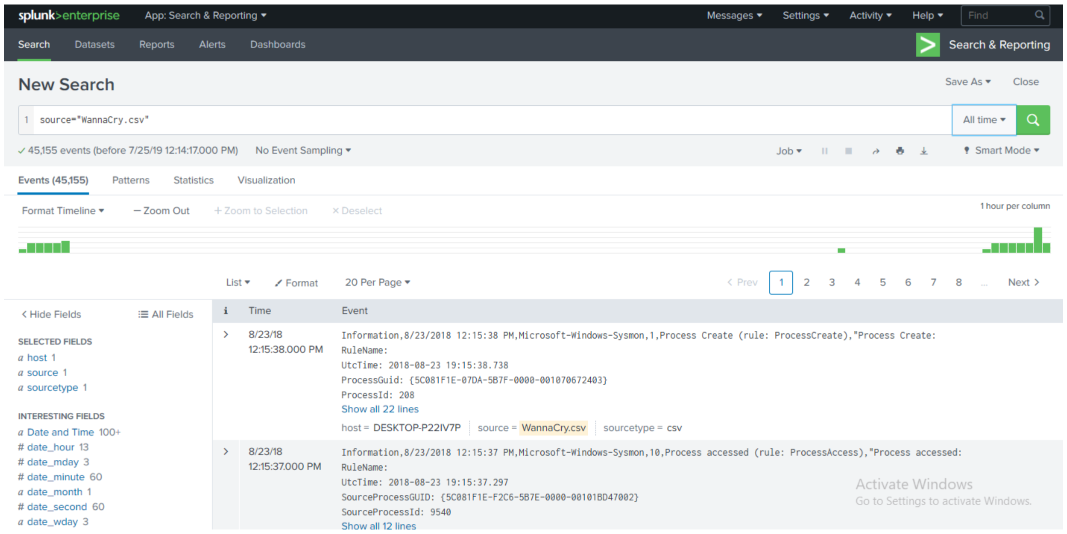
```
http://localhost:5601/app/kibana#/dashboard/Dashboard-1?_g=()&_a=(filters:!(),options:(da
```

We recommend sharing shortened snapshot URLs for maximum compatibility. Internet Explorer has URL length restrictions, and some wiki and markup parsers don't do well with the full-length version of the snapshot URL, but the short URL should work great.

Figure 3.3: A Kibana environment, the visualization engine implemented to visualize Elasticsearch data.

3.3.2 Splunk

Splunk is a software platform that can search, analyze, and visualize various types of logs and other machine-generated data. Splunk supports several different sources of input data streams and can provide analytical insights from them. Splunk's add-on for Microsoft Sysmon helps import Sysmon data and extract relevant fields from the logs, which makes it a common destination for Sysmon logs for the analysis and threat detection.



```
source="WannaCry.csv"
```

✓ 45,155 events (before 7/25/19 12:14:17.000 PM)   No Event Sampling

SELECTED FIELDS
- a host 1
- a source 1
- a sourcetype 1

INTERESTING FIELDS
- a Date and Time 100+
- # date_hour 13
- # date_mday 3
- # date_minute 60
- a date_month 1
- # date_second 60
- a date_wday 3

Time	Event
8/23/18 12:15:38.000 PM	Information,8/23/2018 12:15:38 PM,Microsoft-Windows-Sysmon,1,Process Create (rule: ProcessCreate),"Process Create: RuleName: UtcTime: 2018-08-23 19:15:38.738 ProcessGuid: {5C081F1E-07DA-5B7F-0000-001070672403} ProcessId: 208 Show all 22 lines host = DESKTOP-P22IV7P source = WannaCry.csv sourcetype = csv
8/23/18 12:15:37.000 PM	Information,8/23/2018 12:15:37 PM,Microsoft-Windows-Sysmon,10,Process accessed (rule: ProcessAccess),"Process accessed: RuleName: UtcTime: 2018-08-23 19:15:37.297 SourceProcessGUID: {5C081F1E-F2C6-5B7E-0000-00101BD47002} SourceProcessId: 9540 Show all 12 lines

Figure 3.4: A Splunk environment while analyzing WannaCry events.

Some of Splunk's features are as follows [2]:

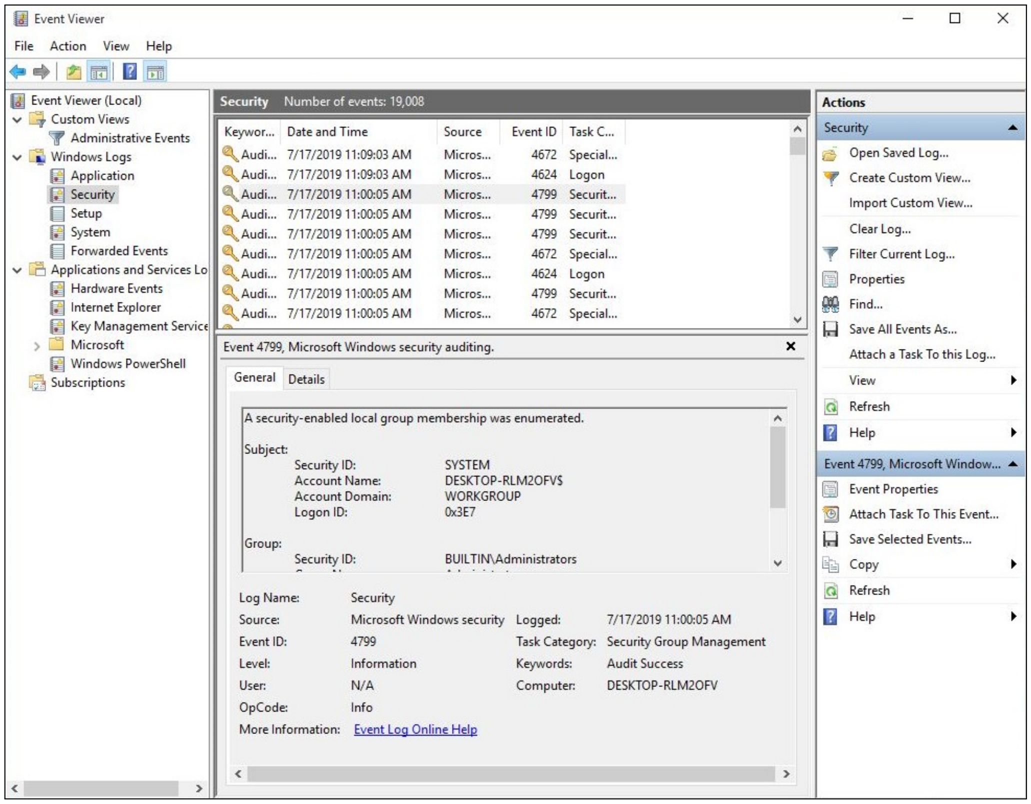
- Gathering all the logs in one place. This makes the search, aggregation, and correlation process easier.
- On top of what Sysmon has to offer in terms of customizability, using Splunk, an administrator can further filter events that are unnecessary including the ones of low value. It can also help reduce operational costs by reducing network communications.
- Collecting information from the host and also any logs that are not necessarily stored in the Windows Event Viewer.

3.3.3 QRadar

QRadar is a SIEM and a platform that provides an all-around overview of an organization's security systems. It is used to monitor and detect security threats and suspicious behaviors and report them. QRadar normalizes all the incoming events from various log sources and based on the configuration and the pre-defined rules, correlates the data and makes them ready for analysis. QRadars Sysmon integration is a content extension that can help detect threats on Windows workstations using Sysmon endpoint logs.
The complicated process of choosing the right software aside, analysis and identification of malicious activities using SIEMs could become costly. They need much effort to implement, install, and properly configure. Also, the costs of running a SIEM could be high and based on the scale and the quantity of the data it is processing, it could easily become a reason for companies to ignore it as not cost-effective [23]. Even after proper configuration, using a SIEM to analyze logs is not a simple process. It normally requires prior knowledge as well as time and practice.

3.4 Windows Event Logs

Windows logs and records its events in the Windows Event Viewer. This is a built-in Windows application that records and manages logs and event information in Windows. The logs include system errors, logs of applications, system messages, warnings, and other system information. It is used to troubleshoot and track activities of the applications and the overall state of the system.



Security Number of events: 19,008

```
Keywor...  Date and Time          Source   Event ID  Task C...
Audi...    7/17/2019 11:09:03 AM  Micros...    4672  Special...
Audi...    7/17/2019 11:09:03 AM  Micros...    4624  Logon
Audi...    7/17/2019 11:00:05 AM  Micros...    4799  Securit...
Audi...    7/17/2019 11:00:05 AM  Micros...    4799  Securit...
Audi...    7/17/2019 11:00:05 AM  Micros...    4799  Securit...
Audi...    7/17/2019 11:00:05 AM  Micros...    4672  Special...
Audi...    7/17/2019 11:00:05 AM  Micros...    4624  Logon
Audi...    7/17/2019 11:00:05 AM  Micros...    4799  Securit...
Audi...    7/17/2019 11:00:05 AM  Micros...    4672  Special...
```

Event 4799, Microsoft Windows security auditing.

```
A security-enabled local group membership was enumerated.

Subject:
        Security ID:        SYSTEM
        Account Name:       DESKTOP-RLM2OFV$
        Account Domain:     WORKGROUP
        Logon ID:           0x3E7

Group:
        Security ID:        BUILTIN\Administrators
```

```
Log Name:          Security
Source:            Microsoft Windows security   Logged:         7/17/2019 11:00:05 AM
Event ID:          4799                         Task Category:  Security Group Management
Level:             Information                  Keywords:       Audit Success
User:              N/A                          Computer:       DESKTOP-RLM2OFV
OpCode:            Info
More Information:  Event Log Online Help
```

Figure 3.5: Event Viewer is an application and a component of Windows New Technology (NT) operating systems lineup.

While being useful in many ways, Windows' built-in tool lacks many required features for security oriented log analysis. The captured information by default is limited for some critical events such as process creation, file creation, and file modification as well as DLL loading. On the other hand, the provided information regarding the network connections is too limited and contains so much irrelevant information which is logged just because the application needs to log too much information to make sure the necessary events are logged. This is not ideal and makes the process of analysis troublesome. Lastly, it is not designed with only security in mind. The security logs are parts of it but not enough effort has been done in terms of security. In other words, it does not include the required tools and information to capture some of the most common attacks such as threat injection [34].

3.5 Sysmon

Sysmon is a sophisticated logging tool designed for Windows workstations by Microsoft. It is a device driver and is run as a Windows service that records system's activities and events.

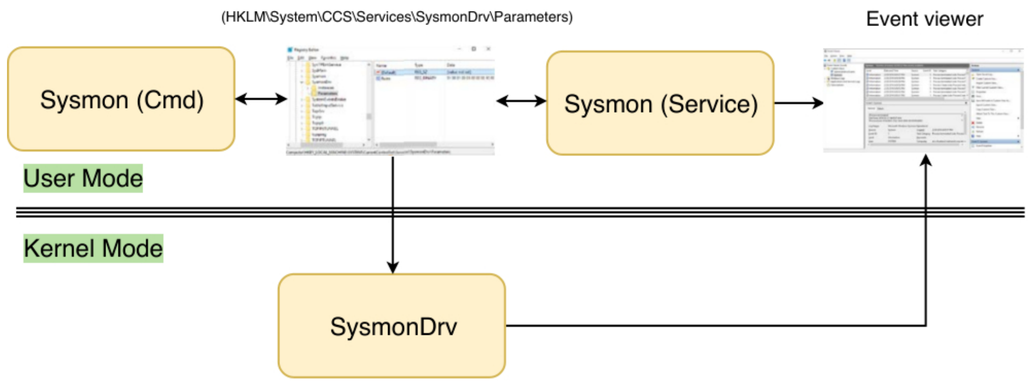


Figure 3.6: Sysmon architecture [34]

The logs are stored in the Windows event log application. Logs generated by Sysmon, in general, contain more information and details than the ones generated using built-in Windows audit logging system. Sysmon logs provide information about process creations, file access and modification, network access logs and several other types of events. Some of Sysmon's capabilities are as follows: Logging the process creation with full command line level details, recording multiple hashes of process image files at the same time, process and session global IDs, logging driver loads and DLLs along with their hashes, recording file creation events with their accurate and real modification times and several other types of events.

Table 3.1: Sysmon Event IDs for each category

Category	ID	Category	ID
Sysmon Service Status Changed	0	Registry Object Create Delete	12
Process Create	1	Registry Value Create	13
File Creation Time Changed	2	Registry Object Rename	14
Network Connection	3	File Create Stream Hash	15
Sysmon Service State Change	4	Sysmon Configuration Changed	16
Process Terminated	5	Pipe Created	17
Driver Loaded	6	Pipe Connected	18
Image Loaded	7	WMI filter	19
Create Remote Thread	8	WMI consumer	20
Raw Access Read	9	WMIN consumer filter	21
Process Access	10	DNS query	22
File Create	11	Error	255

Table 3.1 is a complete list of events that Sysmon can record along with the mapping between each Sysmon category and its associated event ID. Since Sysmon is highly configurable, the administrator can specify what events are to be logged. This way, the user can not only reduce the number of logs generated but also can add or remove different types of logs based on their usefulness.
The following are different types of events generated by Sysmon [7].

- **Event ID 1: Process creation:** Provides extended information on the newly created processes. Context on the process execution as well as unique ID and information necessary to correlate the process across a domain are provided by the Command Line and Process GUID values respectively.

- **Event ID 2: A process changed a file creation time:** When a file creation time modification is performed by a process, this event can help to track and find out the real time that the process was created. Change in time creation of files can be used by attackers to create backdoors that look legitimate because they existed when the operating system was installed.

- **Event ID 3: Network connection:** A network connection is an event that records and logs TCP and UDP connections on the Windows machine. This event needs to be enabled and configured before it can log the connections.

- **Event ID 4: Sysmon service state changed:** Sysmon service state change shows whether Sysmon services are running or not (have started or stopped).

- **Event ID 5: Process terminated:** This event logs a report when a process terminates. UTC time, Process ID, and Process GUID are also logged in this event.

- **Event ID 6: Driver loaded:** When a driver is loading on the system, this event provides information about it. Hashes, as well as signature information, are also recorded by this event type. Using a signature one can find out whether the driver has been removed at a later time after loading or not.

- **Event ID 7: Image loaded:** This event records if a module is loaded in a process. It provides various information about the image as well as the process in which the module has been loaded. This information include hashes and signature information. This event log needs extensive configuration as it can generate a large number of logs if set to monitor all image load events.

- **Event ID 8: CreateRemoteThread:** The Create Remote Thread records the log when some process generates a thread in another process. Malwares that try to inject code in other processes are caught by this log. Source and target are specified and various information about the code running in the other process is recorded by the event.

- **Event ID 9: RawAccessRead:** The Raw Access Read event is to log and detect if a process is conducting reading operations. Some malware use this method to data exfiltrate locked files or the files with limited read access. Auditing tools are also detected by this event log. The event records both the source process and the target device.

- **Event ID 10: ProcessAccess:** Process Access event logs if a process has accessed another process. This is usually followed by a sequence of actions such as read and write operations and memory access. Hacking tools that rely on reading memory contents will be detected by Process Access event.

- **Event ID 11: FileCreate:** A File Create event records when a file is created. It also logs when a file is overwritten. Detection of the initial malware drop location is one useful application of this event.

- **Event ID 12: RegistryEvent (Object create and delete):** This event logs when a registry operation occurs. When a registry key and value is created or deleted, this event records it. It is useful to detect if a malware has changed or modified the registry.

Table 3.2: Mapping of Registry root key names to abbreviated versions used by Sysmon

Key name	Abbreviation
`HKEY_LOCAL_MACHINE`	`HKLM`
`HKEY_USERS`	`HKU`
`HKEY_LOCAL_MACHINE\System\ControlSet00x`	`HKLM\System\CurrentControlSet`
`HKEY_LOCAL_MACHINE\Classes`	`HKCR`

- **Event ID 13: RegistryEvent (Value Set):** This event type tracks and records modifications in Registry values. Any value written for DWORD and QWORD value types will be recorded by this event.

- **Event ID 14: RegistryEvent (Key and Value Rename)** This event type tracks and records modifications in Registry values or names. Any new key name or value will be recorded by this event.

- **Event ID 15: FileCreateStreamHash** When a named file stream is created, this event type records and logs the hashes of the contents of the file stream.

- **Event ID 16: Sysmon Configuration Changed** This records if Sysmon configuration file has been changed.

- **Event ID 17: PipeEvent (Pipe Created)** This event logs of pipe with a name is created. Interprocess communication by malware is often associated with named pipes.

- **Event ID 18: PipeEvent (Pipe Connected)** When a named pipe connection is established between a server and a client, this event type logs it.

- **Event ID 19: WmiEvent (WmiEventFilter activity detected)** Filter name, WMI namespace, and filter expression of a newly registered WMI event filter are recorded by this event type.

- **Event ID 20: WmiEvent (WmiEventConsumer activity detected)** Name and log, as well as the destination of a consumer that binds to a filter, are logged by the event type.

- **Event ID 21: WmiEvent (WmiEventConsumerToFilter activity detected)** Name and filter path of a consumer that binds to a filter, are logged by the event type.

- **Event ID 22: DNSEvent (DNS query)** If a process executes a DNS query, regardless of the results being successful or not and regardless of it being cached or not, this event is recorded.

- **Event ID 255:** When an error occurs within Sysmon, this event is generated. It could be the result of a heavily loaded system or an internal Sysmon bug.

3.6 Concluding Remarks

In this chapter, we discussed log analysis in Windows environments, the motivation and challenges of log-based security solutions and also introduced some of the most commonly used log analysis software applications used by enterprises in large networks. This chapter introduced Sysmon, a Windows-based logging tool developed by Microsoft along with some of its capabilities. We discussed the reasons why Sysmon is a reliable replacement for the built-in Windows logging tool. However, as Sysmon is new and far from being widely used, the resources available as to how to deploy it properly are limited. This is mainly due to the lack of support by analysis software and parsing tools. We will address some of these issues in the following chapters.

Chapter 4

Configuration and Dataset

4.1 Sysmon Configuration

The first contribution of this thesis is proposing a proper way to reduce the enormous volume of Windows generated log data while preserving the interesting information inside them. To monitor and trace the events using Sysmon, while not necessary, it is highly recommended to have a configuration based on the use case. The filtering abilities of Sysmon are a part of what makes it a more sophisticated tool than the built-in Windows tools for auditing.

Based on the configuration suggested by [36], we made a configuration file that carefully filters events that are probably not malicious or do not help with the security investigation process. Network traffic was also filtered as the main idea of the project was to focus on host-based logs. Furthermore, the logs generated by the virtual environment (AWS and VirtualBox) were also filtered as they are not present in a real-world physical machine. It is worth noting that as suggested in the base configuration, some of the highly sensitive network logs were not filtered as they might be needed in the investigation process and for the future work. The details of the configuration file are discussed in Section 4.2. It is expected that any malicious activity and behavior would be detected at its early stages of running on the machine.

4.2 Configuration Review

Since the configuration file is so large and usually can be any size from several hundred to thousands of XML code lines, we briefly review some of the more important sections of it to get an idea of how our Sysmon configuration file works.

4.2.1 Process Access Exclusions

One important part of any Sysmon configuration file is Process Access rules. While most process access events are important, since some processes generate a large number of events and we are sure that they are part of the tools we are using, we can simply exclude them. This way, instead of including what needs to be logged, we can exclude things that we are actually sure that are not important or we have full control of. These can include software like Process Explorer which is a free tool that acts as a more sophisticated task manager and system monitor tool in the Windows environment. This is used by us to monitor the processes that are being logged so we can exclude the logs that are generated by this process itself. Windows' built-in task manager is also excluded for the same reason. One other example of what should be excluded could be the logs generated by virtual machine-related software and drivers. Since we are not running the test Windows clients on physical machines, we can simply exclude the logs generated by the virtual machine's drivers.

```
<ProcessAccess onmatch="exclude">
            <SourceImage condition="contains">procexp64.exe</SourceImage>
            <SourceImage condition="is">C:\WINDOWS\System32\Taskmgr.exe</SourceImage>
            <SourceImage condition="is">C:\WINDOWS\System32\VBoxService.exe</SourceImage>
            <SourceImage condition="is">
              C:\Program Files (x86)\VMware\VMware Workstation\vmware-authd.exe</SourceImage>
</ProcessAccess>
```

Figure 4.1: An example of excluding some software and virtual machine related logs.

4.2.2 Process Access Inclusion

Process Inclusion rules are also an important part of a well written Sysmon configuration file. While we are able to include every process and exclude the ones that are not necessary, defining what exactly should be logged is a better way to keep track of what processes have been accessed by which user or other processes. It will also help keep the load that Sysmon puts on the system minimal; thus the resources are used more efficiently. Some processes like LSASS are important for security analysis of the logs.

Enforcing security policies in Windows is done by LSASS whose tasks include verifying users that are logging on to the machine, handling and keeping track of password changes, creating access tokens, etc. LSASS is also responsible for writing Windows Security logs which is a part of Windows' built-in logging and auditing tool. Since LSASS is an important process and the name lsass.exe is often used by various malware to bypass antivirus software and look legitimate, and also can be accessed by malware and other hacking tools to potentially steal passwords and sensitive data from memory, it is very important to monitor it.

Sysmon is also able to monitor inter-process access. Processes accessing other processes or memory are some examples of this type of access. Since it generates an enormous number of logs and puts a substantial load on the system resources, we do not include it in our general-purpose configuration.

```
<ProcessAccess onmatch="include">
            <TargetImage condition="is">C:\Windows\system32\lsass.exe</TargetImage>
            <TargetImage condition="is">C:\Windows\system32\winlogon.exe</TargetImage>
            <TargetImage condition="is">C:\Windows\system32\explorer.exe</TargetImage>
</ProcessAccess>
```

Figure 4.2: An example of including rules in the configuration file.

4.2.3 Network-related Events

While network logs are not a part of the data that are intended for this study, namely host-based log analysis in the Windows environment, this configuration logs some of the sensitive network activities, which might be necessary for malicious activity detection. Similarly to Process Access events, we can include and exclude what we do and what we do not need. In this configuration, as suggested in the base configuration, we include some critical network-connecting binaries and the suspicious sources that can be used by them. Similarly, we exclude some of the binaries that are safe to eliminate but generate noisy data like Spotify and Dropbox, both of which are constantly using network communication.

```xml
<NetworkConnect onmatch="exclude">
            <Image condition="image">Spotify.exe</Image> <!--Spotify-->
            <Image condition="end with">AppData\Roaming\Dropbox\bin\Dropbox.exe</Image> <!--Dropbox-->
            <Image condition="image">OneDrive.exe</Image> <!--Microsoft:OneDrive-->
            <Image condition="image">OneDriveStandaloneUpdater.exe</Image> <!--Microsoft:OneDrive-->
            <Image condition="end with">AppData\Local\Microsoft\Teams\current\Teams.exe</Image>
            <!--Microsoft: Teams-->
            <DestinationHostname condition="end with">microsoft.com</DestinationHostname>
            <!--Microsoft:Update delivery-->
            <DestinationHostname condition="end with">microsoft.com.akadns.net</DestinationHostname>
            <!--Microsoft:Update delivery-->
            <DestinationHostname condition="end with">microsoft.com.nsatc.net</DestinationHostname>
            <!--Microsoft:Update delivery-->
</NetworkConnect>
```

Figure 4.3: An example of excluding network connection rules in the configuration file.

4.2.4 File Creation and Modification Events

To monitor what files are being created and/or modified on the system, we need to configure Sysmon for file creation and modification events. A File Create event is generated when a new file is created either by the user or by a process. Overwritten files are also considered to be newly created files, thus they are logged as well. Events of this kind are very important when tracing malicious activities. They are particularly useful to find out when and where malware is offloaded into the system and where it moves or is copied from. Also, some malware modify the file creation times in order to look like a legitimate file. An example of this can be changing the creation time of a file to an earlier time like when the operating system was first installed. This way, malware might be able to disguise itself from some time-dependent, signature-based malware detection techniques. Sysmon allows the user to monitor this type of change, which ultimately will help catch the malware easier. Like other event types, file creation and modification events should be logged carefully to prevent inefficient use of the system resources and also to prevent generating noisy and unnecessary logs.

In Figure 4.4, it is shown that some file-related activities by some processes can be safely ignored. For example, Windows update-related file operations are in this category.

```
<FileCreateTime onmatch="exclude">
            <Image condition="image">OneDrive.exe</Image>
            <Image condition="image">C:\Windows\system32\backgroundTaskHost.exe</Image>
            <Image condition="contains">setup</Image>
            <Image condition="contains">install</Image>
            <Image condition="contains">Update\</Image>
            <Image condition="end with">redist.exe</Image>
            <Image condition="is">msiexec.exe</Image>
            <Image condition="is">TrustedInstaller.exe</Image>
</FileCreateTime>
```

Figure 4.4: An example of excluding file creation rules in the configuration file.

4.3 Configuration Deployment

Configuration deployment is the next step towards setting up a Sysmon-based logging system. It is an important process which needs to be done simultaneously across all the machines in an enterprise network. When a new configuration is set, Sysmon automatically replaces it with the old configuration. From that point onwards, the logs are generated based on the most recent configuration, and the old one is deleted. Therefore, the administrators need to make sure that they take the changes into consideration while doing analysis on the data.

A Sysmon event ID 16 is generated when the configuration is updated. This way, the administrators can distinguish between the logs generated based on the old and new configurations. After deploying the configuration, the number of logs generated is hugely reduced. In some cases, the reduction can be close to 95% compared to when we use Sysmon to log every action that is performed on the system. Also, compared to the built-in Windows logging system which stores the logs in different sections, when combined, the result is outstanding, and it is kept above 90% reduction in volume all the time. This is the first step of a two-step volume reduction process. The details of the second step are explained in Section 5.5.

4.4 Dataset Creation

4.4.1 Environment

Windows, as the most widely used desktop operating system, is one of the main targets of attackers [14]. Motivated by this and the fact that Sysmon is a tool written specifically for Windows, we decided to gather and analyze host-based logs, which are for the most part different for every operating system. To achieve this goal, we created five different datasets, three of which contain malicious activity and two of which are benign. For each dataset, either AWS or VirtualBox was used as a host to install the operating system. One of the two benign datasets was created using VirtualBox and the other one was an AWS based virtual machine. This ensured that we have a ground truth to know what normal activities in each environment would look like.

In order to create the log data, we prepared a Windows environment that was as close to a real-world scenario as possible. Since one of the main targets of attackers is enterprise workstations, the test Windows environments were fully equipped with normal day-to-day office tools and applications like

Acrobat Reader, Microsoft Office, Slack, Google Chrome, Spotify etc. As for the version of Windows operating system, Windows 10 was used for 4 datasets and Windows 7 was used to create the last one.

4.4.2 Logging Process

The logging process for each dataset was slightly different in terms of system usage, applications, and the duration of running the operating system. However, all datasets were generated using similar behaviors and baselines. The final datasets have tens of thousands of events, each of which normally contains more than 20 lines of information.

The systems were run from a few days to a couple of weeks until the malware cause them to crash or they were stopped manually. The systems were constantly turned on and off daily, like they would be in normal use, and in some cases were kept on for several days. In the end, some reached more than 45,000 events and over a million lines of logs. Thanks to the configuration, we managed to shrink the log data to a relatively small set while keeping useful events. Some datasets are slightly smaller than others. More than 20 sets of data were generated in six months, five of which were chosen for the experiments. Datasets are not modified or anonymized.

4.4.3 Primary Datasets

To test the malware detection engine, we needed to run malware on some of the datasets. Some of the most well-known malware were chosen for the experiments. A ransomware, a Trojan and an adware were selected, which are explained as follows:

WannaCry: Ransomware is a computer malware that locks the user's file data using encryption techniques and then holds the entire or some of the device data hostage until the user pays the ransom to the attacker. The attacker then might give the decryption key to the user so that they can recover the files. Ransomware is one the most common types of malware these days, and WannaCry attack was one of the most destructive ransomwares ever known [30].

Kuaizip: As an adware, Kuaizip is bundled with several free but potentially unwanted programs that are usually installed on users' computers without them asking for it and, in some cases without users' knowledge. However, while in most cases they can do what they were advertised for, they might also collect the victims' online and offline activities and perform unwanted actions. Kuaizip is a free-to-download software that claims to be a highly effective file archiver application. This adware auto starts with Windows, tracks online activity, adds unknown background services, injects advertising banners on visited web pages, turns random web pages texts into malicious URLs and installs other unwanted programs. These are only some of the malicious activities of this adware.

```
<EventID>1</EventID>
<TimeCreated SystemTime='2018-10-11T21:56:47.330914000Z'/>
<EventRecordID>17836</EventRecordID>
<Execution ProcessID='1964' ThreadID='2068'/>
<Channel>Microsoft-Windows-Sysmon/Operational</Channel>
<Computer>test-PC</Computer>
<Security UserID='S-1-5-18'/>
<EventData>
<Data Name='RuleName'></Data>
<Data Name='UtcTime'>2018-10-11 21:56:47.315</Data>
<Data Name='ProcessGuid'>{3904188F-C71F-5BBF-0000-00100EA60B00}</Data>
<Data Name='ProcessId'>3016</Data>
<Data Name='Image'>C:\Program Files\KuaiZip\Update.exe</Data>
<Data Name='FileVersion'>1, 0, 0, 1</Data>
<Data Name='Description'>kuaizip</Data>
<Data Name='Product'>Kuaizip automatic updates application</Data>
<Data Name='Company'>Suzhou Shijie Software Co., LTD</Data>
<Data Name='CommandLine'>"C:\Program Files\KuaiZip\Update.exe" -ke 1</Data>
<Data Name='CurrentDirectory'>C:\Windows\system32\</Data>
<Data Name='User'>test-PC\test</Data>
<Data Name='LogonGuid'>{3904188F-C637-5BBF-0000-00209B240100}</Data>
<Data Name='LogonId'>0x1249b</Data>
<Data Name='TerminalSessionId'>1</Data>
<Data Name='IntegrityLevel'>Medium</Data>
<Data Name='Hashes'>MD5=960055F63557F450EBDA9F8CC293D659,
                    SHA256=233D809EE5756E1AB2FD9B552E154DE772A83A83031A24E791778601AB2A676B</Data>
<Data Name='ParentProcessGuid'>{3904188F-C71F-5BBF-0000-0010E39A0B00}</Data>
<Data Name='ParentProcessId'>3036</Data>
<Data Name='ParentImage'>C:\Program Files\KuaiZip\KZMount.exe</Data>
<Data Name='ParentCommandLine'>"C:\Program Files\KuaiZip\KZMount.exe" -ke</Data>
</EventData>
```

Figure 4.5: The Sysmon log event in XML format related to Kuaizip adware

CoinMiner: With the recent rise in cryptocurrency prices, dozens of unwanted malicious software that mine cryptocurrencies have been developed by attackers. These programs, while in most cases they do not threaten the users and their data, are run on the victim's machines and use their resources to mine cryptocurrencies for the attacker. CoinMiner, which is a Bitcoin miner Trojan was chosen for the experiments. This sophisticated malware hides and automatically terminates when the user tries to check the systems resources. It will appear neither in task manager nor any other user-accessible application. The Trojan then relaunches after the task manager is closed, which makes it hard to detect by the user.
The version of the Windows operating system, as well as other details of the primary datasets, are provided in Table 4.1.

Table 4.1: Details of the primary datasets.

Dataset	Environment	Operating System	No. of Events	No. of Lines	Days of Run
CoinMiner	Amazon Web Services	Windows 10	33793	777151	16
Kuaizip	Oracle VM VirtualBox	Windows 7	12728	292647	8
WannaCry	Oracle VM VirtualBox	Windows 10	45156	1038488	5
Benign	Amazon Web Services	Windows 10	34697	797345	17
Benign	Oracle VM VirtualBox	Windows 10	30024	690478	10

4.4.4 Secondary Datasets

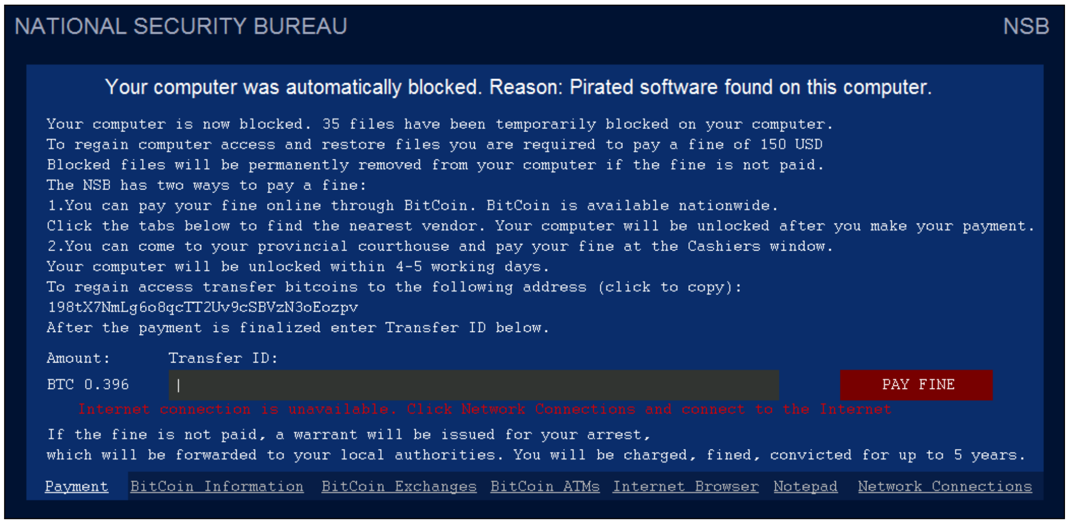
Unlike the primary data, in this part, we create 10 different malware datasets with a different approach. Instead of trying to mimic a real user's behavior, we only ran the malware for a short period of time on a clean installed Windows 10 with no third-party application installed except Sysmon and the pre-installed system applications.

The running time of all datasets was approximately the same. This ensured that the difference in the datasets is minimal besides the changes caused by the malware that is run on the system. Furthermore, even highly sensitive network logs were ignored, and the machine had no network connection. This helps to identify the nature of the malware before it is triggered by an external actor. Malware chosen for this part were selected based on the top 10 Windows malware in 2017/2018 reported by [4]. The table 4.2 represents the list of malware along with their share of Windows malware.

The primary goal of gathering this set of malware and creating this dataset was to understand the exact changes the malware makes to the operating system and also to find the exact differences between a clean installation of Windows 10 before and after infection by malware. This way, distinguishing the changes solely caused by malware is easier. The findings of this part were mostly used in the analysis, design, and implementation of the main anomaly detection engine.

Table 4.2: List of the top 10 Windows malware in 2017/2018 reported by [4] along with their share of Windows malware.

Rank	Malware	Share
1	Ramnit	21.54%
2	Agent	13.78%
3	Virut	6.37%
4	Virlock	6.25%
5	Allaple	4.69%
6	VB	3.98%
7	Sivis	2.39%
8	Upatre	2.03%
9	Injector	2.00%
10	Kryptik	2.00%



Your computer was automatically blocked. Reason: Pirated software found on this computer.

```
Your computer is now blocked. 35 files have been temporarily blocked on your computer.
To regain computer access and restore files you are required to pay a fine of 150 USD
Blocked files will be permanently removed from your computer if the fine is not paid.
The NSB has two ways to pay a fine:
1.You can pay your fine online through BitCoin. BitCoin is available nationwide.
Click the tabs below to find the nearest vendor. Your computer will be unlocked after you make your payment.
2.You can come to your provincial courthouse and pay your fine at the Cashiers window.
Your computer will be unlocked within 4-5 working days.
To regain access transfer bitcoins to the following address (click to copy):
198tX7NmLg6o8qcTT2Uv9cSBVzN3oEozpv
After the payment is finalized enter Transfer ID below.

Amount:      Transfer ID:
BTC 0.396    |
   Internet connection is unavailable. Click Network Connections and connect to the Internet
If the fine is not paid, a warrant will be issued for your arrest,
which will be forwarded to your local authorities. You will be charged, fined, convicted for up to 5 years.
```

Figure 4.6: A machine infected by Virlock ransomware used in the experiments.

Figure 4.6 shows a version of Virlock used in the experiments. The Windows' screen will look like the picture after infection. Virlock is a sophisticated mainstream polymorphic ransomware that turns every infected file into a version of its own kind. Therefore, every infected file becomes a Virlock itself. Any infected file sent by the user or applications to others will result in the infection of the target machine [3].

4.5 Concluding Remarks

This chapter discussed the first step in the reduction of logs volume of our two-step volume reduction process. The configuration presented in this chapter helps to make our logging process more efficient. It makes the data smaller but at the same time more valuable. The chapter covers the details of our configuration with examples of how to add or remove rules for different parts and why each rule is used. We also discussed the logging process and environment setup for dataset creation.

Additionally, the chapter describes the details of our datasets including the five primary datasets as well as the 10 secondary ones along with the reason as to why and how each of them was created. The malware used to create the datasets were carefully chosen and described in the chapter.

Chapter 5

Implementation and Experiments

In this chapter, we introduce a new log-based anomaly detection framework. The framework is proposed to utilize host-based logs for anomaly detection in Windows environments. Moreover, we discuss the details of our Sysmon parser to parse Sysmon logs. We also give a detailed representation of various parts of our anomaly detection system, its architecture, implementation, and deployment along with the analysis process.

5.1 Proposed Framework for Log-based Anomaly Detection

Our proposed framework is designed to take advantage of host-based log data generated by Sysmon to track and hunt malicious activities performed in a Windows environment. The framework also helps accelerate the anomaly detection process for administrators and analysts by the amount of data they need to inspect in order to find potential deviations from the normal operation of a Windows machine.

There are three main steps to capture anomalies: Preparation, detection, and analysis. Our framework will address all three steps thoroughly to ensure the comprehensiveness of the framework. Figure 5.1 shows an overview of the framework.

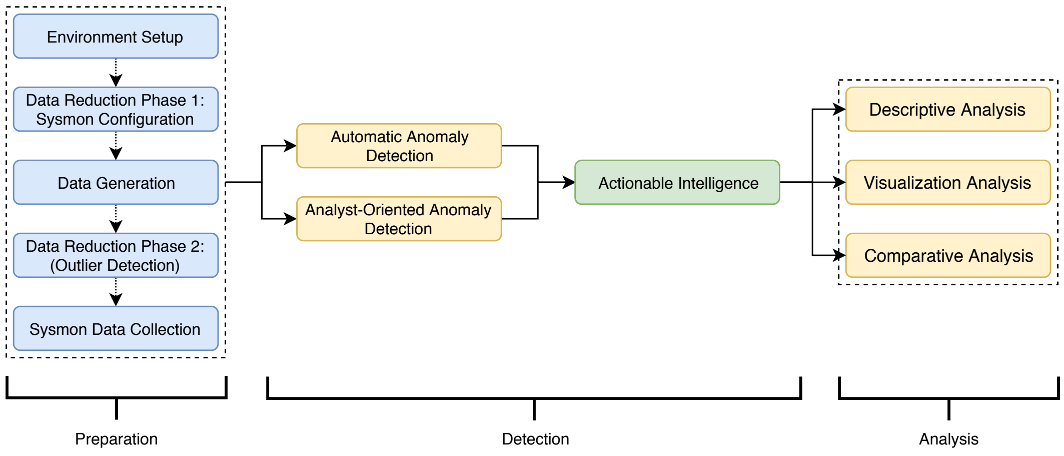


Figure 5.1: An overview of the proposed framework for anomaly detection.

- **Step 1 (Preparation):** The first step of our framework is to prepare the data for the anomaly detection system. In this step, we first prepare a Windows machine with some of the most commonly used application installed on it. This step ensures that we get as close to a real-world scenario as possible. We then perform the first data reduction step by configuring Sysmon to log the necessary data and filter out noisy and redundant events. Next up is data generation, where we generate the data by letting the System run from several hours to several days, depending on the dataset. Another data reduction step in then performed by using clustering techniques to detect outliers from the data. At the end, we collect, centralize, and pass the final data over to the anomaly detection system.

- **Step 2 (Detection):** In this step, we first analyze the data automatically using our VirusTotal malware detection system. This data along with what is left from the previous step is analyzed by a human analyst to find any potential remaining malicious activity. This step ensures that the analyst can analyze the important data in a reasonably short period of time to find anomalies. The analysis of anomalies is performed in the next step.

- **Step 3 (Analysis):** The last step of our framework is dedicated to the analysis of the results from the previous steps. In this step, we analyze the results using descriptive, visualization, and comparative analysis.

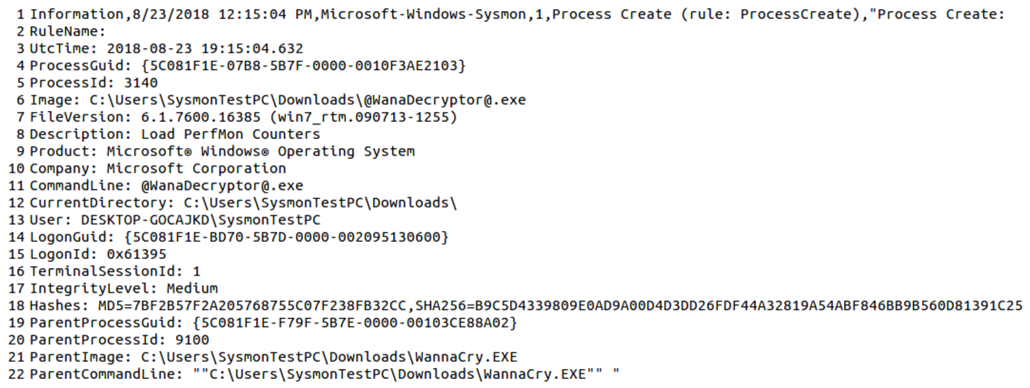
5.2 Parsing Sysmon Logs

One of the contributions of our work is to implement a flexible Sysmon parsing tool to manage, process, and extract features from the logs. This is, to the best of our knowledge, the first independent publicly available tool for parsing and processing Sysmon-generated log data. There is no publicly available configurable parser for Sysmon other than Microsoft's own Log Parser 2.2, which is a general-purpose parsing tool for different types of text-based Windows logs and data. Since Log Parser is almost 14 years old and was not designed to support Sysmon logs, its output is not anything close to what it should be. Therefore, an independent parser was implemented to parse the logs.

The proposed parser is a highly configurable and powerful tool written in Python that collects, parses, and saves the output as a CSV file that is ready for analysis. It can parse more than 50,000 events in less than a fraction of a second. It is a very versatile and flexible tool that can be configured for different types of analysis.

There are many fields and features that the user can specify and select from to configure the parser for what they need. The features can be chosen from 'UTC Time and Date', 'Sequence Number, 'Event's Rule', 'Executable file's Hash (SHA-256 or MD5 or both), 'File Version', 'Description', 'Product', 'Company', 'User', 'Current Directory', 'Process ID', 'Process GUID', 'Parent Process ID', 'Parent Process GUID', 'Logon ID', 'Logon GUID', 'Image', 'Parent Image', 'Terminal Session ID', 'Integrity Level', and 'Process Command Line', 'Parent Process Command Line'. Not all events have every field. Therefore, in some cases, the value of 'Null' is returned if they do not exist. Limiting the number of events, filtering the events by type or the code, and exporting the output in vertical or horizontal shapes are some of this parser's other abilities.

Figure 5.2: Raw log messages of a WannaCry-related event (top) and structured logs after parsing (bottom). Events are stored sequentially without any particular separator. The separation of the events is done by the parser. The fields are randomly selected, but in various analysis scenarios fields can be chosen based on the application.



```
Information,8/23/2018 12:15:04 PM,Microsoft-Windows-Sysmon,1,Process Create (rule: ProcessCreate),"Process Create:
RuleName:
UtcTime: 2018-08-23 19:15:04.632
ProcessGuid: {5C081F1E-07B8-5B7F-0000-0010F3AE2103}
ProcessId: 3140
Image: C:\Users\SysmonTestPC\Downloads\@WanaDecryptor@.exe
FileVersion: 6.1.7600.16385 (win7_rtm.090713-1255)
Description: Load PerfMon Counters
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
CommandLine: @WanaDecryptor@.exe
CurrentDirectory: C:\Users\SysmonTestPC\Downloads\
User: DESKTOP-GOCAJKD\SysmonTestPC
LogonGuid: {5C081F1E-BD70-5B7D-0000-002095130600}
LogonId: 0x61395
TerminalSessionId: 1
IntegrityLevel: Medium
Hashes: MD5=7BF2B57F2A205768755C07F238FB32CC,SHA256=B9C5D4339809E0AD9A00D4D3DD26FDF44A32819A54ABF846BB9B560D81391C25
ParentProcessGuid: {5C081F1E-F79F-5B7E-0000-00103CE88A02}
ParentProcessId: 9100
ParentImage: C:\Users\SysmonTestPC\Downloads\WannaCry.EXE
ParentCommandLine: ""C:\Users\SysmonTestPC\Downloads\WannaCry.EXE"" "
```

$$\Downarrow$$

Event ID	1
Event Type	`Process Create`
Time	`19:15:04.632`
Date	`2018-08-23`
MD5 Hash	`7BF2B57F2A205768755C07F238FB32CC`
Process GUID	`5C081F1E-07B8-5B7F-0000-0010F3AE2103`
Parent Process GUID	`5C081F1E-F79F-5B7E-0000-00103CE88A02`
Logon ID	`0x61395`
Logon GUID	`5C081F1E-BD70-5B7D-0000-002095130600`
Image	`C:\Users\SysmonTestPC\Downloads\@WanaDecryptor@.exe`
Command Line	`@WanaDecryptor@.exe`
Integrity Level	Medium
Parent Proess ID	9100
Parent Image	`C:\Users\SysmonTestPC\Downloads\WannaCry.EXE`
Parent Command Line	`C:\Users\SysmonTestPC\Downloads\WannaCry.EXE`

5.3 Proposed Approach

The last contribution of this thesis is a new anomaly detection system. While there are different anomaly detection software and tools available, many of them are too complicated to use and require many different sources of data to detect malicious activities. In this study, the focus has been strictly on host-based Windows logs without taking advantage of information in the network and firewall logs.

Our anomaly detection system is designed to capture different types of threats in a Windows environment using host-based logs. The architecture of the system is shown in Figure 5.3. The results show that many types of malicious activities are easily detectable solely using host-based logs generated by the system. This is interesting in several ways. One is that it reveals the potential of the host-based logs and the amount of information that can be extracted from them without relying on network and firewall logs. Another reason is that it provides useful information that traps many malicious activities before they destroy or cause damage to the files. And finally, the comprehensibility of the output will help the analyst detect and analyze malicious events that have happened before, which is very difficult when doing so on raw log messages.

5.4 System Architecture

The system includes several components that can cooperate to detect malware or malicious activities. The output of the system is the security analysis of the workstation logs. First off, activities of different users are distinguishable based on the logon ID, so any action performed by a user will be logged with their own identity. This way, if malicious activity is performed on a multi-user workstation, the user responsible for that action can easily be identified. Furthermore, the logs produced by the Windows machines are gathered and stored in three different formats: EVTX, XML, and CSV. This adds more flexibility if the logs are taken to other tools and software for further investigation.

In this work, only the CSV files are used. The XML and EVTX versions are stored but are not used. Next, the logs are passed to the Parsing and Feature Extraction Engine to extract useful features for analysis. The extracted information is then stored in the database to prevent reprocessing in the future. Ultimately, the features are passed to the Outlier Detection Engine, the VirusTotal Analyzer, and the Human Analyst respectively to extract knowledge and interesting activities from them.

5.5 The Analysis Process

In this section, the analysis steps are briefly introduced and explained. When logs from a properly configured workstation are exported, they are sent to the parser for the parsing process. Based on its configuration, the parser generates the output as a CSV file. The following is a description of how the output file is thoroughly analyzed in three steps.

Step 1: Outlier Detection Process: Due to VirusTotals free API limitations, and also to further ease processing the events, we decided to perform the second step of volume reduction. Therefore, the events that are more likely to be interesting are identified. To achieve this, seven attributes, namely 'Time', 'Date', 'Event Type', 'MD5 Hash', 'Process GUID', 'Parent Process GUID', and 'Logon GUID', are selected. The attributes are selected on an experimental basis mostly based on their availability, correlation, and robustness. The outliers are detected using k-means algorithm implemented

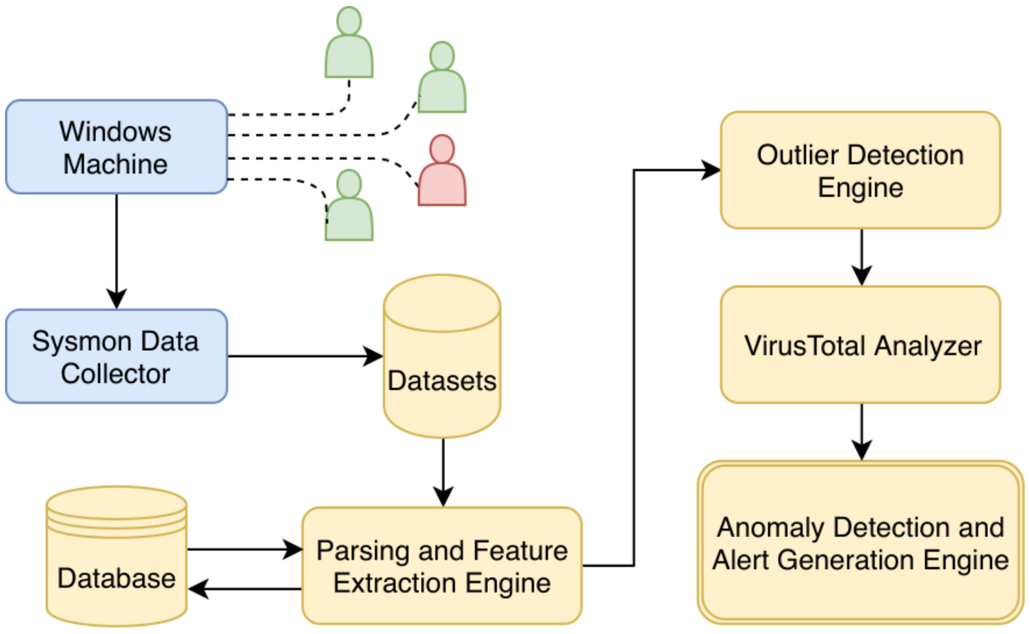


Figure 5.3: Architecture of the anomaly detection system

by Weka [18], which is an open-source knowledge analysis and data mining software.

5.5.1 k-means Clustering Algorithm

k-means is a simple, unsupervised machine learning algorithm and is one of the most widely used algorithms in clustering analysis and data mining. The algorithm is used to partition a given set of observations into a predefined amount of k clusters. As explained by [1, 26], the algorithm first creates a set of randomly generated k center points (μ).
After that, with each step of running the algorithm, any observation x will be mapped to its closest center point (see 5.1). There can only be one observation assigned to a center point. So if there are multiple center points with the same distance to a specific observation, one will randomly be assigned to those observations.

$$S_i^{(t)} = \left\{ x_p : \left\| x_p - \mu_i^{(t)} \right\|^2 \leq \left\| x_p - \mu_j^{(t)} \right\|^2 \ \forall j, 1 \leq j \leq k \right\} \tag{5.1}$$

Thenceforth, the position of each center point will be recalculated according to the mean of each of the observations assigned to their corresponding center point.

$$\mu_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j \tag{5.2}$$

In general, in each iteration, the k-means algorithm optimizes the objective function 5.3. Since the number of feasible assignments is limited, and with each step the algorithm generates better clusters, it will always finish when trapped in a local minimum.

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2 \tag{5.3}$$

$$\text{with } r_{nk} = \begin{cases} 1 & x_n \in S_k \\ 0 & \text{otherwise} \end{cases}$$

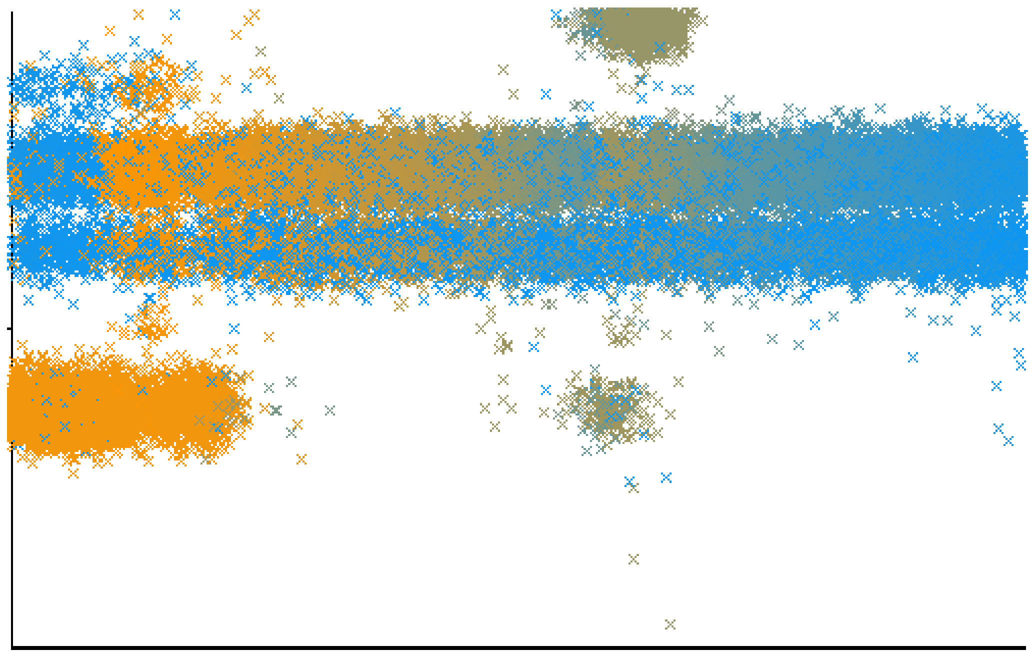


Figure 5.4: Visualization of clusters found by k-means based on 'Rule' and 'Time' attributes.

Dividing the data into three clusters causes some highly noisy outliers to stand out. Based on our observations, we can simply ignore the larger cluster, which mostly contains highly repeated events that are the results of normal actions by the operating system and the user. The remaining is several hundreds of events that have been detected as outliers. These events are then sent to the VirusTotal Analyzer for analysis.

Step 2: VirusTotal Analyzer: VirusTotal has one of the largest malware databases in the world. It is also equipped with over 70 antivirus products and other malware scanning engines. One can submit several types of information online on the website to get the analysis results of the submitted file or data. It gives a thorough examination of the data to the user. The request can be a file, URL, IP address, domain, or a hash. Using the API, a user can automatically get the analysis result of the submitted data in their application.

The analyzer uses the Sysmon parser to extract MD5 hash of previously detected outliers and then stores them in a set container to prevent duplicates. The result set of hashes is then automatically sent to VirusTotal for analysis. Once ready, the results are sent back to the user as a JSON file for each hash. A simple parser was implemented to parse the output JSON results, capture the required information, and report them as the output. The outputs are then classified in the following four classes:

- **Benign:** This is the expected output for most of the hashes. A hash is considered clean when the output of VirusTotal indicates that zero engines have detected the hash as malicious.

- **Malicious:** This happens if the VirusTotal's output is a non-zero number, which is usually higher than 20. This means that, for example, more than 20 engines have confirmed that the hash belongs to a malicious file.

- **Suspicious:** In some rare cases, the output for some hashes is a small non-zero number, normally between 1 to 5. While most engines consider the file to be benign, it is flagged it as a suspicious file that needs further investigation.

- **Unknown:** The last case is where the returned result of VirusTotal shows the executable file has not been uploaded and thus has not been analyzed by anyone yet. This type of output is unusual because many new or unknown malwares are in this category. Another possible scenario is that the file is benign but so new or so unknown that nobody has tried uploading it yet. Either way, this type of output should be treated very carefully since it can be dangerous. In the case of new malware, it will probably get caught by the analyst at its early stages of activities.

Figure 5.5 shows the analysis process using VirusTotals API as well as the thresholds used in the flagging process. When an event is flagged as unknown or suspicious by the VirusTotal analyzer, it is sent to a human analyst for further investigation.

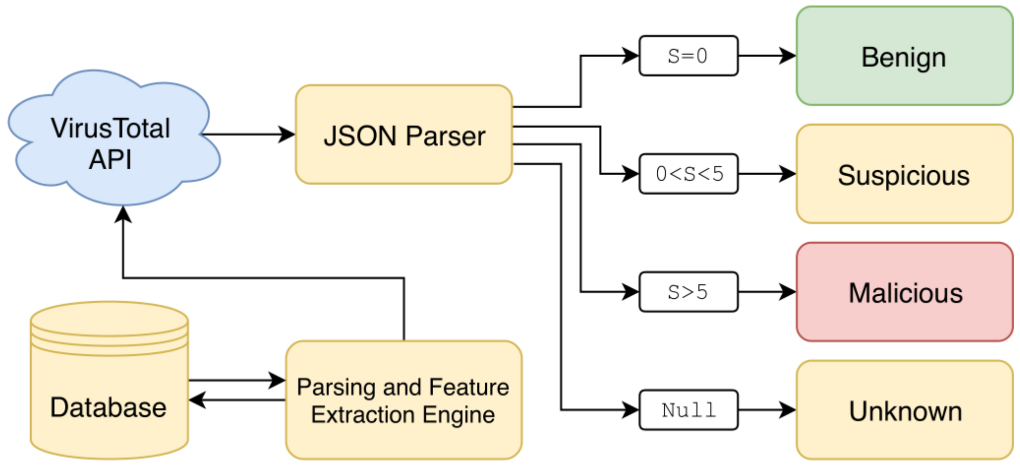


Figure 5.5: Architecture of VirusTotal analyzer

Step 3: Analysis Most malicious activities and attackers' attempts are automatically traced and detected at the previous stages. However, there are some events that are considered suspicious or unknown which require more analysis. In this step, a human analyst analyzes suspicious activities. Hereupon, since the events are normally limited to a dozen, therefore, a human analyst can do the analysis easier, faster, and more accurately. Table 6.1 shows that on average, less than 0.01% of events need to be analyzed by the human analyst.

5.6 Concluding Remarks

In this chapter, we introduced a host-based anomaly detection framework that is capable of capturing malicious activities automatically. It also can help analysts to speed up the anomaly detection and analysis process. Moreover, we introduced our Sysmon parser, which is to the best of our knowledge, the only publicly available parser for Sysmon. We used our parser to parse and extract meaningful features from the logs and utilized them in our anomaly detection system. Moreover, we presented the details of our anomaly detection system including the system's architecture and its analysis process. The details of our k-means outlier detection algorithm to perform the second step of the log volume of our reduction process are also explained. We proposed the VirusTotal analyzer, the automatic malware detection system as a part of our anomaly detection system. The analyzer is able to extract and look for details of every executable on the machine in real-time, and bring back the malware analysis results from the VirusTotal's database. The events are flagged based on the returned results and are then sent to an analyst for further analysis.

Chapter 6

Results

We analyze and conclude our findings from three different perspectives. The first approach is to describe the results and perform analysis using descriptive analysis. The second approach is to visualize the clusters and outliers and observe the results visually to perform analysis. The last perspective is comparative analysis where we compare our anomaly detection system against state-of-the-art security information and event management software.

6.1 Descriptive Analysis

Out of the five primary datasets, three were infected, and two were benign. In the first case, the dataset was infected by a Trojan called CoinMiner. The second one was infected by WannaCry and the last one by an adware called Kuaizip. In all the three malware datasets, the malicious files were all

Table 6.1: Analysis results of main datasets (number of events per category).

(Dataset)	Total	Analyzed	Malicious	Suspicious	Unknown
CoinMiner	33793	86	1	2	5
Kuaizip	12728	117	5	2	3
WannaCry	45156	107	6	1	12
Benign AWS	34697	69	0	3	3
Benign VBox	30024	124	0	2	9

detected at the VirusTotal stage less than five seconds after analysis started. The results as shown in Table 6.1, demonstrate that in two cases, malware was detected in one or more file creation processes before they ran. This shows that an effective logging procedure for file creation and file modification processes can help easily detect known malware instantly right after they are downloaded and before they are even run on the system. In all three cases, there is a significant time gap between the malware download time and the time when the infected file was run on the system, which can help the analyst to take necessary actions before the malware causes damage to the system. However, even if they are not detected at the download stage, all of them are detected as soon as their executables run on the system. This especially helps with malware like CoinMiner, as they are completely hidden while running. In the two benign cases, 17 suspicious and unknown files turned out to be either false positives triggered by some antivirus engines or some other benign software such as new device drivers.

Table 6.1 shows the massive reduction in volume while preserving the important events. In all cases, every component of the malware was detected. The CoinMiner malware was a single executable, which was successfully detected while running. In the WannaCry and Kuaizip cases, malware and all its components, including the ones that were installed later by the malware, were successfully detected both at download and running stages.

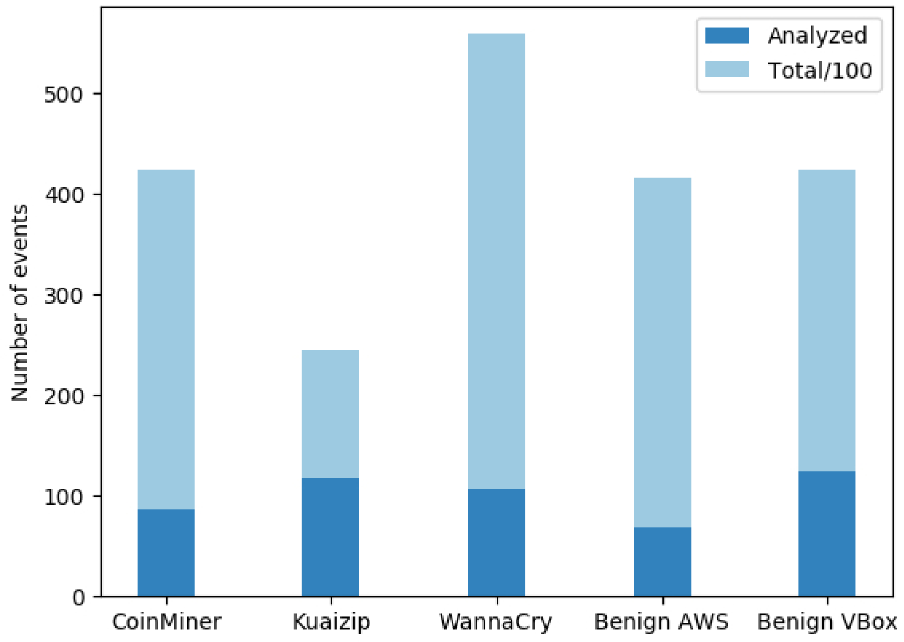


Figure 6.1: Ratio of the number of total and analyzed events

Another important point to derive from the results is that despite the noticeable difference in the size of the datasets, the number of remaining events is relatively similar and close to each other. This shows that the detection engine selects certain events that are known to be more important, and the number of these events normally remains consistent in different datasets. Figure 6.1 shows the number of raw events divided by 100 compared to the ones that are picked for analysis by the anomaly detection engine. Looking closer, we can see that despite the remarkable difference between Kuaizip and WannaCry raw events, the number of final events is close and even slightly higher for Kuaizip. The average number of selected events is around 100, which is a reasonably small number for a human analyst to perform the necessary and immediate investigation on the data to find the potential threats.

In terms of speed and efficiency, there are three main stages that should be considered to evaluate the performance of the proposed method. The first stage is the preprocessing and parsing of the logs. Even the largest datasets were run in a fraction of a second. Despite being so fast, since the parser is written in Python, the performance can even further improve if implemented in some other languages such as C or C++.

The second stage is analysis by the VirusTotal analyzer. This is the bottleneck of the system and is limited by the limitations of the free API. However, when the premium and unlimited API is used, and since the number of analyzed events is so limited, it can be done very quickly and even in real-time. The last part is the clustering part which again, runs in under a second. Overall, due to heavy preprocessing on the data, the performance is fast mostly because the remaining data is always much smaller in size. Moreover, with minor optimization and improvements, the system can run in real-time on very large scales.

The secondary set of data were mostly intended to be used for the implementation of the actual anomaly detection engine and also for differential analysis of the logs. This makes distinguishing the malware logs from the malicious logs much easier by comparing different datasets to each other and to a baseline. Consequently, we compared the data to a benign baseline, separated new logs from the remaining logs, and flagged them as malicious activity.

Table 6.2: Analysis results of secondary datasets (number of events per category).

Malware	Total Events	Unique Executables	Malicious Detected
Baseline (benign)	826	38	0
Ramnit	12728	46	0
Agent	1501	46	0
Virut	1066	47	1
Virlock	2771	52	3
Allaple	1483	46	0
VB	1249	50	1
Sivis	1463	47	1
Upatre	1106	47	0
Injector	1648	47	1
Kryptik	1157	45	0

The results are shown in Table 6.2. In most cases, the malware-related logs are perfectly separated. However, some malware, like the HTML-based variant of Ramnit, which is used to create one of the datasets, mostly depend on Windows services. Therefore, since the suggested configuration does not monitor all services' activities, Ramnit bypasses the logging process and it is thus never detected by the system before it is completely activated. Nonetheless, in a more realistic environment, it will be detected before causing harm to any component of the system, merely because all executable, important services such as Local Security Authority Subsystem Service (LSASS) and file modifications are monitored.
Even though the substantial differences in number of events is the result of activities by the specific malware run on the machine, in half of the datasets, the malware remains completely undetected by the VirusTotal analyzer. This further proves that signature-based malware detection methods are limited when it comes to detection capabilities. Furthermore, since the volume of the data is low, and the malware was only run for a certain and limited amount of time, there is not enough data to properly feed and train the main anomaly detection algorithm.

6.2 Visualization Analysis

Visualization is one of the easiest ways to find and analyze outliers in anomaly detection. Once the clusters are found by the algorithm, the analyst can find the potential anomalies by using visualization techniques. Apart from the outlier detection process used in the anomaly detection system, we have also performed the k-means clustering algorithm on some of our datasets, such as Wannacry. In this example, there are three main clusters found by the algorithm which are visualized in three different colors: orange, blue, and dark green. In Figure 6.2, the clusters are created based on the attributes 'Rule' and 'Time & Date'. These two attributes mainly represent the type of logs generated at any given time.

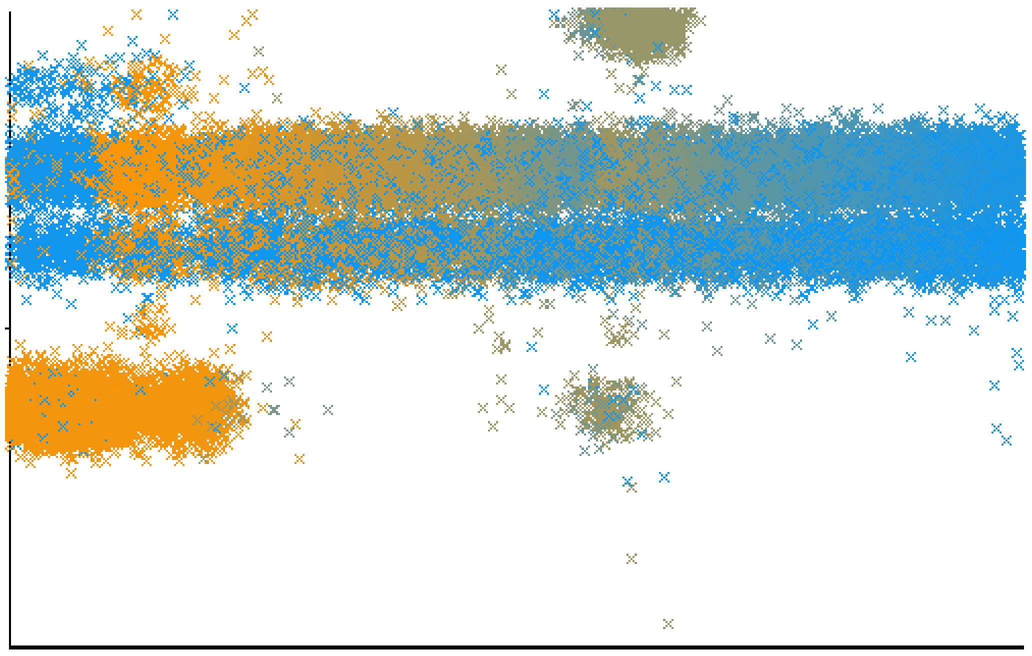


Figure 6.2: Visualization of clusters found by k-means based on 'Rule' and 'Time & Date' attributes in Wannacry dataset.

Based on our observations, the cluster showing normal activity is mostly shown in blue, which includes the events that are consistently generated over time with no major changes. While this represents normal data, the other two smaller clusters stand out as outliers. Our findings show that the distribution of the green cluster is due to changes in the registry and drivers of Windows with the larger part being during a Windows update. As this is not something that happens on a regular basis, it does look like an outlier. The third cluster is displayed in orange. Despite existing many 'File Create' and 'Process Create' events throughout the whole dataset, at some point in time, the number of events grows at a much higher pace. This process starts when the machine is infected by the malware. The denser orange cluster is an indicator of the high number of 'File Create' events over a short period of time. The more sparse orange part is the executables run by Wannacry to run the encryption algorithm. This results in a high number of 'Process Create' events being generated.

Using a different form of analysis, we cluster the logs based on 'Rule' and 'MD5 Hash'. The reason for this choice of attributes is that the 'Rule' attribute plays a more important role than any other attribute that we can derive from the logs. Also, 'MD5 Hash' is a unique value for each unique file and executable. In our experiments, these two important attributes can help find better clusters than most other combinations. Figure 6.3 shows the three clusters found by the algorithm. This time, the algorithm particularly shows the distribution of unique files and executables with respect to their corresponding event rule. This means that, for instance, we have a better idea of where there is an unusual number of unique files and executables with the same type of rule. This can help identify most malware that are dependent on generating or manipulating a large number of files.

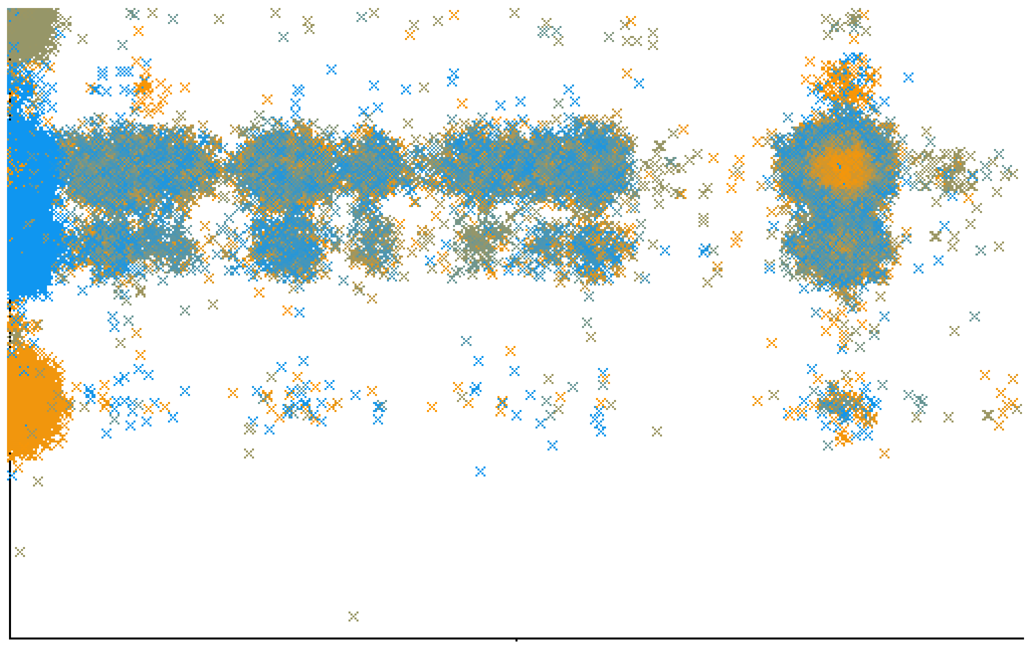


Figure 6.3: Visualization of clusters found by k-means based on 'Rule' and 'MD5 Hash' attributes in Wannacry dataset.

6.3 Comparative Analysis

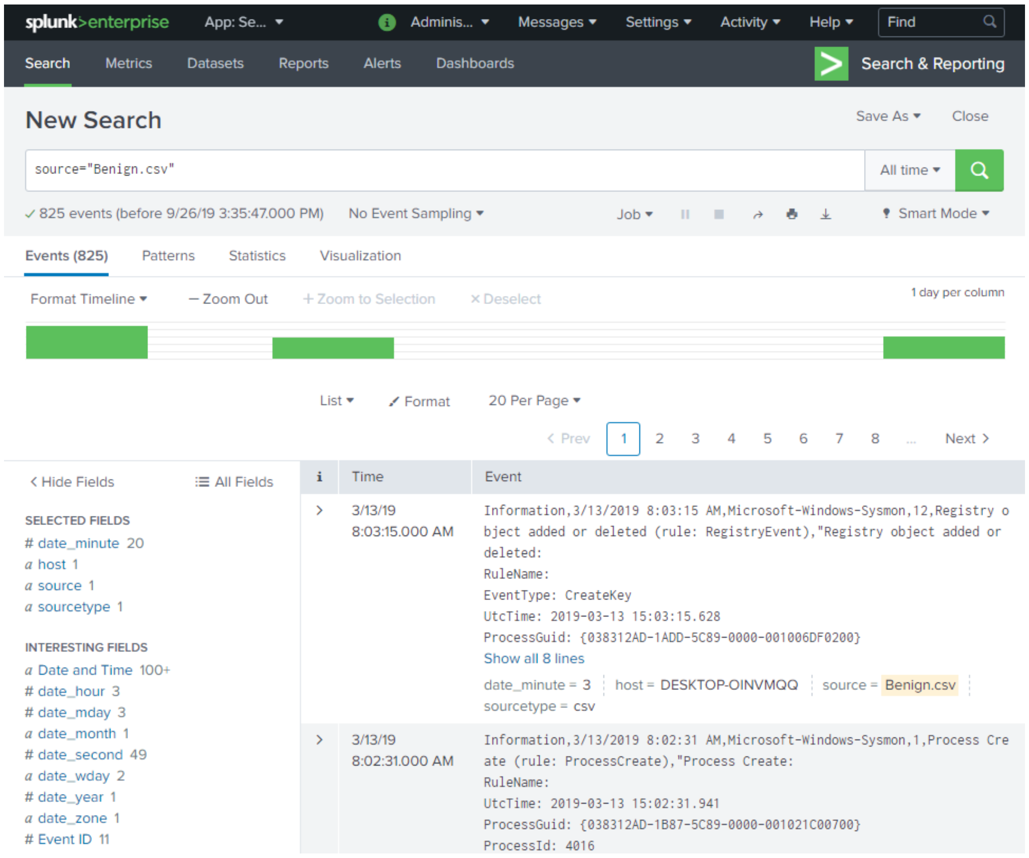
Sysmon is a relatively new tool and is fully dependent on its configuration. There are a few works and datasets available to which we can compare our method. The nature of the tool means it can be configured differently in various environments. Based on the application, the output log can vary greatly, which makes it difficult to compare different datasets to each other.

As for general log analysis systems, not many works strictly focus on host-based logs and even then, they use vastly different sources of logs and data to do the analysis. Therefore, having a useful comparative analysis in this new domain is still impractical.

Considering that there are not many options with which to compare our experiments, we decided to use one of the best and most commonly used security information and event management software (SIEM) applications to compare our findings against what the log analysis software has to offer.

In order to get the most accurate results, we compare our baseline benign data from the secondary dataset to the one infected by Virlock ransomware. This is to make sure that both datasets have been generated in a completely equal situation for the same period of time with the only difference between them being the one single infected file run on one of the machines.

The first and most basic observation of data can be seen in Figures 6.4 and 6.5, where the number of events over the same period of time is substantially higher in the infected dataset than the one with no malware on it. This is mostly caused by the number of excessive executables run on the system. It is also caused by file operations mainly the encrypted files created by the ransomware and the original files that have been deleted afterward.



New Search

```
source="Benign.csv"
```

✓ 825 events (before 9/26/19 3:35:47.000 PM) No Event Sampling

SELECTED FIELDS
- # date_minute 20
- a host 1
- a source 1
- a sourcetype 1

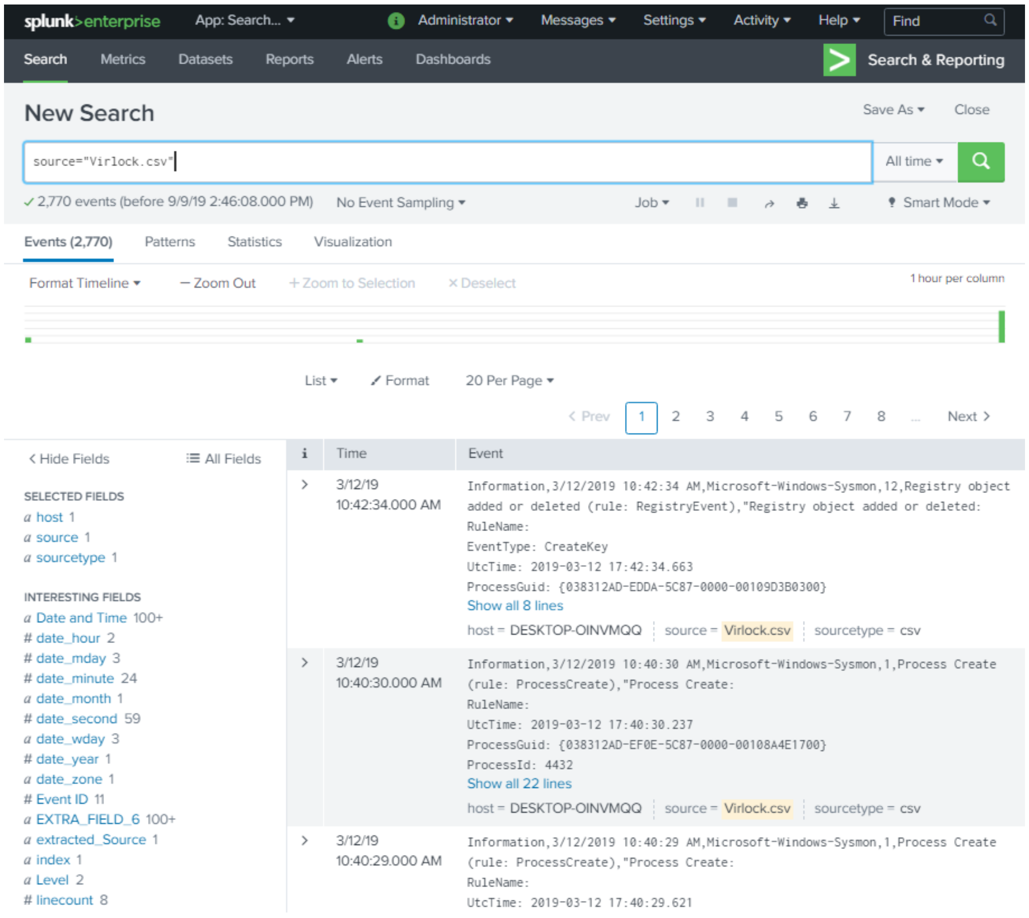
INTERESTING FIELDS
- a Date and Time 100+
- # date_hour 3
- # date_mday 3
- a date_month 1
- # date_second 49
- a date_wday 2
- # date_year 1
- a date_zone 1
- # Event ID 11

Time	Event

3/13/19 8:03:15.000 AM	`Information,3/13/2019 8:03:15 AM,Microsoft-Windows-Sysmon,12,Registry object added or deleted (rule: RegistryEvent),"Registry object added or deleted:`
	`RuleName:`
	`EventType: CreateKey`
	`UtcTime: 2019-03-13 15:03:15.628`
	`ProcessGuid: {038312AD-1ADD-5C89-0000-001006DF0200}`
	Show all 8 lines
	date_minute = 3 host = DESKTOP-OINVMQQ source = Benign.csv sourcetype = csv

3/13/19 8:02:31.000 AM	`Information,3/13/2019 8:02:31 AM,Microsoft-Windows-Sysmon,1,Process Create (rule: ProcessCreate),"Process Create:`
	`RuleName:`
	`UtcTime: 2019-03-13 15:02:31.941`
	`ProcessGuid: {038312AD-1B87-5C89-0000-001021C00700}`
	`ProcessId: 4016`

Figure 6.4: 'Search & Reporting' by Splunk on the benign dataset.



```
source="Virlock.csv"
```

✓ 2,770 events (before 9/9/19 2:46:08.000 PM)

SELECTED FIELDS
a host 1
a source 1
a sourcetype 1

INTERESTING FIELDS
a Date and Time 100+
# date_hour 2
# date_mday 3
# date_minute 24
a date_month 1
# date_second 59
a date_wday 3
# date_year 1
a date_zone 1
Event ID 11
a EXTRA_FIELD_6 100+
a extracted_Source 1
a index 1
a Level 2
# linecount 8

i	Time	Event
>	3/12/19 10:42:34.000 AM	Information,3/12/2019 10:42:34 AM,Microsoft-Windows-Sysmon,12,Registry object added or deleted (rule: RegistryEvent),"Registry object added or deleted: RuleName: EventType: CreateKey UtcTime: 2019-03-12 17:42:34.663 ProcessGuid: {038312AD-EDDA-5C87-0000-00109D3B0300} Show all 8 lines host = DESKTOP-OINVMQQ ¦ source = Virlock.csv ¦ sourcetype = csv
>	3/12/19 10:40:30.000 AM	Information,3/12/2019 10:40:30 AM,Microsoft-Windows-Sysmon,1,Process Create (rule: ProcessCreate),"Process Create: RuleName: UtcTime: 2019-03-12 17:40:30.237 ProcessGuid: {038312AD-EF0E-5C87-0000-00108A4E1700} ProcessId: 4432 Show all 22 lines host = DESKTOP-OINVMQQ ¦ source = Virlock.csv ¦ sourcetype = csv
>	3/12/19 10:40:29.000 AM	Information,3/12/2019 10:40:29 AM,Microsoft-Windows-Sysmon,1,Process Create (rule: ProcessCreate),"Process Create: RuleName: UtcTime: 2019-03-12 17:40:29.621

Figure 6.5: 'Search & Reporting' by Splunk on the dataset infected by Virlock ransomware.

The second experiment is to determine if Splunk can find any interesting patterns that can distinguish the normal dataset from the one infected by the ransomware. As shown in Figures 6.6 and 6.7, we are looking at the top five most interesting patterns detected by Splunk in our benign dataset. In a normal Windows environment, events of 'Process Access' type are the most frequent ones. The two largest clusters include more than 80% of all the events. While the clusters can have overlapping partitions, based on Splunk's analysis, 80.48% (66.3% + 14.18% found in two separate clusters) of all events are in 'Process Access' related clusters. This implicates that the overall behavior of the machine, based on statistic analysis, is normal.

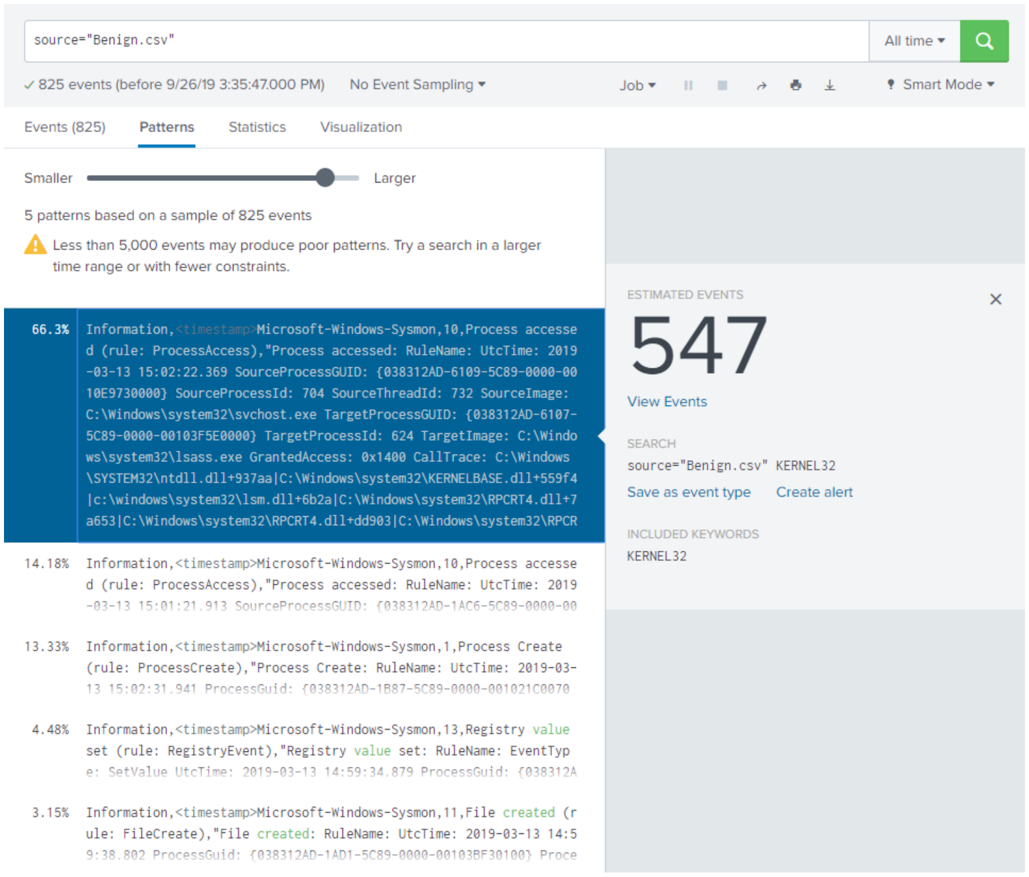
On the other hand, 35.35% (29.27% + 6.08% found in two separate clusters) of all the events on the infected machine are found to be in 'Process Access' related clusters. While this is a different statistic from the exact number of 'Process Access' events, it gives us an insight of differences between the baseline logs and the logs from the infected machine. Table 6.3 shows the details of the clusters found by Splunk in each dataset.

Table 6.3: Distribution of clusters' types found in the two datasets.

Dataset	Process Access Clusters	Process Create Clusters	File Create Clusters	Registry Clusters
Benign	**80.48% (66.3 + 14.18)**	13.33%	3.15%	4.48
Virlock	35.35% (29.27 + 6.08)	**32.96% (27.31 + 5.65)**	**0%**	9.98% (7.87 + 2.11)

Moreover, the number of 'Process Create' related clusters are also substantially different (32.96% vs 13.33%). As there should not be a large number of newly created processes on a normal machine, the difference can give the analyst some clues as to whether or not something suspicious is happening on the machine.

One other rather interesting finding is that Splunk finds no clusters around 'File Create' event type which was previously found by our k-means clustering. In addition to that, some of the clusters found by Splunk are basically the number of events in that specific task category. Two examples of this include 'Process Create' and 'File Create' events and clusters in the benign dataset which are exactly the same size. This shows that it is very difficult to have a general-purpose log clustering module that works perfectly fine with various types of log data. It also further proves that analysts need to rely on different types of analysis to fully capture malicious activities.



```
source="Benign.csv"
```

✓ 825 events (before 9/26/19 3:35:47.000 PM) No Event Sampling

5 patterns based on a sample of 825 events

Less than 5,000 events may produce poor patterns. Try a search in a larger time range or with fewer constraints.

```
66.3%   Information,<timestamp>Microsoft-Windows-Sysmon,10,Process accesse
        d (rule: ProcessAccess),"Process accessed: RuleName: UtcTime: 2019
        -03-13 15:02:22.369 SourceProcessGUID: {038312AD-6109-5C89-0000-00
        10E9730000} SourceProcessId: 704 SourceThreadId: 732 SourceImage:
        C:\Windows\system32\svchost.exe TargetProcessGUID: {038312AD-6107-
        5C89-0000-00103F5E0000} TargetProcessId: 624 TargetImage: C:\Windo
        ws\system32\lsass.exe GrantedAccess: 0x1400 CallTrace: C:\Windows
        \SYSTEM32\ntdll.dll+937aa|C:\Windows\system32\KERNELBASE.dll+559f4
        |c:\windows\system32\lsm.dll+6b2a|C:\Windows\system32\RPCRT4.dll+7
        a653|C:\Windows\system32\RPCRT4.dll+dd903|C:\Windows\system32\RPCR

14.18%  Information,<timestamp>Microsoft-Windows-Sysmon,10,Process accesse
        d (rule: ProcessAccess),"Process accessed: RuleName: UtcTime: 2019
        -03-13 15:01:21.913 SourceProcessGUID: {038312AD-1AC6-5C89-0000-00

13.33%  Information,<timestamp>Microsoft-Windows-Sysmon,1,Process Create
        (rule: ProcessCreate),"Process Create: RuleName: UtcTime: 2019-03-
        13 15:02:31.941 ProcessGuid: {038312AD-1B87-5C89-0000-001021C0070

 4.48%  Information,<timestamp>Microsoft-Windows-Sysmon,13,Registry value
        set (rule: RegistryEvent),"Registry value set: RuleName: EventTyp
        e: SetValue UtcTime: 2019-03-13 14:59:34.879 ProcessGuid: {038312A

 3.15%  Information,<timestamp>Microsoft-Windows-Sysmon,11,File created (r
        ule: FileCreate),"File created: RuleName: UtcTime: 2019-03-13 14:5
        9:38.802 ProcessGuid: {038312AD-1AD1-5C89-0000-00103BF30100} Proce
```

ESTIMATED EVENTS

547

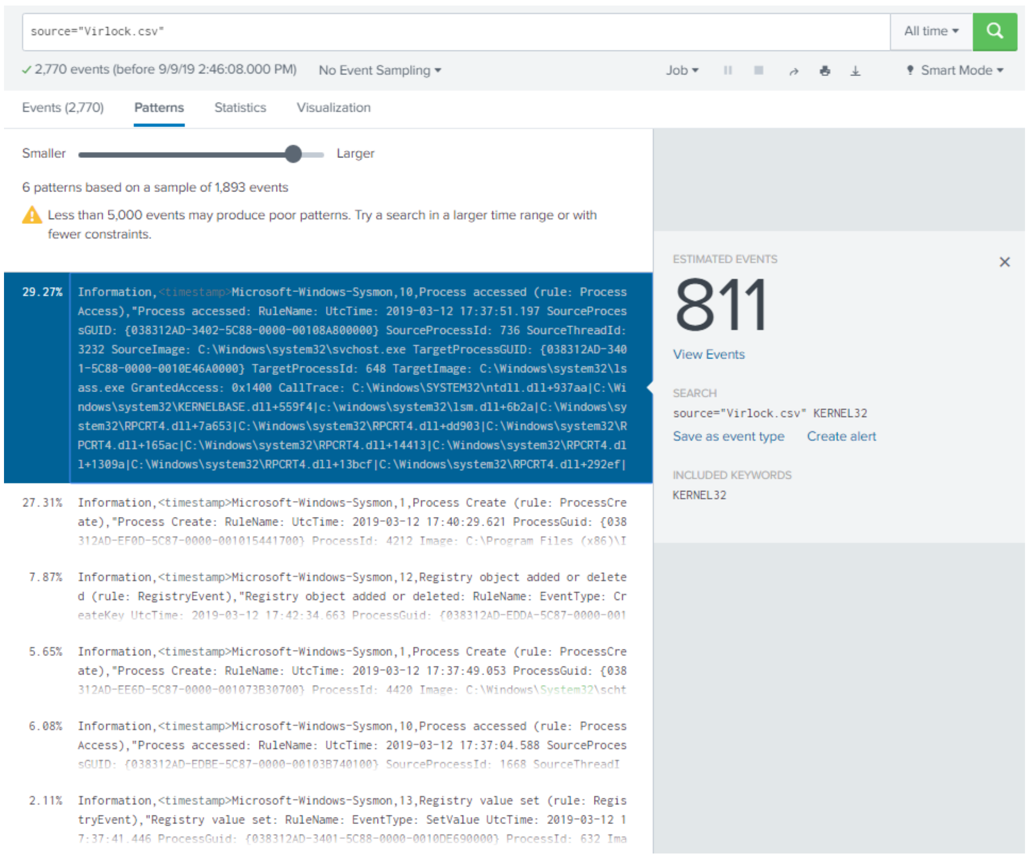
SEARCH

```
source="Benign.csv" KERNEL32
```

INCLUDED KEYWORDS

KERNEL32

Figure 6.6: Interesting patterns found by Splunk in the benign dataset.



```
source="Virlock.csv"
```

✓ 2,770 events (before 9/9/19 2:46:08.000 PM)   No Event Sampling

6 patterns based on a sample of 1,893 events

Less than 5,000 events may produce poor patterns. Try a search in a larger time range or with fewer constraints.

```
29.27%  Information,<timestamp>Microsoft-Windows-Sysmon,10,Process accessed (rule: Process
        Access),"Process accessed: RuleName: UtcTime: 2019-03-12 17:37:51.197 SourceProces
        sGUID: {038312AD-3402-5C88-0000-00108A800000} SourceProcessId: 736 SourceThreadId:
        3232 SourceImage: C:\Windows\system32\svchost.exe TargetProcessGUID: {038312AD-340
        1-5C88-0000-0010E46A0000} TargetProcessId: 648 TargetImage: C:\Windows\system32\ls
        ass.exe GrantedAccess: 0x1400 CallTrace: C:\Windows\SYSTEM32\ntdll.dll+937aa|C:\Wi
        ndows\system32\KERNELBASE.dll+559f4|c:\windows\system32\lsm.dll+6b2a|C:\Windows\sy
        stem32\RPCRT4.dll+7a653|C:\Windows\system32\RPCRT4.dll+dd903|C:\Windows\system32\R
        PCRT4.dll+165ac|C:\Windows\system32\RPCRT4.dll+14413|C:\Windows\system32\RPCRT4.dl
        l+1309a|C:\Windows\system32\RPCRT4.dll+13bcf|C:\Windows\system32\RPCRT4.dll+292ef|

27.31%  Information,<timestamp>Microsoft-Windows-Sysmon,1,Process Create (rule: ProcessCre
        ate),"Process Create: RuleName: UtcTime: 2019-03-12 17:40:29.621 ProcessGuid: {038
        312AD-EF0D-5C87-0000-001015441700} ProcessId: 4212 Image: C:\Program Files (x86)\I

 7.87%  Information,<timestamp>Microsoft-Windows-Sysmon,12,Registry object added or delete
        d (rule: RegistryEvent),"Registry object added or deleted: RuleName: EventType: Cr
        eateKey UtcTime: 2019-03-12 17:42:34.663 ProcessGuid: {038312AD-EDDA-5C87-0000-001

 5.65%  Information,<timestamp>Microsoft-Windows-Sysmon,1,Process Create (rule: ProcessCre
        ate),"Process Create: RuleName: UtcTime: 2019-03-12 17:37:49.053 ProcessGuid: {038
        312AD-EE6D-5C87-0000-001073B30700} ProcessId: 4420 Image: C:\Windows\System32\scht

 6.08%  Information,<timestamp>Microsoft-Windows-Sysmon,10,Process accessed (rule: Process
        Access),"Process accessed: RuleName: UtcTime: 2019-03-12 17:37:04.588 SourceProces
        sGUID: {038312AD-EDBE-5C87-0000-00103B740100} SourceProcessId: 1668 SourceThreadI

 2.11%  Information,<timestamp>Microsoft-Windows-Sysmon,13,Registry value set (rule: Regis
        tryEvent),"Registry value set: RuleName: EventType: SetValue UtcTime: 2019-03-12 1
        7:37:41.446 ProcessGuid: {038312AD-3401-5C88-0000-0010DE690000} ProcessId: 632 Ima
```

ESTIMATED EVENTS

811

SEARCH

```
source="Virlock.csv" KERNEL32
```

INCLUDED KEYWORDS

KERNEL32

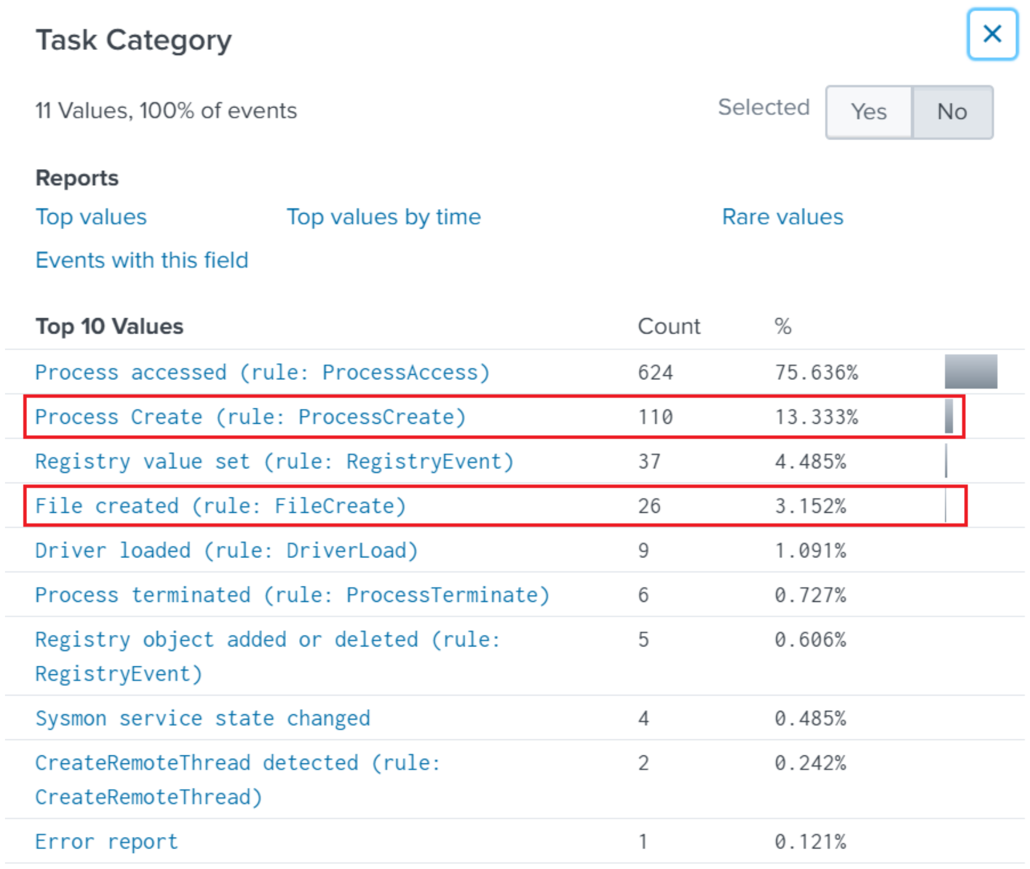
Figure 6.7: Interesting patterns found by Splunk in the logs collected from the machine infected by Virlock ransomware.

Another statistical-driven information is the percentage of each task category. In a normal environment, based on our specific configuration, most events should be of the 'Process Access' category. This is indeed the case for our baseline clean logs, 75.636% of which are of this specific category. On the other hand, not only the infected data includes substantially less 'Process Access' events, but also it is not even the largest task category. Interestingly enough, 'Process Create' events secure the top spot for themselves. This only means that at this period of time the machine was actually acting suspiciously unless some unlikely activity, such as simultaneously reinstalling several programs, was performed on it. Table 6.4 shows the percentage of events in the top three task categories for both datasets.

Table 6.4: Distribution of logs volume in the top three largest task categories.

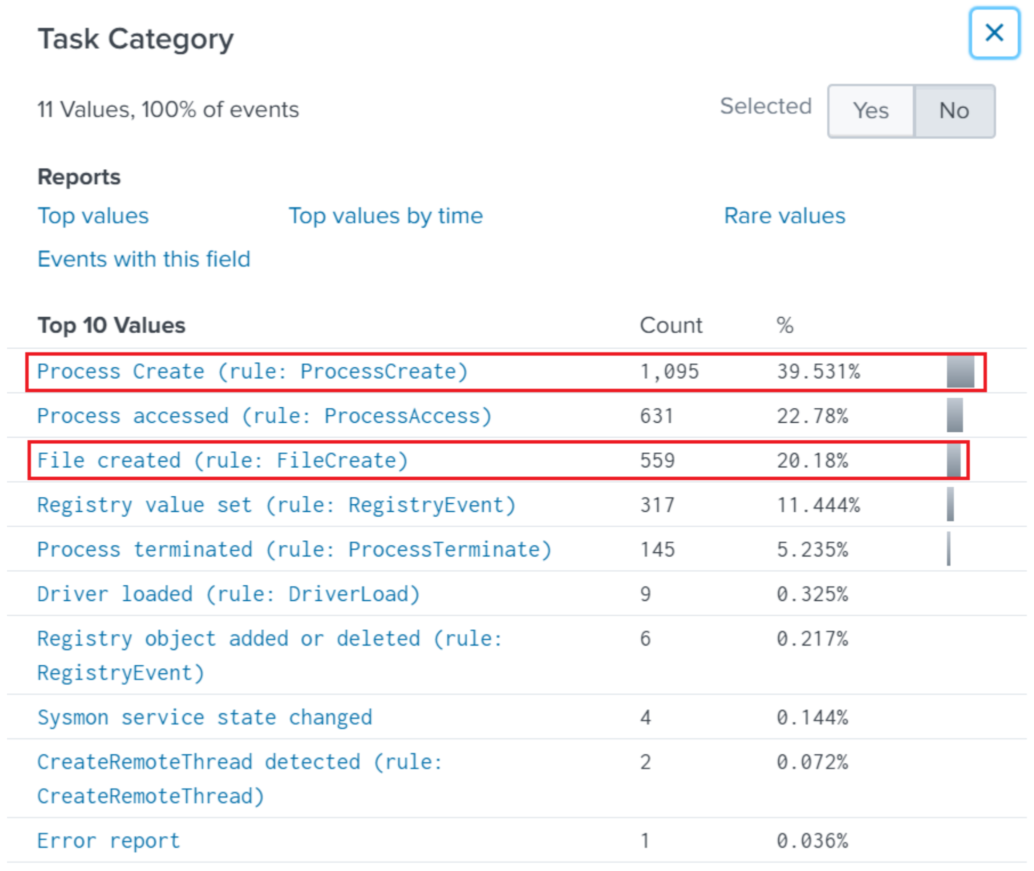
Dataset	Total Events	Process Access	Process Create	File Create
Benign	825	**75.636%**	13.333%	3.152%
Virlock	2770	22.78%	**39.531%**	**20.18%**

Additionally, as shown in Figure 6.9, 'File Create' events include a shocking 20.18% of all the events. This strange activity further proves that some kind of malicious activity has been performed on the machine. In this case, the ransomware has been busy creating encrypted files on the victim's machine, which is why more than 20% of task categories are of this kind.



Top 10 Values	Count	%
Process accessed (rule: ProcessAccess)	624	75.636%
Process Create (rule: ProcessCreate)	110	13.333%
Registry value set (rule: RegistryEvent)	37	4.485%
File created (rule: FileCreate)	26	3.152%
Driver loaded (rule: DriverLoad)	9	1.091%
Process terminated (rule: ProcessTerminate)	6	0.727%
Registry object added or deleted (rule: RegistryEvent)	5	0.606%
Sysmon service state changed	4	0.485%
CreateRemoteThread detected (rule: CreateRemoteThread)	2	0.242%
Error report	1	0.121%

Figure 6.8: Splunk's report on the top 10 values found in the logs in the benign dataset.


Task Category

11 Values, 100% of events

Reports

Top 10 Values	Count	%
Process Create (rule: ProcessCreate)	1,095	39.531%
Process accessed (rule: ProcessAccess)	631	22.78%
File created (rule: FileCreate)	559	20.18%
Registry value set (rule: RegistryEvent)	317	11.444%
Process terminated (rule: ProcessTerminate)	145	5.235%
Driver loaded (rule: DriverLoad)	9	0.325%
Registry object added or deleted (rule: RegistryEvent)	6	0.217%
Sysmon service state changed	4	0.144%
CreateRemoteThread detected (rule: CreateRemoteThread)	2	0.072%
Error report	1	0.036%

Figure 6.9: Splunk's report on the top 10 values found in the logs in the malicious dataset infected by Virlock ransomware.

6.4 Concluding Remarks

In this chapter, we evaluated our findings from the previous detection stages using descriptive analysis. We also performed two additional types of analysis to ensure that we thoroughly analyze the results from different aspects. Our descriptive analysis results show the effectiveness of our anomaly detection system. We found and captured every malicious activity performed on our primary datasets. It is found that in a real-world scenario, after a certain period of time, every known malware would be detected normally before its activation. Also, unknown malware and activities are carefully selected for further investigation. Our secondary datasets show that not all malware are detectable using signature-based malware detection methods. Therefore, having an all-inclusive anomaly detection system is essential.

Moreover, we further analyze the results by performing two more different types of analysis, namely visualization analysis, and comparative analysis. As a quick way to analyze the outliers, we used visualization techniques to visualize the data and find interesting patterns inside them. The results show the effectiveness of visualization analysis for anomaly detection.

Lastly, we conclude our findings by comparing them to the anomalies found by Splunk, a very sophisticated log analysis and security information and event management software. We performed three different comparisons: Overview of Splunk's 'Search & Reporting', clustering analysis, and task category comparison. As expected, Splunk was able to find and detect most anomalies on our select datasets. However, there is still no general way to analyze all types of log data. While being very useful, Splunk falls short to detect some specific activities that were previously detected by our anomaly detection system. Therefore, we believe that such software applications still heavily rely on developers and analysts to design and implement input-specific add-ons for them to be able to show their full potential.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

Host-based logs are a rich source of information that can help to detect intrusions and anomalies and to understand attackers' behavior and tools. These logs can be used to trace and hunt malicious activities as well as for forensics and incident response. The logs, however, can become so large that sometimes need several extensive preprocessing and size reduction steps before we can use them for analysis. The retention of useful information is one of the most important factors in the preprocessing stage.

Conclusively, we believe while log analysis software are very helpful in regards to manage and analyze the logs, having a tool that can safely shrink the data is very helpful for a security analyst to speed up the detection and response process. Also, no matter how powerful a software and tool is, if the data fed into it is not properly generated, its capabilities will be limited.

This thesis discussed the use of Windows-generated logs data, mainly Sysmon logs, to trace and hunt attackers on Windows hosts. It also covered how to prepare the data for investigation and incident response. A log-based anomaly detection approach was proposed, and the use of different tools and related essential software and solutions were presented.

We first discussed the motivations along with the reasons as to why host-based Sysmon logs are a rich source of information for anomaly detection in Windows environments. We then designed and implemented a comprehensive highly configurable Sysmon parser for parsing and extracting features from the logs. A hierarchical approach was proposed that considerably reduced the volume of Windows generated Sysmon logs. We proved that we can extensively shrink the log data while preserving actionable intelligence. We created 15 different Sysmon events datasets with no modification or anonymization to achieve the closest experiments to a real-world scenario. The next step was to design and implement an anomaly detection engine that could successfully detect attacks and malware on different datasets. A comprehensive descriptive, as well as inferential statistical analysis of the results, was presented followed by a comparative analysis of the proposed method against state-of-the-art SIEM software. The results showed the effectiveness of our approach for threat hunting in a Windows environment.

7.2 Future Work

Future trends include the online analysis of big log data, which will be helpful to profile and capture different behaviors on the host. It can also be used to detect anomalies and attackers and to improve the security of the system.

As one of the most common destinations of machine-generated log data, creating add-ons for SIEMs for specific types of logs can help improve the detection and monitoring process. In future work, we can port our system to a modern SIEM through add-ons. The ability to control SIEM's detection process brings such benefits as to use our extracted features and our parser to improve the software's capabilities. This is a great way to contribute to these software applications and ultimately improve their abilities. Also, using additional SIEMs and log analysis software such as Logrhythm [6] can give us a better glimpse into the shortcomings of our work and the future directions of log analysis and such software applications.

Additionally, the possible privacy issues that come with the gathering of a large amount of log data from users' workstations, and how to conform to the related laws like GDPR, can be discussed. Furthermore, the work presented here could be extended to other popular operating systems like GNU/Linux, Mac OS X, or even mobile operating systems like Android.

Bibliography

[1] *Github - bluec0re/thesis-latex*, https://github.com/bluec0re/Thesis-Latex, 2015, (Accessed on 10/10/2019).

[2] *Final_scf123678_kentfarries_transalta_effectivelyenhancingsoc - marcom reviewed*, https://conf.splunk.com/files/2017/slides/effectively-enhancing-our-soc-with-sysmon-powershell-logging-and-machine-lear pdf, 2017, (Accessed on 10/07/2019).

[3] *Virlocker's comeback; including recovery instructions - malwarebytes labs — malwarebytes labs*, https://blog.malwarebytes.com/threat-analysis/2017/01/virlockers-comeback-including-recovery-instructions/, 2017, (Accessed on 07/26/2019).

[4] *Av-test_security_report_2017-2018.pdf*, https://www.av-test.org/fileadmin/pdf/security_report/AV-TEST_Security_Report_2017-2018.pdf, 2018, (Accessed on 07/25/2019).

[5] *Analyzing logs for security information event management*, `https://www.manageengine.com/products/eventlog/Analyzing-Logs-for-SIEM-Whitepaper.html`, 2019, (Accessed on 07/22/2019).

[6] *Logrhythm, the security intelligence company — logrhythm*, `https://logrhythm.com/`, 2019, (Accessed on 11/25/2019).

[7] *Sysmon - windows sysinternals — microsoft docs*, `https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon`, 2019, (Accessed on 07/15/2019).

[8] *Sysmon - windows sysinternals — microsoft docs*, `https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon`, 2019, (Accessed on 10/17/2019).

[9] Robert G Abbott, Jonathan McClain, Benjamin Anderson, Kevin Nauer, Austin Silva, and Chris Forsythe, *Log analysis of cyber security training exercises*, Procedia Manufacturing **3** (2015), 5088–5094.

[10] Amruta Ambre and Narendra Shekokar, *Insider threat detection using log analysis and event correlation*, Procedia Computer Science **45** (2015), 436–445.

[11] Russ Anthony, *Detecting security incidents using windows workstation event logs*, SANS Institute, InfoSec Reading Room Paper (2013).

[12] Liang Bao, Qian Li, Peiyao Lu, Jie Lu, Tongxiao Ruan, and Ke Zhang, *Execution anomaly detection in large-scale systems through console log analysis*, Journal of Systems and Software **143** (2018), 172–186.

[13] Konstantin Berlin, David Slater, and Joshua Saxe, *Malicious behavior detection using windows audit logs*, Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security, ACM, 2015, pp. 35–44.

[14] Sarah Delasko and Weifeng Chen, *Operating systems of choice for professional hackers*, ICCWS 2018 13th International Conference on Cyber Warfare and Security, Academic Conferences and publishing limited, 2018, p. 159.

[15] Min Du and Feifei Li, *Spell: Online streaming parsing of large unstructured system logs*, IEEE Transactions on Knowledge and Data Engineering (2018).

[16] Qiang Fu, Jian-Guang Lou, Yi Wang, and Jiang Li, *Execution anomaly detection in distributed systems through unstructured log analysis*, Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on, IEEE, 2009, pp. 149–158.

[17] Gabriele Giuseppini and Mark Burnett, *Microsoft log parser toolkit: A complete toolkit for microsoft's undocumented log analysis tool*, Elsevier, 2005.

[18] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten, *The weka data mining software: an update*, ACM SIGKDD explorations newsletter **11** (2009), no. 1, 10–18.

[19] Pinjia He, Jieming Zhu, Shilin He, Jian Li, and Michael R Lyu, *An evaluation study on log parsing and its use in log mining*, 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), IEEE, 2016, pp. 654–661.

[20] Pinjia He, Jieming Zhu, Pengcheng Xu, Zibin Zheng, and Michael R Lyu, *A directed acyclic graph approach to online log parsing*, arXiv preprint arXiv:1806.04356 (2018).

[21] Pinjia He, Jieming Zhu, Zibin Zheng, and Michael R Lyu, *Drain: An online log parsing approach with fixed depth tree*, Web Services (ICWS), 2017 IEEE International Conference on, IEEE, 2017, pp. 33–40.

[22] Esa Heikkinen and Timo D Hämäläinen, *Logdig log file analyzer for mining expected behavior from log files.*, SPLST, 2015, pp. 266–280.

[23] Markus Horisberger, *Centeractive ag: When siem is too much*, `https://retrospective.centeractive.com/blog_retrospective_whenSIEM.html`, 2013, (Accessed on 10/18/2018).

[24] Duhoe Kim, Dongil Shin, Dongkyoo Shin, and Yong-Hyun Kim, *Attack detection application with attack tree for mobile system using log analysis*, Mobile Networks and Applications (2018), 1–9.

[25] Logz.io, *The complete guide to the elk stack*, `https://logz.io/learn/complete-guide-elk-stack/`, 2018, (Accessed on 01/16/2019).

[26] J MacQueen, *Some methods for classification and analysis of multivariate observations*, Proc. Fifth Berkeley Symp. Math. Stat. Probab. Vol. 1 Stat. (Berkeley, Calif.), University of California Press, 1967, pp. 281–297.

[27] Adetokunbo AO Makanju, A Nur Zincir-Heywood, and Evangelos E Milios, *Clustering event logs using iterative partitioning*, Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2009, pp. 1255–1264.

[28] Vasileios Mavroeidis and Audun Jøsang, *Data-driven threat hunting using sysmon*, (2018).

[29] Paul McNeil, Sachin Shetty, Divya Guntu, and Gauree Barve, *Scredent: Scalable real-time anomalies detection and notification of targeted malware in mobile devices*, Procedia Computer Science **83** (2016), 1219–1225.

[30] Savita Mohurle and Manisha Patil, *A brief study of wannacry threat: Ransomware attack 2017*, International Journal of Advanced Research in Computer Science **8** (2017), no. 5.

[31] Meiyappan Nagappan and Mladen A Vouk, *Abstracting log lines to log event types for mining software system logs*, Mining Software Repos-

itories (MSR), 2010 7th IEEE Working Conference on, IEEE, 2010, pp. 114–117.

[32] Adam Oliner, Archana Ganapathi, and Wei Xu, *Advances and challenges in log analysis*, Communications of the ACM **55** (2012), no. 2, 55–61.

[33] Alina Oprea, Zhou Li, Ting-Fang Yen, Sang H Chin, and Sumayah Alrwais, *Detection of early-stage enterprise infection by mining large-scale log data*, Dependable Systems and Networks (DSN), 2015 45th Annual IEEE/IFIP International Conference on, IEEE, 2015, pp. 45–56.

[34] Mark Russinovich, *How to go from responding to hunting with sysinternals sysmon*, RSA Conference 2017 (2017).

[35] Statista, *Global market share held by operating systems for desktop pcs*, `https://www.statista.com/statistics/218089/global-market-share-of-windows-7/`, 2015, (Accessed on 05/22/2019).

[36] SwiftOnSecurity, *Github - swiftonsecurity/sysmon-config*, 1 2018, (Accessed on 10/18/2018).

[37] Duygu Sinanc Terzi, Ramazan Terzi, and Seref Sagiroglu, *Big data analytics for network anomaly detection from netflow data*, 2017 Inter-

national Conference on Computer Science and Engineering (UBMK), IEEE, 2017, pp. 592–597.

[38] Risto Vaarandi, *A data clustering algorithm for mining patterns from event logs*, IP Operations & Management, 2003.(IPOM 2003). 3rd IEEE Workshop on, IEEE, 2003, pp. 119–126.

[39] Risto Vaarandi and Mauno Pihelgas, *Using security logs for collecting and reporting technical security metrics*, Military Communications Conference (MILCOM), 2014 IEEE, IEEE, 2014, pp. 294–299.

[40] ———, *Logcluster-a data clustering and pattern mining algorithm for event logs*, Network and Service Management (CNSM), 2015 11th International Conference on, IEEE, 2015, pp. 1–7.

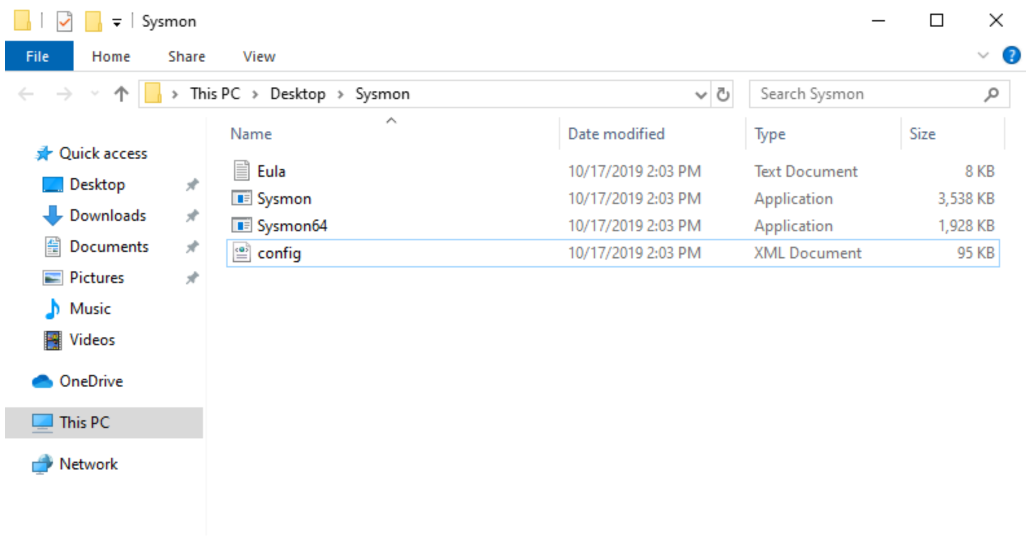
[41] Ting-Fang Yen, Alina Oprea, Kaan Onarlioglu, Todd Leetham, William Robertson, Ari Juels, and Engin Kirda, *Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks*, Proceedings of the 29th Annual Computer Security Applications Conference, ACM, 2013, pp. 199–208.

[42] Chen Zhuge and Risto Vaarandi, *Efficient event log mining with logclusterc*, Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS), 2017 IEEE 3rd International Conference on, IEEE, 2017, pp. 261–266.

Appendix A

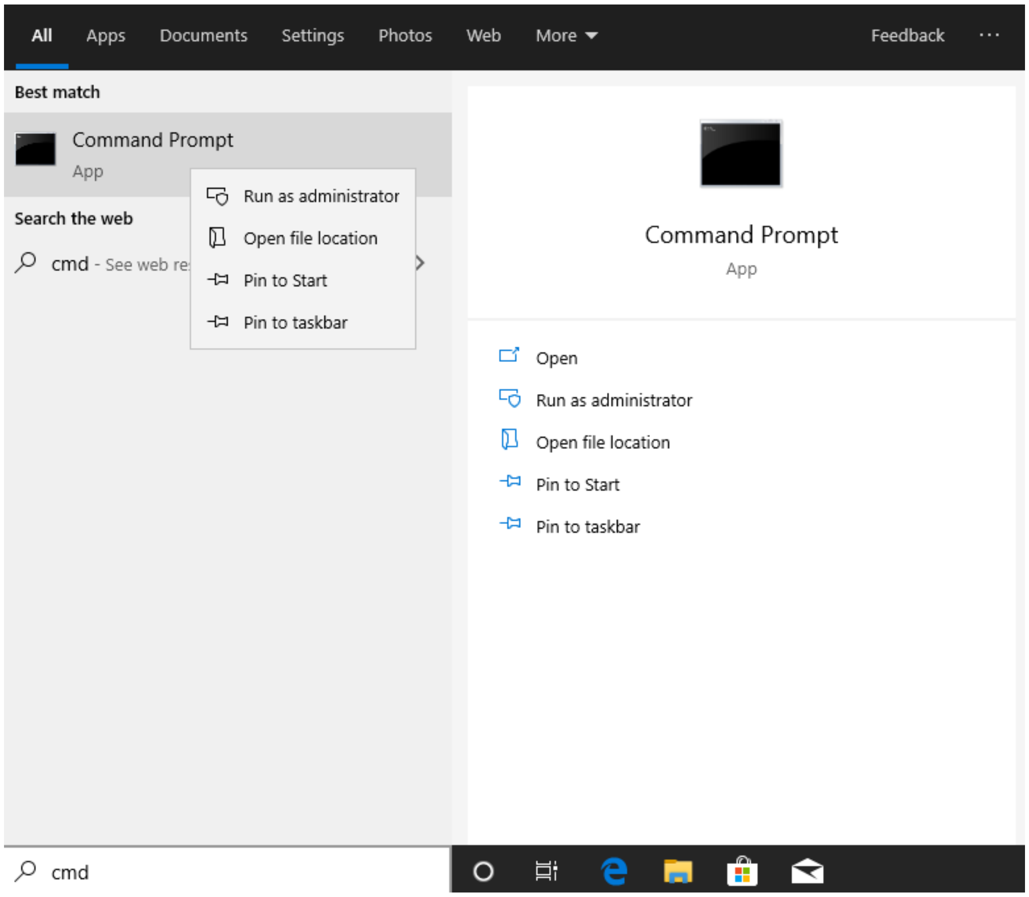
Sysmon Installation Guide

After downloading the installation file from the Microsoft website [8], we need to first extract it.

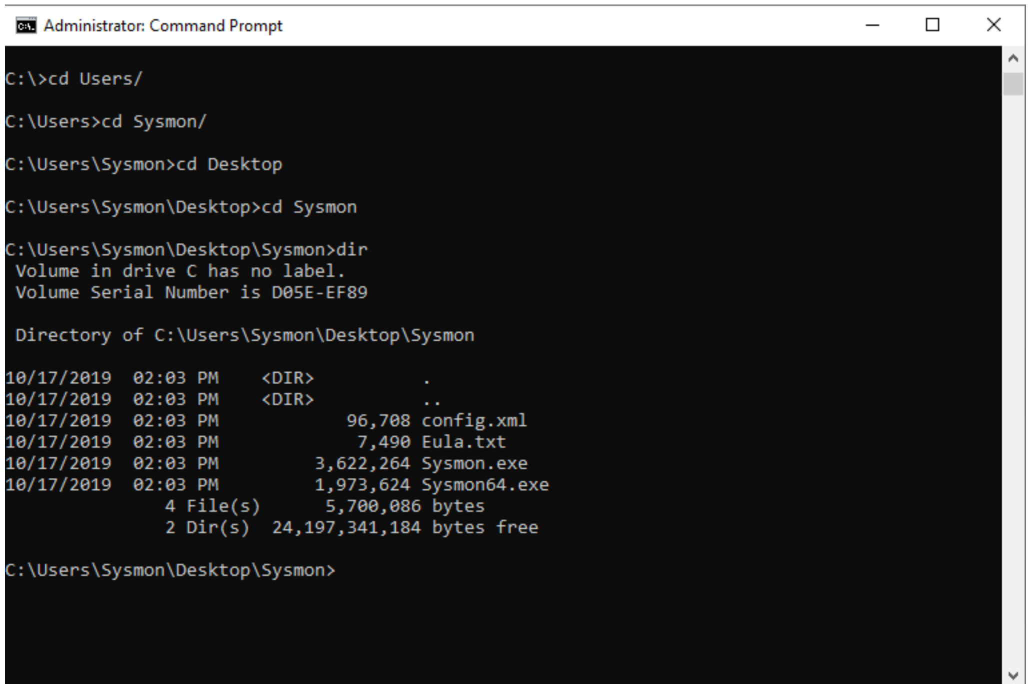


Name	Date modified	Type	Size
Eula	10/17/2019 2:03 PM	Text Document	8 KB
Sysmon	10/17/2019 2:03 PM	Application	3,538 KB
Sysmon64	10/17/2019 2:03 PM	Application	1,928 KB
config	10/17/2019 2:03 PM	XML Document	95 KB

The next step is to download and a setup a configuration file. While this step is optional, it is strongly recommended to have a properly written configuration file based on your need. Put the configuration file in the same folder as `Sysmon.exe`file and Eula. After that, open a command prompt as administrator.



Navigate to the folder where the Sysmon installation file and the configuration are located.

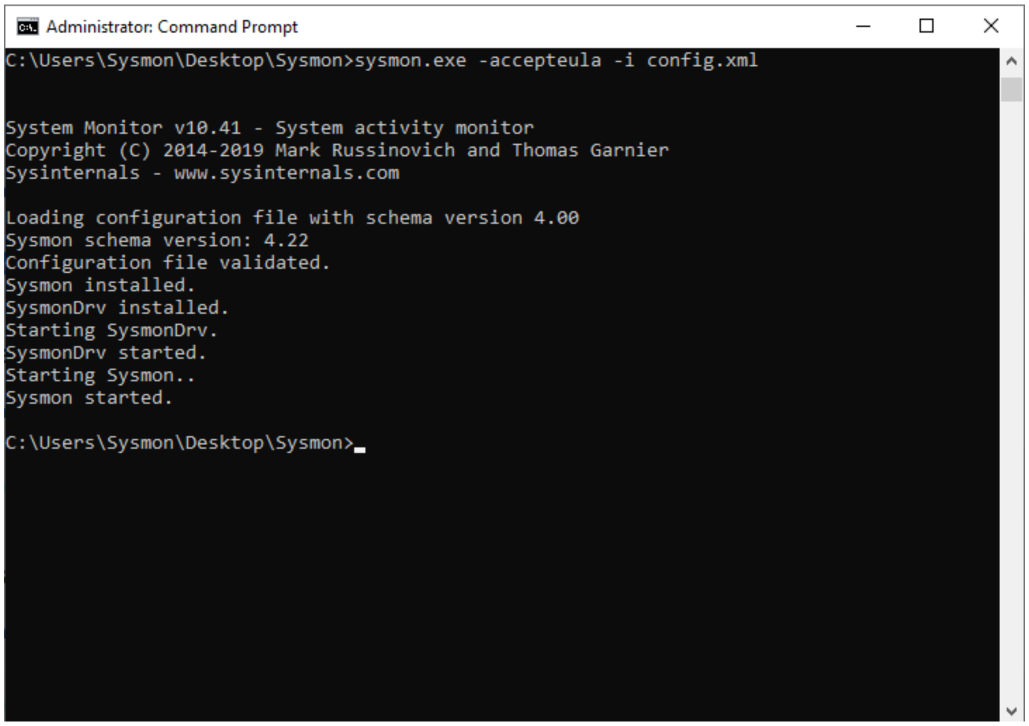

Administrator: Command Prompt

```
C:\>cd Users/

C:\Users>cd Sysmon/

C:\Users\Sysmon>cd Desktop

C:\Users\Sysmon\Desktop>cd Sysmon

C:\Users\Sysmon\Desktop\Sysmon>dir
 Volume in drive C has no label.
 Volume Serial Number is D05E-EF89

 Directory of C:\Users\Sysmon\Desktop\Sysmon

10/17/2019  02:03 PM    <DIR>          .
10/17/2019  02:03 PM    <DIR>          ..
10/17/2019  02:03 PM            96,708 config.xml
10/17/2019  02:03 PM             7,490 Eula.txt
10/17/2019  02:03 PM         3,622,264 Sysmon.exe
10/17/2019  02:03 PM         1,973,624 Sysmon64.exe
               4 File(s)      5,700,086 bytes
               2 Dir(s)  24,197,341,184 bytes free

C:\Users\Sysmon\Desktop\Sysmon>
```

We can now start the installation process by executing the following command.

```
sysmon.exe -accepteula -i config.xml
```

Replace the name of the files with the appropriate names if necessary.

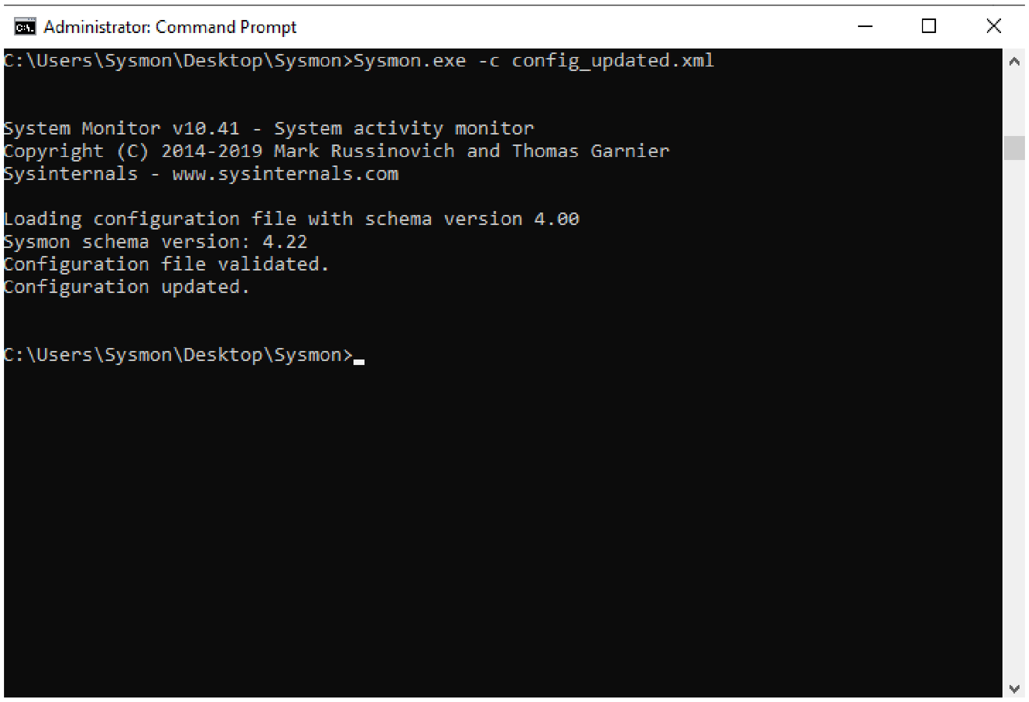


Administrator: Command Prompt

```
C:\Users\Sysmon\Desktop\Sysmon>sysmon.exe -accepteula -i config.xml


System Monitor v10.41 - System activity monitor
Copyright (C) 2014-2019 Mark Russinovich and Thomas Garnier
Sysinternals - www.sysinternals.com

Loading configuration file with schema version 4.00
Sysmon schema version: 4.22
Configuration file validated.
Sysmon installed.
SysmonDrv installed.
Starting SysmonDrv.
SysmonDrv started.
Starting Sysmon..
Sysmon started.

C:\Users\Sysmon\Desktop\Sysmon>
```

To update the configuration file, execute the following command followed by the configuration file name.

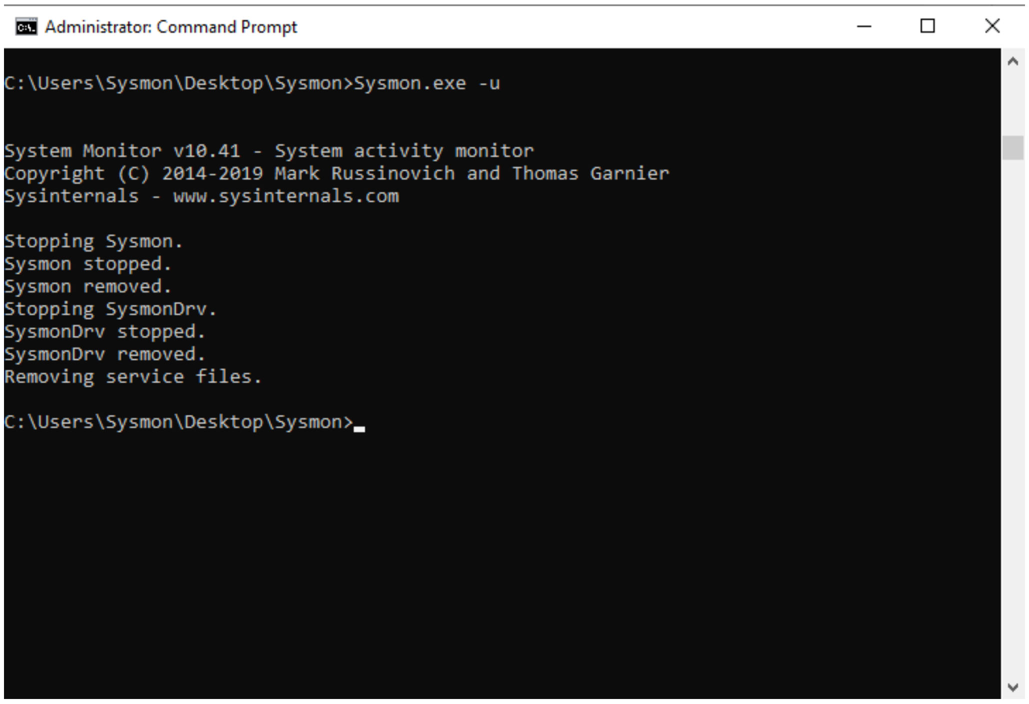
```
sysmon.exe -c config.xml
```



```
C:\Users\Sysmon\Desktop\Sysmon>Sysmon.exe -c config_updated.xml


System Monitor v10.41 - System activity monitor
Copyright (C) 2014-2019 Mark Russinovich and Thomas Garnier
Sysinternals - www.sysinternals.com

Loading configuration file with schema version 4.00
Sysmon schema version: 4.22
Configuration file validated.
Configuration updated.


C:\Users\Sysmon\Desktop\Sysmon>
```

To uninstall Sysmon, you can use the following command.

```
sysmon.exe -u
```

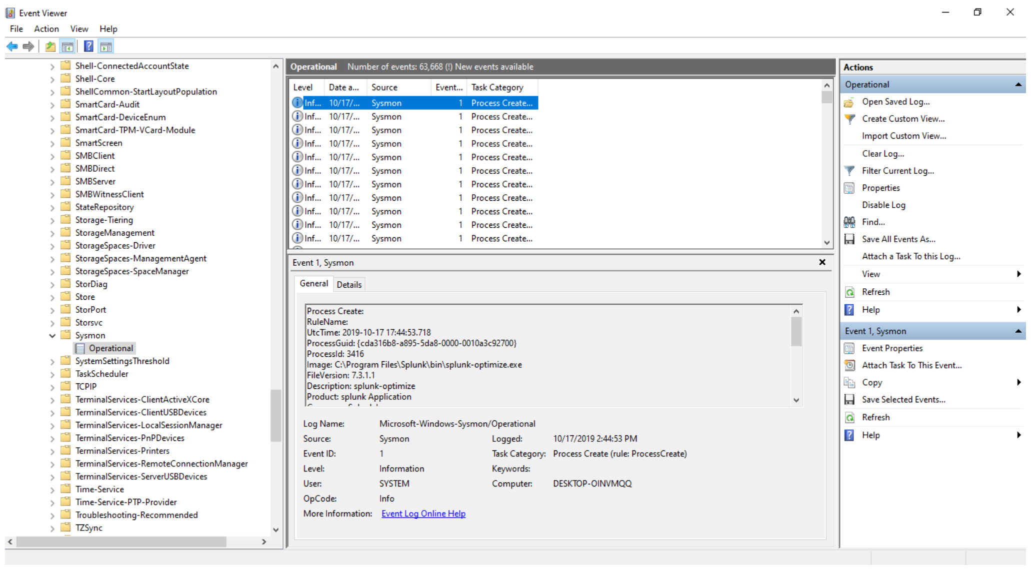


```
C:\Users\Sysmon\Desktop\Sysmon>Sysmon.exe -u


System Monitor v10.41 - System activity monitor
Copyright (C) 2014-2019 Mark Russinovich and Thomas Garnier
Sysinternals - www.sysinternals.com

Stopping Sysmon.
Sysmon stopped.
Sysmon removed.
Stopping SysmonDrv.
SysmonDrv stopped.
SysmonDrv removed.
Removing service files.

C:\Users\Sysmon\Desktop\Sysmon>
```

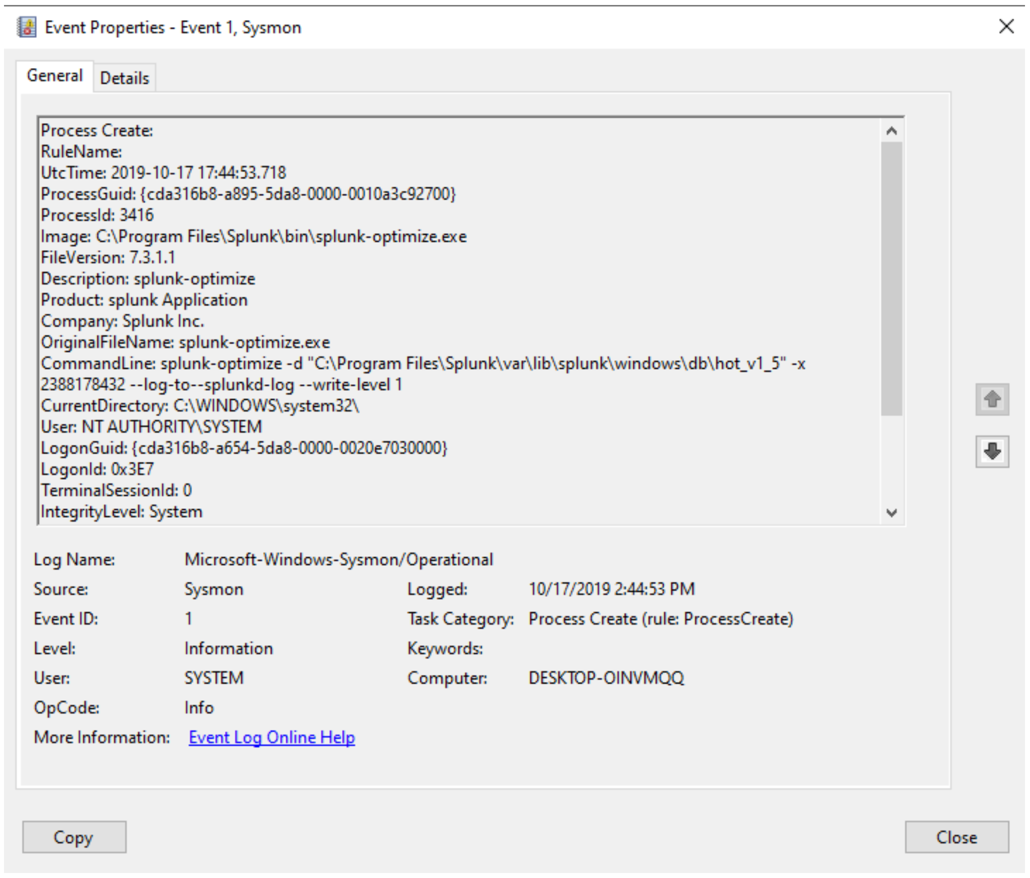
There is no need to restart the machine. All the steps will take effect right after they are executed.

To view the logs, open Event Viewer and navigate to the following on Windows Vista and newer.

```
Applications and Services Logs/Microsoft/Windows/Sysmon/Operational
```



```
Operational   Number of events: 63,668 (!) New events available

Level   Date a...   Source   Event...   Task Category
Inf...  10/17/...   Sysmon   1          Process Create...
Inf...  10/17/...   Sysmon   1          Process Create...
Inf...  10/17/...   Sysmon   1          Process Create...
Inf...  10/17/...   Sysmon   1          Process Create...
Inf...  10/17/...   Sysmon   1          Process Create...
Inf...  10/17/...   Sysmon   1          Process Create...
Inf...  10/17/...   Sysmon   1          Process Create...
Inf...  10/17/...   Sysmon   1          Process Create...
Inf...  10/17/...   Sysmon   1          Process Create...
Inf...  10/17/...   Sysmon   1          Process Create...
Inf...  10/17/...   Sysmon   1          Process Create...

Event 1, Sysmon
General   Details

Process Create:
RuleName:
UtcTime: 2019-10-17 17:44:53.718
ProcessGuid: {cda316b8-a895-5da8-0000-0010a3c92700}
ProcessId: 3416
Image: C:\Program Files\Splunk\bin\splunk-optimize.exe
FileVersion: 7.3.1.1
Description: splunk-optimize
Product: splunk Application

Log Name:           Microsoft-Windows-Sysmon/Operational
Source:             Sysmon              Logged:          10/17/2019 2:44:53 PM
Event ID:           1                   Task Category:   Process Create (rule: ProcessCreate)
Level:              Information         Keywords:
User:               SYSTEM              Computer:        DESKTOP-OINVMQQ
OpCode:             Info
More Information:   Event Log Online Help
```



```
Process Create:
RuleName:
UtcTime: 2019-10-17 17:44:53.718
ProcessGuid: {cda316b8-a895-5da8-0000-0010a3c92700}
ProcessId: 3416
Image: C:\Program Files\Splunk\bin\splunk-optimize.exe
FileVersion: 7.3.1.1
Description: splunk-optimize
Product: splunk Application
Company: Splunk Inc.
OriginalFileName: splunk-optimize.exe
CommandLine: splunk-optimize -d "C:\Program Files\Splunk\var\lib\splunk\windows\db\hot_v1_5" -x
2388178432 --log-to--splunkd-log --write-level 1
CurrentDirectory: C:\WINDOWS\system32\
User: NT AUTHORITY\SYSTEM
LogonGuid: {cda316b8-a654-5da8-0000-0020e7030000}
LogonId: 0x3E7
TerminalSessionId: 0
IntegrityLevel: System
```

Log Name:	Microsoft-Windows-Sysmon/Operational		
Source:	Sysmon	Logged:	10/17/2019 2:44:53 PM
Event ID:	1	Task Category:	Process Create (rule: ProcessCreate)
Level:	Information	Keywords:	
User:	SYSTEM	Computer:	DESKTOP-OINVMQQ
OpCode:	Info		
More Information:	Event Log Online Help		

Vita

Candidate's full name: Mohammad Rasool Fatemi

University attended:

Master of Computer Science
University of New Brunswick
2017-2019

Bachelor of Science in Software Engineering
Ferdowsi University of Mashhad
2012-2017

Publications:

M. R. Fatemi & Ali. A. Ghorbani: Threat Hunting in Windows Using Big Security Log Data. In Joshi, Ramesh C., and Brij B. Gupta. "Security, Privacy, and Forensics Issues in Big Data." IGI Global, 2020, (pages 168-188)

Samaneh Mahdavifar, A. Fitriah Abdul Kadir, M. R. Fatemi, Dima Alhadidi, Ali A. Ghorbani: Dynamic Android Malware Category Classification Using Semi-Supervised Deep Learning. In Computing Conference 2020

M. R. Fatemi, B. Bakhshi, A. Zamani, and B. Behkamal: A scenario-based approach for the behavior analysis of talented students. In 7th International Conference on Computer and Knowledge Engineering (ICCKE), 2017

Conference Presentations:

M. R. Fatemi & Ali. A. Ghorbani: Threat Hunting in Windows Using Big Security Log Data. In 16th Annual Research Expo, UNBF, 2019.

Samaneh Mahdavifar, M. R. Fatemi, Dima Alhadidi: Android Malware Categorization using a Semi-Supervised Deep Learning Architecture Based on LadderNetworks. In 15th Annual Research Expo, UNBF, 2018.