

The logo of the University of South Wales, featuring a red shield with a white border. The text "University of South Wales" and "Prifysgol De Cymru" is written in white on the red background.

University of
South Wales
Prifysgol
De Cymru

**An Integrated Cyber Threat Hunting Program
Applying Machine Learning for Enhanced Intelligence
Capabilities**

By

JOSHUA THOMAS RICHARDS
17025745

Supervised By

DR RICHARD WARD

A dissertation submitted in partial fulfilment
of the requirements of the degree of
MSc Computer Systems Security

University of South Wales | Prifysgol De Cymru
Faculty of Computing, Engineering and Science

9th of September 2021

Word Count: 14,728

STATEMENT OF ORIGINALITY

This is to certify that, except where specific reference is made, the work described within this project is the result of the investigation carried out by myself, and that neither this project, nor any part of it, has been submitted in candidature for any other award other than this being presently studied.

Any material taken from published texts or computerised sources have been fully referenced, and I fully realise the consequences of plagiarising any of these sources.

Student Name (Printed) JOSHUA THOMAS RICHARDS

Student Signature J. Richards

Registered Course of Study MSc Computer Systems Security

Date of Signing 09/09/2021

Abstract

This project entails the creation of a simplified event log filterer that can be used by new threat hunters to the field of threat intelligence. The event log filterer employs three filtering options that allow the threat hunter to analyse application, system, security and *Sysmon* logs, which are the most common logs used for hunting for threats. This tool was created to allow new users to get to grips with threat hunting without the need go through the hundreds of unnecessary logs that *Windows Event Viewer* collects and displays and the array of options that may seem overwhelming to those starting out in this field. The tool also analyses *Sysmon* logs using anomaly spike detection machine learning to highlight any logs that may be anomalous and should be investigated further. The idea of this is to allow the new threat hunter to pinpoint which *Sysmon* logs require attention, whereas the conventional *Windows Event Viewer* approach does not afford this anomaly detection and is a laborious task to sift through all the logs looking for anomalies, which is difficult for experienced hunters, let alone novices.

Keywords: Anomaly, Events, Detection, Logs, Machine Learning, Threat Hunting

Contents

Abstract.....	ii
List of Abbreviations and Acronyms	v
List of Figures.....	vi
List of Tables.....	vii
List of Equations.....	viii
1 Introduction	1
1.1 Project Aim	3
1.2 Project Objectives	3
1.3 Dissertation Structure	4
2 Literature Review	5
2.1 Current Threat Hunting Research and Projects	5
2.1.1 Data-driven threat hunting using <i>Sysmon</i>	5
2.1.2 Learning the Associations of MITRE ATT&CK Adversarial Techniques	8
2.2 Frameworks and Knowledge Bases	9
2.2.1 MITRE ATT&CK	9
2.2.2 Cyber Kill Chain.....	10
2.2.3 Diamond Model.....	11
3 Theory & Methodology.....	13
3.1 Threat Hunting	13
3.1.1 Methodologies	14
3.1.1.1 Hypothesis Driven Investigation.....	14
3.1.1.2 IoC Based Investigation	15
3.1.1.3 Analytics and Machine Learning Based Investigation.....	15
3.1.2 Steps and Processes	16
3.1.2.1 The Threat Hunting Loop.....	16
3.2 Windows Event Logs.....	18
3.2.1 Windows Event Viewer.....	18
3.2.1.1 Event Types.....	19
3.2.1.2 Event Sources	19
3.2.1.3 System Monitor (Sysmon)	20
3.3 Indicators of Compromise.....	21
3.4 Machine Learning	22
3.4.1 Anomaly Detection using ML.NET	22

4	Software Design and Development.....	23
4.1	VMware ESXi Lab Environment	23
4.1.1	Initial Setup	23
4.1.2	Testing Configuration.....	25
4.1.3	Pre-installing required applications	27
4.2	C# .NET Threat Hunting Application.....	28
4.2.1	Initial Layout and Design	28
4.2.2	Visual Studio and GitHub Setup	29
4.2.3	Core Functionality	30
4.2.3.1	Processing of Windows Event Logs.....	30
4.2.3.2	Processing of Sysmon Logs	32
4.2.3.3	Sysmon Machine Learning Anomaly Detection	33
5	Analysis.....	37
6	Evaluation and Conclusion.....	39
7	LSEPI	40
7.1	Legal	40
7.2	Social	41
7.3	Ethical	41
7.4	Professional.....	41
8	References	42
9	Appendices	45

List of Abbreviations and Acronyms

AD DS	<i>Active Directory Domain Service</i>
API	<i>Application Programming Interface</i>
APT	<i>Advanced Persistent Threat</i>
ATT&CK	<i>Adversarial Tactics, Techniques and Common Knowledge</i>
C2	<i>Command and Control</i>
CTIO	<i>Cyber Threat Intelligence Ontology</i>
CVE	<i>Common Vulnerabilities and Exposures</i>
DiD	<i>Defence in Depth</i>
DLL	<i>Dynamic-link Library</i>
DHCP	<i>Dynamic Host Configuration Protocol</i>
DNS	<i>Domain Name System</i>
GUI	<i>Graphical User Interface</i>
GUID	<i>Globally Unique Identifier</i>
ICMP	<i>Internet Control Message Protocol</i>
IDS	<i>Intrusion Detection System</i>
IoA	<i>Indicator of Attack</i>
IoC	<i>Indicator of Compromise</i>
IP	<i>Internet Protocol</i>
IPS	<i>Intrusion Prevention System</i>
NVD	<i>National Vulnerability Database</i>
OS	<i>Operating System</i>
SIEM	<i>Security Information and Event Management</i>
SOC	<i>Security Operations Centre</i>
Sysmon	<i>System Monitor</i>
UI	<i>User Interface</i>
UTC	<i>Coordinated Universal Time</i>
vCPU	<i>Virtual Central Processing Unit</i>
VLAN	<i>Virtual Local Area Network</i>

List of Figures

Figure 1 – Jøsang and Mavroeidis' (2018) CTIO relationship diagram	7
Figure 2 – The Diamond Model framework for threat hunting.....	11
Figure 3 – Threat Hunting 'Dwell Time' timeline	13
Figure 4 – The Sqrll Threat Hunting Loop.....	16
Figure 5 – Windows Event Viewer layout	19
Figure 6 – Main screen of the VMware ESXi web portal	25
Figure 7 – VMware ESXi deliverable testing environment network diagram (NEEDS UPDATING)	25
Figure 8 – Pinging and tracert the Pfsense firewall from the Windows Server 2019	26
Figure 9 – PowerShell script used to add CSV generated users to AD DS.....	27
Figure 10 – Installing Sysmon on the Windows Server 2019 VM.....	27
Figure 11 – Basic GUI layout for the primary WinForms deliverable application.....	28
Figure 12 – Deliverable application showing event logs that have been filtered.....	31
Figure 13 – Deliverable application showing more information for the specified event log	31
Figure 14 – Non-existent and non-numerical Instance ID error messages.....	32
Figure 15 – Comparison between EventLog and EventLogQuery classes for retrieving logs	32
Figure 16 – Sysmon log showing more event information when double-clicked.....	33
Figure 17 – GUI layout for the machine learning window of the WinForms deliverable application	34
Figure 18 – Generating a CSV file on the server using EvtxExplorer	35

List of Tables

Table 1 – Jøsang and Mavroeidis' (2018) Sysmon software threat level classification ...	6
Table 2 – Associations between the Cyber Kill Chain and MITRE ATT&CK frameworks	10
Table 3 – Windows Event Log Types	19
Table 4 – Sysmon twelve core capabilities	20

List of Equations

Equation 1 - pValue prediction of an anomaly.....	36
---	----

1 Introduction

This research project will necessitate the research, development and creation of an integrated cyber threat hunting tool that applies machine learning to enhance cyber threat intelligence capabilities. The final deliverable will be a *C# .NET* program that incorporates many diverse and advantageous hunting techniques, tools, and methods in one place for security researchers and analysts to use for a more streamlined approach to threat detection and hunting.

Applying machine learning models to these tools, techniques and query processes will preferably allow for greater success at hunting APTs (Advanced Persistent Threats) in the systems and cut down the current laborious work of threat hunters, which must examine vast amounts of logs of data and events daily from multiple diverse sources, using an array of different applications.

Threat hunting in essence is an involved and proactive defence strategy that is used by security analysts to detect, isolate, and analyse threats that manage to evade through current traditional security systems, such as firewalls, IDSs (Intrusion Detection Systems) and IPSs (Intrusion Prevention Systems) (Sqrrl Data, Inc., 2018). This is mainly done by analysing IOCs (Indicators of Compromise) on the target infrastructure. These IOCs are normally discovered during active hunts or are shared by security researchers globally for analysts to employ in their toolsets.

There are a broad number of techniques and tools that are used in industry to conduct threat intelligence hunting presently, and as such, there is no one size fits all approach, with different tools, processes and techniques being used for different types of perceived threats to the individual or organisation. One organisation may perceive threats differently to another organisation based on internal and external factors. Threat hunting requires that analysts can detect and then improve on these successful detections to develop new processes that can be used on future successful hunts (SANS, 2021).

This project will begin with the literature review into current research conducted in this field and then move onto an evaluation of current techniques, tools and processes that are currently used in industry in order to compare and contrast their advantages and disadvantages for their specified threat hunting tasks. Currently, many tools are either outdated or are not integrated into a single suite or application, which makes threat hunting a more difficult and laborious task than it needs to be. Guaranteeing that core

threat hunting processes are in a single application allows for greater productivity and collaboration of tools for threat analysts.

Finally, the second half of this dissertation report will discuss the design and development phase, as well as the evaluation of the *C# .NET* integrated program. This process also includes the testing of the application against a custom virtual environment to measure its performance against currently used tools available to threat analysts.

1.1 Project Aim

The main aim of this dissertation research project is to produce a fully functional and integrated threat hunting tool with the *C# .NET* programming language and framework which uses machine learning models to enhance intelligence gathering and analysis capabilities further than is currently limited by available open-source tools.

1.2 Project Objectives

To ensure that the main research project aim is achieved, there are eight objectives specified below that detail how the project will achieve its deliverable:

- Conduct a literature review to examine what developments and research have already been made in the field.
- Research the different threat hunting tools and understand how they are currently used and how effective they are.
- Analyse current machine learning algorithms and models used by ML.NET.
- Design and develop a draft program that collects logs and event data from the operating system and network.
- Expand on the program to analyse and categorise the data and IOCs based on the *MITRE ATT&CK™* (Adversarial Tactics, Techniques and Common Knowledge) framework.
- Use the most effective machine learning algorithms from the program to categorise logs and events to determine system baselines and deviations.
- Tweak the machine learning algorithm in the programme in order to optimise the hunting and categorisation results.
- Test the program in a custom virtual environment by simulating custom APTs and standard attacks on a virtualised network infrastructure.

1.3 Dissertation Structure

This master's dissertation has been organised, excluding the previous chapter; Chapter 1 – Introduction, into the following key chapters for clarity and convenience of reading:

Chapter 2 – Literature Review: This chapter covers the background knowledge regarding the project and includes introductory theoretical analysis of the current threat hunting tools, current research projects and machine learning algorithms, which will be expanded on further in Chapter 3.

Chapter 3 – Theory and Methodology: Presents the main theories of the project and analyses their application. Expands on the theories of threat hunting, machine learning algorithms used in this project as well as necessary network, log, and event data theory.

Chapter 4 – Software Design and Development: Covers the steps taken to design, implement, and develop the *C# .NET* integrated threat hunting program using *Visual Studio 2019*. Discusses the stages of development and versioning control using *GitHub*. A detailed breakdown of the creation of the virtual environment will also be covered in this chapter.

Chapter 5 – Analysis: This chapter will cover the results, validation, and testing of the program deliverable by critically analysing its effectiveness against a virtual environment consisting of APTs and conventional threats. This stage will also cross reference with existing tools to evaluate the differences between their use cases, speed and hunting success rates.

Chapter 6 – Evaluation and Conclusion: Includes the project conclusion and lessons learnt. Discusses further research that can be carried out regarding this project's outcomes and what could be done to improve on this project.

Chapter 7 – LSEPI: A thorough discussion of the legal, social, ethical, and professional issues within this project and how said issues have been managed or curtailed.

2 Literature Review

A literature review is a research and evaluation conducted into the current literature in a given topic and field, which documents the state of the art with regards to the topic being researched (Royal Literary Fund, 2019). This literature review will specifically look at current research and projects that could be expanded on in this field, as well as an overview of the current open-source tools that are used to conduct threat hunting. Alongside this, the frameworks which govern the approach and stratagem of threat hunting will be analysed, as they are useful in understanding how the program deliverable will be designed and developed to be most effective.

Threat hunting, and the development of hunting tools, has become more and more beneficial and necessary for organisations as it allows them to map out adversaries and their actions on their infrastructure in the event that traditional security systems such as IDSs and firewalls are compromised or fail. In this sense, threat hunting can be considered to be a proactive approach to securing company networks and assets by analysing and understanding the TTPs (tactics, techniques and procedures) used by hostile threat actors to intrude, access, manipulate, obfuscate and exfiltrate data from the network or to achieve their specified goal (Xiong, et al., 2021).

2.1 Current Threat Hunting Research and Projects

The cyber security field is truly diverse and ever-changing with new discoveries and techniques being conceptualised daily. This literature review will therefore home in on an exceedingly small aspect of this colossal field by reviewing and analysing research and projects that have been conducted mainly with the use of open-source tools within the threat hunting and intelligence domain. This literature review will pay little attention to industry-standard or open-source tools, as these will be analysed and reviewed in the subsequent chapters individually.

2.1.1 Data-driven threat hunting using *Sysmon*

Successful threat hunts all use an assortment of collected system and network data in order to determine and analyse the threats posed to them. The data that is collected is analysed by different tools and processes in order to categorise and prioritise them based on the possible modus operandi of the threat actor (Palacín, 2021). With large amounts of data being collected constantly, the majority of which is not necessarily useful for hunts, and as a consequence, does more harm than good by prolonging and potentially

jeopardising the threat hunting process, it is important that appropriate filtering is done to limit, sanitise and select the most useful data from these diverse sources.

One of the main tools used by hunters and administrators alike that is available for *Windows* OSs (Operating System) is *System Monitor*, also more commonly known as *Sysmon*. The *Sysmon* service does not provide analysis of data or logs, however, it is a service that monitors and logs system activity to the *Windows* event log. Some of the main attributes of *Sysmon* are that it collects data on network connections, process creations, changes to file creation times, loading of drivers and libraries as well as other useful records (Garnier & Russinovich, 2021).

Sysmon has been around since 2014, and is free to download, as it does not come pre-installed on *Windows* OSs as standard, as there is already a basic built-in event logger. As stated previously, it does not analyse the data and only collects it, therefore, it is used in conjunction with either *Windows Event Viewer*, SIEM (Security Information and Event Management) software or custom code and APIs (Application Programming Interface) that can be programmed to dissect and analyse the collected logs and events. With these logs it is possible to detect and identify malicious activity taking place on the system as well as understanding how the hostile threat actor operates in particular circumstances on the network infrastructure where the logs were collected.

A conference paper published by Jøsang and Mavroeidis (2018), researchers from the University of Oslo, discusses the use of *Sysmon* to create a real-time threat assessment system to analyse and classify software based on four devised categories: high, medium, low, and unknown threats. This categorisation of software is based on characteristics they have identified in the paper, as noted in *Table 1* below.

Threat Level	Identified Software Characteristics
Low	Possibly non-malicious software
Medium	Legitimate software but vulnerable/ used by hostile threat actor to perform attack
High	Malicious software/ Legitimate or unknown software with relationship with malicious IOCs
Unknown	Unknown software without known relationships to threat actors or malicious IOCs

Table 1 – Jøsang and Mavroeidis' (2018) Sysmon software threat level classification

Jøsang and Mavroeidis (2018) also introduce the idea of a CTIO (Cyber Threat Intelligence Ontology) that is used in order to support decision making by representing a wide array of information such as threat actors, target infrastructure, their objectives and goals, their motivations, their TTPs, IOCs and vulnerabilities. This information is used to create the diagram (figure 1) that displays the relations and links between them.

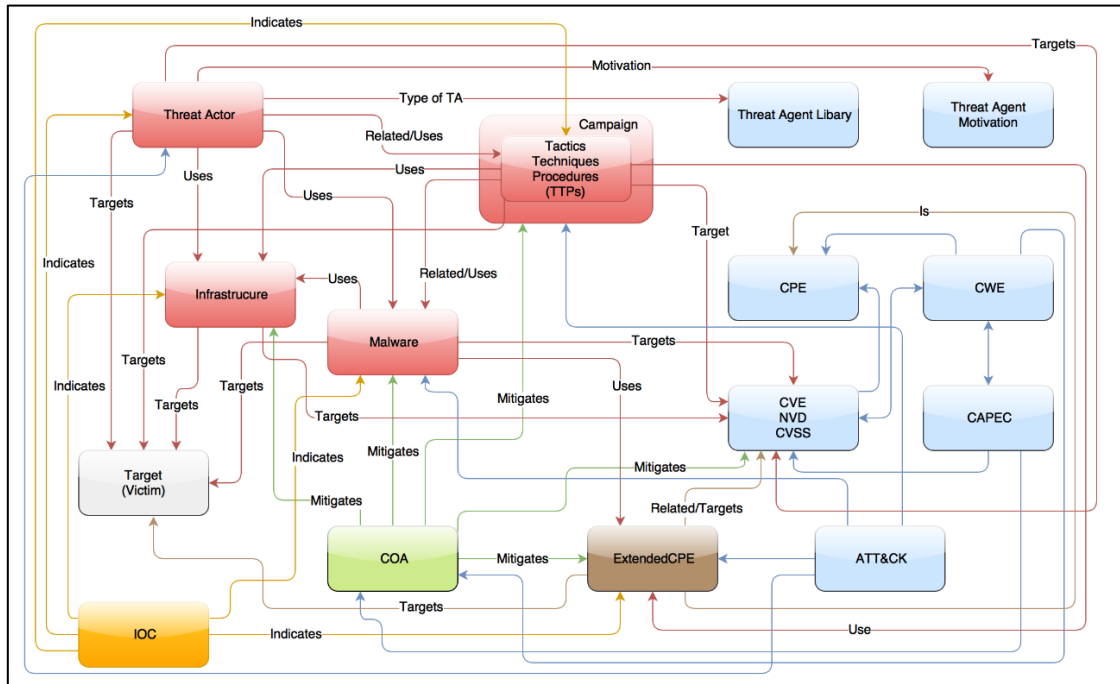


Figure 1 – Jøsang and Mavroeidis' (2018) CTIO relationship diagram

The main take away from Jøsang and Mavroeidis' diagram is that it conveniently portrays the main relations between the malware that influences the IOCs, which are in turn related to the TTPs of the associated threat actor. These TTPs are then extensions of the information held within the *MITRE ATT&CK*TM knowledge base as well as the CVE (Common Vulnerabilities and Exposures) and NVD (National Vulnerability Database) databases. This diagram also implies that different threat actors use different infrastructure and TTPs that could provide knowledge as to who is conducting the attack and allows the analyst to pin-point their threat hunting further by targeting and analysing specific elements related to the threat actor in particular.

The diagram provides a useful reference as to how the threat hunting tool deliverable of this project should interact with the resources and logs collected by the system infrastructure to maximise its proficiency and overall efficacy. As this project will also be looking at how machine learning can assist with the integrated threat hunting

process, it will be useful to consider how the threat hunting program created in the research paper compares with the final deliverable of this project.

Further analysis of the inner workings and use cases of *Sysmon* will be discussed in the next chapter of this dissertation (Chapter 3 – Background & Theory), with analysis on its effectiveness against the final deliverable being conducted in Chapter 6 – Analysis.

2.1.2 Learning the Associations of MITRE ATT&CK Adversarial Techniques

The research paper published by Al-Shaer, Christou and Spring (2020) focuses on the use of statistical analysis and machine learning on already collected datasets used by *MITRE ATT&CK* to determine the correlations and associations between the techniques used by adversarial threat actors. The results from this paper show that hierarchical clustering machine learning models have a confidence of 95% in explaining significant associations between adversarial techniques (Christou, et al., 2020). They note that although the *MITRE ATT&CK* knowledge base provides ample information on the TTPs that a hostile threat actor would use, it is lacking in how those different TTPs are combined with each other in order for the attacker to achieve their goal.

By using machine learning models, they have shown that some technique associations can be predicted to help the analyst understand the adversarial behaviour of the threat actor based on their mixed usage of multiple TTPs. As of yet, limited research has been conducted specifically into the correlation between techniques, and as such, this paper provides a great foundation to build upon, specifically using the findings in order to produce a machine learning model that can be used against real-time data and log gathering services for enhanced threat hunting capabilities.

There are however limitations outlined in this research paper, the main being that the APT and software attacks registered on the *MITRE ATT&CK* knowledge base are not representative of all possible techniques that a threat actor is capable of, and as such, is only a small portion of the available open-source data gathered by analysts based on already occurred events and incidents. Therefore, it is fairly difficult to surmise and determine what techniques were used for specific attacks without the supporting data, however, it is still possible to make fairly accurate and useful predictions that can be used for more effective threat hunting compared to non-machine learning analysis (Christou, et al., 2020).

2.2 Frameworks and Knowledge Bases

Cyber threat hunting tools are inadequate unless there is a solid and continuously updated knowledge base for them to reference and utilise. There are various frameworks and knowledge bases that are used by analysts to determine and analyse the TTPs that are used by threat actors and how those techniques correlate and merge to form effective attacks against an individual entity or an organisation's physical and logical infrastructure.

2.2.1 MITRE ATT&CK

The most globally used and respected knowledge base to date is *MITRE ATT&CK* that was conceived in 2013. It was designed in order to detail the TTPs and activities that attackers and adversaries may use and are capable of carrying out at different stages of the cyberattack to achieve their objectives within the victim environment (Palacín, 2021).

The most important aspect of the *MITRE ATT&CK* framework is that it provides a standardised way for security researchers and analysts alike to describe the supposed behaviour of adversaries and threat actors. This allows for the researchers and analysts to share their ideas and insights with a more structured approach which overall fosters a better understanding of the adversarial behaviours, both between experienced and non-industry professionals who may be interested.

The framework is not static, in the sense that it is not immutable, it can and should be used to develop and support the building of custom bespoke TTPs based on in-field experience and analysis of uniquely observed adversarial behaviours and capabilities. In total there are fourteen tactics in this framework that in turn envelop multiple different sets of techniques and sub-techniques that a threat actor may use (Palacín, 2021). These techniques and sub-techniques describe the threat actor's and adversaries' behaviours while the tactics represents the goal of the threat actor's behaviour. Finally, the 'P' in TTP refers to the procedure, which is the way in which the adversary carries out the specific techniques.

The fourteen tactics used in the *MITRE ATT&CK* framework are as follows: Reconnaissance, Resource Development, Initial Access, Execution, Persistence, Privilege Execution, Defence Evasion, Credential Access, Discovery, Lateral Movement, Collection, Command and Control, Exfiltration, and Impact. Each of these tactics have

numerous techniques that can be used ranging from seven techniques all the way up to thirty-nine techniques (MITRE, 2021).

2.2.2 Cyber Kill Chain

Both the *MITRE ATT&CK* and the *Cyber Kill Chain* frameworks follow the basic sequence of the attacker infiltrating, conducting their activities and then exfiltrating from the infrastructure, ideally, without getting caught or suspected of any intrusion. The major difference between the two frameworks however is that the *MITRE ATT&CK* framework denotes a list of techniques that an adversary may use in any particular order to achieve their goals, whereas the *Cyber Kill Chain* primarily lists the order of operation of an adversarial attack without further detail as of how to conduct said attack against a victim (Hutchins, et al., 2011).

The *Cyber Kill Chain* consists of seven distinct stages that must be completed by a threat actor in order for a successful cyber attack to take place. Subsequently, the *Cyber Kill Chain* also allows for system administrators and blue teams to develop an intelligence feedback loop which enables them to understand the adversary’s actions at specific stages whilst establishing a state of intelligence superiority over the threat actor. This in turn enables the defenders to mitigate the likelihood of success for the adversary during any future intrusions that may occur (Hutchins, et al., 2011).

The seven stages of the *Cyber Kill Chain* are Reconnaissance, Weaponisation, Delivery, Exploitation, Installation, C2 (Command and Control), and Actions on Objectives. These seven stages can be informally and loosely associated with the fourteen tactics used in the *MITRE ATT&CK* framework as shown below in *Table 2*.

Steps	Cyber Kill Chain	MITRE ATT&CK
1	Reconnaissance	Reconnaissance
2	Weaponisation	Resource Development
3	Delivery	Initial Access
4	Exploitation	Execution
5	Installation	Persistence
6	C2	Privilege Execution Defence Evasion Credential Access Discovery Lateral Movement Collection Command and Control
7	Actions on Objectives	Exfiltration Impact

Table 2 – Associations between the Cyber Kill Chain and MITRE ATT&CK frameworks

2.2.3 Diamond Model

This model puts emphasis on the characteristics and relationships between four of the essential components of threat intelligence analysis, these include: Adversary, Capabilities, Infrastructure and Victim vectors. The paper presented by Betz et al. (2013) conceptualises this model and in its simplest form is described as “*an adversary deploys a capability over some infrastructure against a victim*”.

The Adversary vector is the organisation or malignant threat actor responsible for carrying out an attack by carrying out a capability against the victim. The Capability vector is the tools, techniques and tactics used by the adversary during an attack. The Infrastructure vector includes the logical and physical infrastructure, such as the network, IP (Internet Protocol) addresses, domain names, emails, websites etc. that are used by the threat actor and adversary to deliver the capability. Finally, the Victim is the target, be it an organisation or an individual, in which the vulnerabilities are exploited, and the adversarial capabilities are enacted against.

In essence, it can be determined that an intrusion event is defined as how the threat actor can demonstrate an attack against a target over an infrastructure using certain distinctive capabilities and techniques (Betz, et al., 2013). The diagram pertaining to the model that is presented in the paper is shown in *figure 2* below.

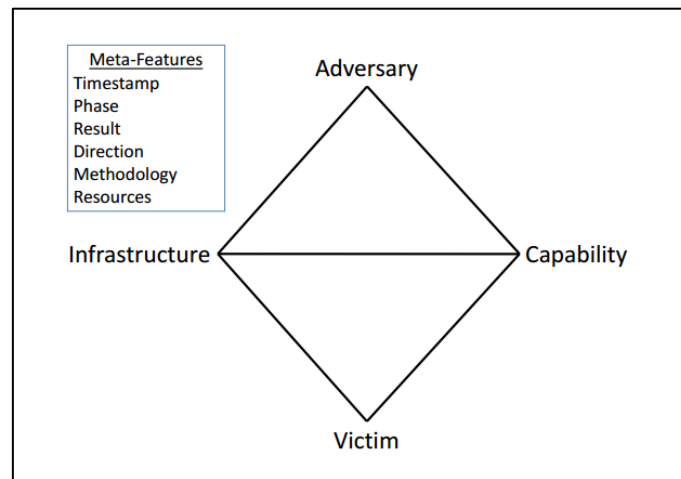


Figure 2 – The Diamond Model framework for threat hunting

Comparable to the *Cyber Kill Chain*, this model does not represent the TTPs used by the adversaries, as is the case with the *MITRE ATT&CK* framework, but instead maps the relations between the four core features of an event. This model allows researchers and analysts to build and understand the definite relationships between the four vectors

and how said relations act with or are restricted by each other. By doing so, threat analysts can understand in greater depth and clarity the intentions and TTPs of the adversarial threat actors throughout this process.

3 Theory & Methodology

This chapter will mainly focus on the theory of threat hunting, networking and IOCs, types of attacks as well as the theory behind the machine learning algorithms used in the final project deliverable. Furthermore, this chapter will cover the methodology used in order to produce the final deliverable including the software and frameworks used.

3.1 Threat Hunting

As discussed previously, threat hunting is a proactive and continuous approach to searching and analysing cyber threats that are lurking and hiding undetected on the network infrastructure. These threats will have bypassed conventional security mechanisms and apparatuses such as IDSs, IPSs, anti-viruses, and firewalls and, as such, pose a threat to the system and wider infrastructure if they continue to remain undetected. It is possible for threat actors and adversaries to remain undetected for many months, even years, whilst exfiltrating sensitive data from the organisation even as security controls are completely oblivious. A threat actor that successfully penetrates the core defences and front lines of the infrastructure and evades detection can be defined as an APT.

The core aim of the threat hunter is to reduce what is known as the dwell time, which is the amount of time that passes from when the adversary has breached and established themselves within the environment and when the breach has been detected by either conventional means or by a human operator (Palacín, 2021). The dwell time can range from a few minutes to many weeks, with the average dwell time, according to *SANS Institute*, being more than ninety days, which has dropped considerably since 2013, where the average dwell time was reported to be well over six months (Lee & Lee, 2018). The following *figure 3* illustrates the theoretical timeline from an adversary entering the network, being detected by conducting threat hunting and then carrying out the incidence response operation for recovery.

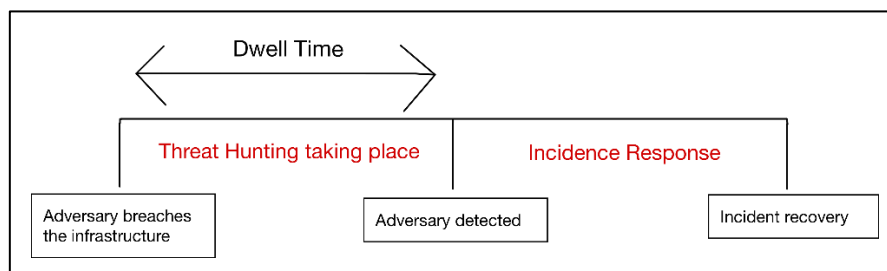


Figure 3 – Threat Hunting 'Dwell Time' timeline

It is important to note that the struggle to lower the dwell time is continuous, that is, the threat actor is going to carry on adapting to the new faster and more advanced detection methods employed by the threat hunters and as such improve their infiltration and evasion techniques to remain undetected whilst breaching the systems quicker in their own right. The majority of organisations and companies lack the required skills, detection methods, tools, and personnel necessary to prevent an APT from taking a foot hold on the network infrastructure once the adversary has bypassed the defensive boundaries, therefore, it is important that threat intelligence and hunting play a hands-on role in the defence stratagem of the organisation alongside traditional security approaches and procedures.

Defence in Depth (DiD), which consists of multiple layers of tailored security controls protecting key assets, seeks to delay rather than completely prevent an attacker from penetrating the network, as there will always be a way inside with new vulnerabilities and zero-days being discovered on a daily basis by researchers to state actors. This strategy can be relied upon to give time for the threat hunters and analysts to detect and analyse the motives as well as the TTPs that the adversaries are currently utilising, however, it should not be completely depended upon to entirely defend the environment and infrastructure. This is where threat hunting plays a significant role once the adversary has bypassed the typical array of defences (Lippmann, et al., 2006).

3.1.1 Methodologies

There are numerous different methodologies and approaches to conducting threat hunting investigations, each using different parameters and data to carry out the analysis of the threats with varying results. Among the many approaches, the three main and most commonly used methodologies include the hypothesis driven approach, IoC approach, and advanced analytics and machine learning method (Taschler, 2021).

3.1.1.1 Hypothesis Driven Investigation

This approach is based on using already collected data from researchers that have reported on novel attacks to a knowledge base and other sources to determine whether a particular attack, or similar, has occurred within the environment that is under investigation. This methodology to threat hunting relies on researchers updating and informing the community of the attacks and their behaviours alongside the attackers unique TTPs and approaches for a specific attack that has been carried out.

The TTPs that are composed and distributed with the community allow the threat hunter and investigators to create a hypothesis to identify whether the same TTPs have been acted upon within the infrastructure, which could lead to a successful hunt and investigation based on the current modus operandi of an active adversary.

3.1.1.2 IoC Based Investigation

The IoC based investigation approach is similar to the hypothesis driven investigation, however, instead of relying solely on community-based research and announcements of a new threat, also known as external sources, to commence a threat hunt, the investigation starts centred on the organisation's threat intelligence data, known as internal sources, that have already been conducted as part of the organisation's continuous investigative threat intelligence cycle.

Threat intelligence can be defined as *“evidence-based knowledge, including context, mechanisms, indicators, implications and actionable advice, about an existing or emerging menace or hazard to assets that can be used to inform decisions regarding the subject's response to that menace or hazard”* (Gartner, Inc., 2014). Threat intelligence is the collection of data that is processed and then analysed in order to understand the adversary's behaviours, motives, targets, and goals.

The threat hunter will leverage the analysed data collected from the threat intelligence stage to identify IoCs and IoAs (Indicators of Attack) associated with new and emerging threats. The IoCs and IoAs are used as triggers that are employed for the uncovering of what could be undetected hidden attacks or malicious activity taking place on the infrastructure as part of a wider APT. A large volume of IoCs and their corresponding data does not necessarily indicate that there is a threat or that a hunt will be carried out successfully, therefore, it is important that the threat hunter takes into consideration quality over quantity when collecting the data and that the data is relevant to the hypothesis posed.

3.1.1.3 Analytics and Machine Learning Based Investigation

The final methodology includes the use of both machine learning and data analytics to filter through vast amounts of logs and data to detect any abnormalities that could predict and indicate if malicious activity is taking place. The machine learning algorithms are used in order to learn relations and links between different parameters and are also trained to predict the sequencing of novel threat events based on the data of

previous events. These algorithms alongside their training data are used in order to create trained models that can be used to predict threats and allows the threat hunter to conduct further analysis based on the results of the prediction.

This research project will mainly be focusing on this final methodology and will incorporate machine learning techniques into a threat hunting tool that can predict new emerging threats based on event logs collected by the computers and central server, where the deliverable executable will be residing and running. **Reference + Example of real life use case.**

3.1.2 Steps and Processes

There is no standardised universally agreed upon list of steps or processes for completing a successful threat hunt amongst specialists in the field, however, there are some useful mechanisms that highlight and lay out the iterative process of a threat hunt for SOC (Security Operations Centre) teams to follow.

One of the first processes conceptualised for threat hunting is the ‘Threat Hunting Loop’ which contains four iterative steps in a loop that hunters should follow as quickly and efficiently as possible to succeed in the hunt (Sqrri Data, Inc., 2018). The four steps are: Hypothesis Creation, Tool and Technique Enabled Investigation, Pattern and TTP Discovery, and Advanced Analytics.

3.1.2.1 The Threat Hunting Loop

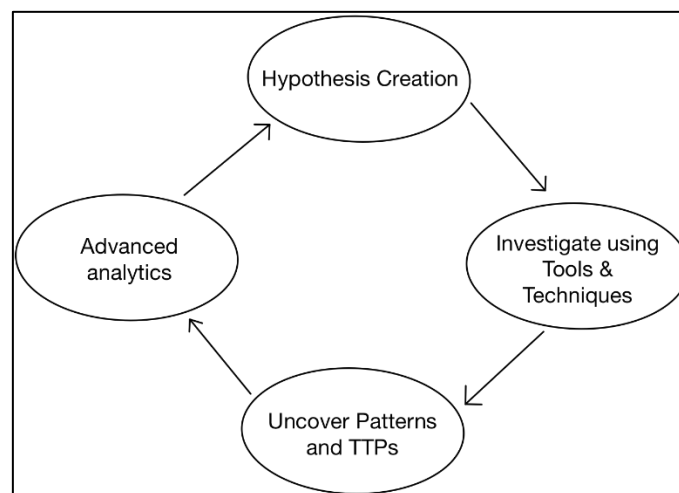


Figure 4 – The Sqrri Threat Hunting Loop

The first step of the loop devised by *Sqrri* is the hypothesis creation. This initial step involves the threat hunters devising hypotheses by approximating and making an

educated guess as to what sort of activity may be going on in the investigative environment. This hypothesis can be built on the intelligence gathered from internal or external sources, or both at the same time, by allowing the hunter to generate specific hypotheses related to the threats posed specifically to the organisation in question, which may be different for different organisations and individuals. The hypotheses that have been devised can also be split into further sub-hypotheses that can be analysed individually in further depth and more detail. For example, an employee may be at risk of being targeted and tracked by state actors if they have recently been abroad, therefore, setting up a hunt to determine if their equipment have been compromised would be a hypothesis.

The second stage in this iterative process is the use of investigative tools and techniques. This step allows the hunters to propose answers to the hypotheses that have been prepared in the first stage using a wide array of data and logs by trying to understand and discover the adversaries TTPs for this specific attack or overall adversarial TTPs for a multitude of ATPs. These tools will use a variety of techniques such as statistical analysis and machine learning by analysing and combining knowledge bases and datasets for model learning and creation.

The third stage involves the hunters uncovering the malicious behaviours and patterns that the adversaries are undergoing whilst also trying to determine their TTPs. This will involve collaboration between different teams and outside bodies to understand the tactics used by adversaries and compare whether similar tactics are being used against other organisations.

Lastly, the fourth stage is the automation and reporting of hunts. There is no reason for the SOC team to be repeating the same hunts over and over again that produce the same results, therefore, automating these hunts will allow the team to concentrate on other activities that are vital for the overall hunting process. These automations can be created in many ways, by creating queries to search specific logs of significance, creating scripts using different programming languages and tools, as well as using supervised machine learning algorithms and providing them with affirmation of an identified pattern of events from the successful hunt to learn from to conduct further automated hunts of a similar nature in the future.

3.2 Windows Event Logs

Every application on the *Windows* OS, as well as other OSs, log events such as errors or general information that may be useful to the system in the future, be it for referencing or reporting errors. These event logs can both be software or hardware events, however, with different applications and hardware using their own proprietary error reporting and logging, it is not straightforward to merge all of these different formats into one reporting system. This is where the event log plays a significant role to standardise, categorise and centralise all system logs, from the hardware, software, and the OS itself (Microsoft, 2018)

These *Windows* event logs can be viewed and examined by the user using the *Windows Event Viewer* administrative tool that is pre-installed with *Windows*, by using *PowerShell* commands and scripts, or they can also be retrieved by using classes within the .NET framework whilst creating a custom application, as is the case with this dissertation's deliverable.

3.2.1 Windows Event Viewer

The *Windows Event Viewer* can be used by administrators to view all application and system logs in a structured manner, with options to search and filter based on types, categories, sources, level and so on. Not all events that are logged in *Windows Event Viewer* are malicious, such as application crashes or services failing to start, *Windows* will log everything imaginable in order to keep a track of events for future reference or troubleshooting.

The main UI (User Interface) features a rather simple-looking GUI (Graphical User Interface), however, it can be quite overwhelming to novice users. The composition of the application contains a tree structure in the left pane that lists the different type of log categories, such as Application, Security, and System. Clicking on one of these event logs will display all those events in the main pane, which includes information such as the level, date and time, source, and event ID (also known as instance ID).

Below the main panel is information regarding the selected event log, such as the computer that generated the log as well as a more detailed message of what has happened. Finally, the right-most pane includes many buttons for filtering, saving, clearing, and refreshing event logs (Hassell & Campbell, 2007). All of these panes have been highlighted in *figure 5* below.

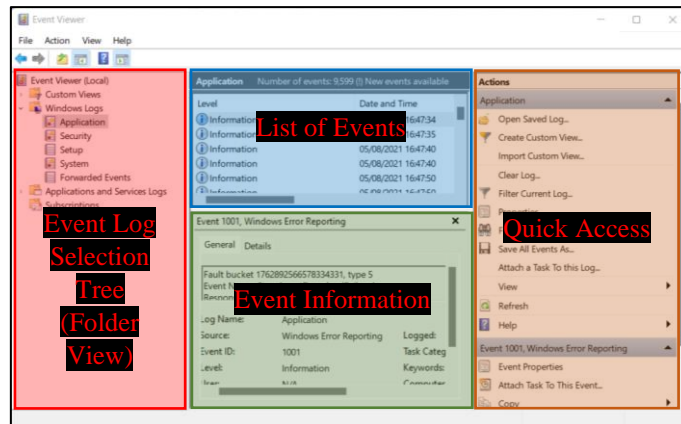


Figure 5 – Windows Event Viewer layout

3.2.1.1 Event Types

There are five types of events that the *Windows* OS can declare and categorise. When the piece of software declares an event log, it will also declare the type of event that is being reported. Only one event type can be declared per event with the *Windows Event Viewer* displaying different icons for each of the five event types for ease of viewing and analysis. The five types of events are: Error, Warning, Information, Success Audit, and Failure Audit (Bridge, et al., 2018). *Table 3* draws attention to the characteristics of the events that are categorised by the OS.

Event Type	Description and Characteristics
Error	Significant issues such as loss of data, functionality or loading failures.
Warning	Non-significant events but could cause a future issue/problem. An example would be low storage space.
Information	Used when an application, service, device, driver etc. is successful in operating its defined task.
Success Audit	This type is given to security audit attempts that are successful such as a user logon attempt.
Failure Audit	This type is given to security audit attempts that fail such as a logon failure.

Table 3 – Windows Event Log Types

3.2.1.2 Event Sources

Event sources are subkeys of the event logs and are named after the subcomponents of the application that logs the events. Within each event log there are multiple sources, such as different applications that are creating each individual log. The main event logs are the Application, System and Security logs. The security log is only used for system use and requires administrative privileges to access, the system log is primarily used for driver logs, while the application log is used for applications and

services to log events (Bridge, et al., 2018). It is also possible for other unique log categories to be created to, such as is the case with the third-party created *Sysmon* logs.

3.2.1.3 System Monitor (Sysmon)

The *Sysmon* service, that is installable from the *Microsoft* website, remains active even after the OS reboots and logs all of its activities to the *Windows Event Log*. Once it has been installed on the log gathering system, normally a server, it has many useful capabilities, especially with regards to identifying malicious activity and trying to identify the TTPs of a threat actor based on how they are operating across the network infrastructure (Russinovich & Garnier, 2021).

According to *Microsoft* (2021), *Sysmon* contains twelve key capabilities that assist threat hunters with their investigations, with details of said capabilities outlined in *Table 4*.

The logging of process creation	The logging of raw access to disk drives and disk volumes.
Logs the hash of image files using MD5, SHA1 and SHA256.	The logging of network connections such as connection source processes, IP addresses, port numbers, port names and hostnames.
Multiple different hashes can be used simultaneously.	Identify and detect any changes in file creation time to verify creation time. Hackers commonly modify file creation timestamps to cover the tracks of the malware.
Includes process GUID (Globally Unique Identifier) during event creation for events that have been denied event IDs.	The automatic reloading of configurations if the registry has changed them.
Includes session GUIDs for every event for correlation between events in the same current session.	The active filtering of events based on rules to include or exclude specific events.
The logging of DLLs (Dynamic-link Library) and drivers including their signatures and corresponding hashes.	Can log and capture event logs from the early stages of the boot process to log activity made by kernel-based malware.

Table 4 – Sysmon twelve core capabilities

The *Sysmon* events in the *Windows Event Viewer* are stored within the ‘Application and Services Logs/Microsoft/Windows/Sysmon/Operational’ folder of the *Event Viewer’s* folder view in the left-most pane. For clarity across systems and to avoid confusion, all *Sysmon* event logs are logged using the UTC (Coordinated Universal Time)

time zone, this is especially important if logs are being collected from servers located in datacentres or offices in different countries across the globe (Russinovich & Garnier, 2021).

Sysmon logs include twenty-seven event IDs (1 through 26 and error ID 255) that cover the capabilities detailed in the aforementioned table. The event IDs for *Sysmon* logs can be used in order to filter based on different characteristics and IoCs and are not too exhaustive in that they can be categorised with relatively little confusion as to what the event is pertaining to. Filtering also allows the hunter to clear a lot of unnecessary noise generated by the service that may not be relevant to the investigation. Event ID 255 is an error event and occurs solely when there's a problem with *Sysmon*, which could occur if there is heavy load on the system or if there is a general bug within the *Sysmon* service that is preventing it from working correctly.

3.3 Indicators of Compromise

The discovery of IoCs is the main way that threat hunters gather evidence and trails as to whether a malicious activity has taken place on the system or network. In essence, IoCs act as the trail of artefacts that threat actors leave behind that allow hunters to detect early on in the attack if anything malicious is taking place or indicate if an attack may take place (Lord, 2020).

IoCs can be anything that the attacker has left behind during their activity, they can be something small such as metadata of files and images, to IP addresses and all the way to more complex bespoke malignant code. Threat hunters and analysts alike will try to create correlations between all the collected and informative IoCs in order to create a pattern by piecing all the parts together to determine the likelihood of an incident.

Similar to IoCs, IoAs will instead focus on artefacts that are generated by an attacker whilst an attack is occurring. IoCs can be used in order to determine what has happened while IoAs can be used to determine what is currently happening and why. Ideally, threat hunts should use both IoCs and IoAs simultaneously to get near enough real time analysis of a security incident.

3.4 Machine Learning

This project deliverable will utilise a basic machine learning algorithm in order to enhance the threat hunting capabilities of a threat hunter. The machine learning algorithm will be using the ML .NET framework specifically for C# applications. This machine learning algorithm and model will be incorporated into the main application deliverable and will complement the basic event viewer and filter that has been implemented using C#.

The algorithm will be analysed against the *Sysmon* event logs collected by the application on the server and will determine whether an anomaly may be present or not.

3.4.1 Anomaly Detection using ML.NET

The machine learning algorithm used in this .NET application is known as a time series anomaly detection. This algorithm is used in order to detect anomalies from a large dataset of events by determining whether any of the datapoints are unexpected that could indicate an unknown attack. The ML.NET library allows for the usage of spike detection, which in essence, will indicate to the user and spikes in abnormal behaviour and deviations (Microsoft, 2021).

It is important to note that this time series anomaly detection, unlike some other machine learning algorithms, such as supervised approaches, does not need any training data, as such, it uses the input data to produce a transform and uses this to produce a data schema to create a data view for transformation and prediction. In this sense, the data that has been collected from the event logs is the same data that will be analysed for any anomalies and spikes, there is no need for any external datasets for training as with supervised machine learning algorithms. Time series anomaly detection can be considered an unsupervised machine learning approach due to the very nature that it uses unlabelled raw data that has no previous classification to determine whether the log is anomalous or not based on a baseline and normal system behaviour.

This unsupervised approach, however, has a few drawbacks in that it cannot know for certain, or to a high degree, whether a log is malicious or not, that is not the designed intention of this algorithm. Instead, the algorithm will determine whether it is an anomaly that deviates from the norm and indicate it as such. The results from this allow the threat hunter to analyse these logs that are anomalous further to determine whether there is any malicious activity. However, the main reason this unsupervised approach has been

selected is that creating or getting a large enough dataset for a supervised model would be difficult, time consuming and computationally immense, especially so is the case where each network environment and infrastructure set-up is different, with different network traffic, activity, and baselines, which would create bias for new unknown data that the algorithm would receive, producing unfavourable results (Landauer, et al., 2018). Generating a clean dataset for a supervised approach would also require an anomaly-free environment, which would be very difficult under most circumstances, as there is always traffic and events that could be considered anomalous and abnormal on all computers, be they small or large anomalies.

4 Software Design and Development

This chapter will explore the software design and development of the C# application using *Visual Studio 2019*, as well as going through the stages taken to set up the *VMware ESXi* lab environment that is used to test the effectiveness of the completed C# application deliverable against open-source tools.

4.1 VMware ESXi Lab Environment

VMware ESXi is a hypervisor product by the same company that creates *VMware Workstation Pro*. *ESXi* can be installed on bare metal infrastructure, such as a standalone server, or it can also be ran within a VM itself, which is the preferable option for this research dissertation as it does not require the purchasing or usage of a physical server. The installation process is as straightforward as it is with installing any regular VM using an ISO disk image file from the *VMware* website.

The lab environment contains multiple machines in order to simulate a small company network for purposes of testing the already available hunting tools, as well as the project deliverable. In total, the *ESXi* environment will include one *Pfsense* firewall VM, one *Windows Server 2019* VM, two *Windows 10* client VMs, and one *Ubuntu Linux* VM.

4.1.1 Initial Setup

Running through the installation process takes a few minutes and is self guided, however, the online guide provided by *VMware* is helpful in ensuring that all configuration options have been set up correctly¹. The *ESXi* hypervisor VM has been

¹ VMware ESXi (2021) available to download at: <https://customerconnect.vmware.com/en/web/vmware/evalcenter?p=free-esxi7>

designated 8 GB of RAM for use initially, however, this may become a limitation with the more resource intensive VMs that are added; therefore, 10 GB of RAM may be required. The total amount of available RAM on the PC being used for this research project is 16GB, so there is enough provision available if needed. The amount of disk space allocated for the hypervisor VM is 142 GB, this is being used for the hosting of the *ESXi* setup and configuration data, the ISO files on the hypervisor server for creating the VMs, as well as the hard disks of the individual VMs.

Each of the two *Windows 10* client VMs have been assigned 15 GB of disk space, 2 GB of RAM and 2 vCPUs (Virtual Central Processing Unit). One vCPU is roughly equivalent to a single physical core on the computer's CPU (Youngjin, et al., 2011). The *Windows Server 2019* server VM is assigned 40 GB of storage, 2 GB of RAM and 2 vCPUs. The requirement for more storage for the server is mandated by *Microsoft* as it contains many more system tools, configurations, an AD DS (Active Directory Domain Service), DNS (Domain Name System) and DHCP (Dynamic Host Configuration Protocol) servers, and more storage may be needed if the server will be used to host files or act as a web server. The *Pfsense* firewall VM requires the least amount of system resources and as such only consumes 8 GB of storage, 1 GB of RAM and 1 vCPU. The firewall has no GUI and is command-line only therefore can run on the minimum amount of resources that it requires. Finally, the *Ubuntu Linux* VM uses 10 GB of storage, 2 GB of RAM and 2 vCPUs.

After the provisioning of the *ESXi* hypervisor within *VMware Workstation 15*, a crucial command is required in order to not provision too much disk space to the hypervisor. During first boot of the VM, holding down 'Shift + O' will bring up a console prompt, entering the value 'autoPartitionOSDataSize=23841' will provision 25 GB (equivalent to 23,841 Mebibyte) of disk space for the hypervisor while the remainder of the 117 GB of storage can be used for the ISO files used for VM creation as well as the VM disk storage themselves where the OS will reside. If this command is not used, then the hypervisor will consume the majority of the storage provisioned to it in *Workstation* and there will be barely any left-over disk space for the individual VMs to use for provisioning.

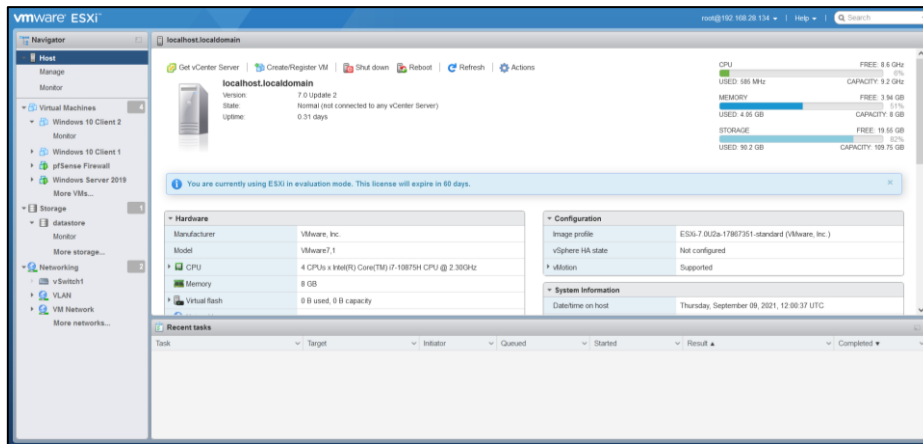


Figure 6 – Main screen of the VMware ESXi web portal

A new VLAN (Virtual Local Area Network) is created within the *ESXi* environment in order to contain the VMs within a single network. The *Pfsence* firewall is connected to both the new VLAN which is named just ‘VLAN’ and the pre-configured *VMware* VLAN which is known as ‘VM Network’, this allows the server to talk to all local clients that are connected to it as well as accessing the Internet via the firewall. Once all the VMs have had their OSs installed it is now required that all the VMs are turned on and are running simultaneously.

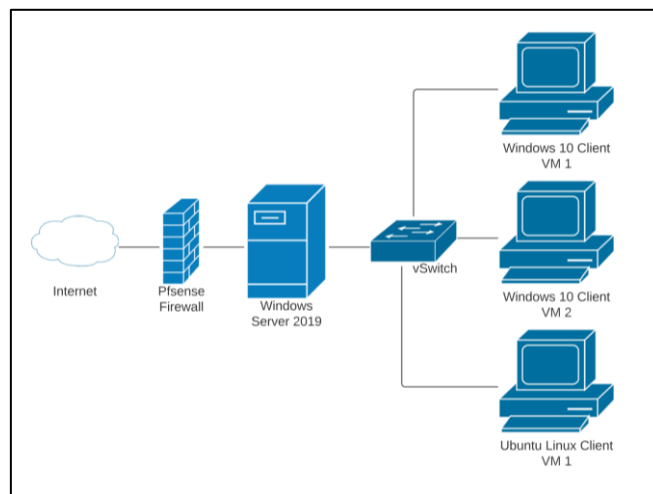
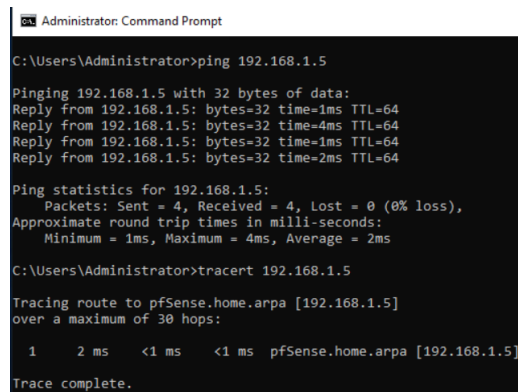


Figure 7 – VMware ESXi deliverable testing environment network diagram

4.1.2 Testing Configuration

In order to verify that the VMs can communicate with each other, it is essential that the configuration of the hypervisor is tested by using simple network tests. It is possible to verify an established connection between two hosts by using an ICMP (Internet Control Message Protocol) echo request, or more commonly known as *ping*, between the two clients (Kahraman & Kocak, 2016). This method should be tested on

each VM to verify that they can all communicate and route to each other via the *Windows Server 2019*.



```
Administrator: Command Prompt
C:\Users\Administrator>ping 192.168.1.5
Pinging 192.168.1.5 with 32 bytes of data:
Reply from 192.168.1.5: bytes=32 time=1ms TTL=64
Reply from 192.168.1.5: bytes=32 time=4ms TTL=64
Reply from 192.168.1.5: bytes=32 time=1ms TTL=64
Reply from 192.168.1.5: bytes=32 time=2ms TTL=64

Ping statistics for 192.168.1.5:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 4ms, Average = 2ms

C:\Users\Administrator>tracert 192.168.1.5
Tracing route to pfSense.home.arpa [192.168.1.5]
over a maximum of 30 hops:
  0  2 ms  <1 ms  <1 ms  pfSense.home.arpa [192.168.1.5]
Trace complete.
```

Figure 8 – Pinging and tracert the Pfsense firewall from the *Windows Server 2019*

The *ping* tool sends and echo request from one host to another with the receiving host responding immediately with an echo reply once it has received the ICMP packet. If the host is unreachable then the ICMP packets will be dropped, and no echo reply will be received by the host initiating the *ping*. It is also useful to use *tracert* to verify the path and hops made in order to reach the destination IP address. This step is done with every VM on the network to verify the routes that they take are according to the networking diagram in *figure 7*.

The AD DS on the *Windows Server 2019* also contains 3,000 users that have been created using a fake name generator and custom *PowerShell* script. The CSV (Comma-Separated Values) file that was generated has been attached as *Appendix A*. The CSV file contains the following fields so that the AD DS can add those users to the directory for use across the network computers: GivenName, Surname, StreetAddress, City, Title, Country, and TelephoneNumber. Every user is given the password ‘Password1!’ and are prompted and forced to change the password upon first login. These users have been added to different security groups in order to emulate a company’s infrastructure with some of these users also having administrative privileges. Testing a sample of users to verify if they can log in on the two separate *Windows 10* client VMs is also past of the testing stage to determine if those devices are indeed establishing a connection with the AD DS hosted on the *Windows Server*. The *PowerShell* script that was used to add the users to the AD DS is shown in *figure 7* which includes all minimum necessary variables required by the server to register a new user.

```
Import-module activedirectory
$UserList = Import-Csv -Path 'C:\Users\Administrator\Downloads\FakeNameGenerator.csv'
foreach ($User in $UserList) {
    $Values = @{}
    Enabled = $true
    #Password must be changed as soon as the user logs in for the first time
    ChangePasswordAtLogon = $true
    Path = "OU=$(($User.Country),OU=Users,OU=LAB-ENVIRONMENT,DC=southwales,DC=ac,DC=uk"
    #Sets the user's names and username for logging in
    Name = "$($User.GivenName) $($User.Surname)"
    UserPrincipalName = "$($User.GivenName).$($User.Surname)@southwales.ac.uk"
    Email = "$($User.GivenName).$($User.Surname)@southwales.ac.uk"
    SamAccountName = "$($User.GivenName).$($User.Surname)"
    Title = $User.Title
    GivenName = $User.GivenName
    Surname = $User.Surname
    #Sets the City, Occupation, Country and phone number of the user
    City = $User.City
    StreetAddress = $User.StreetAddress
    OfficePhone = $User.TelephoneNumber
    Country = $User.Country
    #Password is set to 'Password1!' for all users until login where they have to change it
    AccountPassword = "Password1!" | ConvertTo-SecureString -AsPlainText -Force
}
New-ADUser @Values
```

Figure 9 – PowerShell script used to add CSV generated users to AD DS

4.1.3 Pre-installing required applications

In order for the deliverable to run on the server, it is required that *Sysmon* as well as *EvtxExplorer* be installed on the system. *EvtxExplorer* is an open-source tool developed by Eric Zimmerman that parses event log files (evtx) to a standardised CSV output². This CSV file will be used for the anomaly detection stage of the application, which requires a CSV file as input for the ML.NET spike detection.

Once *Sysmon* has been downloaded and extracted from the *Microsoft* website, it is installed using the following command shown in *figure 10* which also starts the *Sysmon* service immediately in the background. The *Sysmon* service will automatically run when the system starts, however, as this is on the server it will be on continuously due to the nature of servers rarely shutting down besides for maintenance.

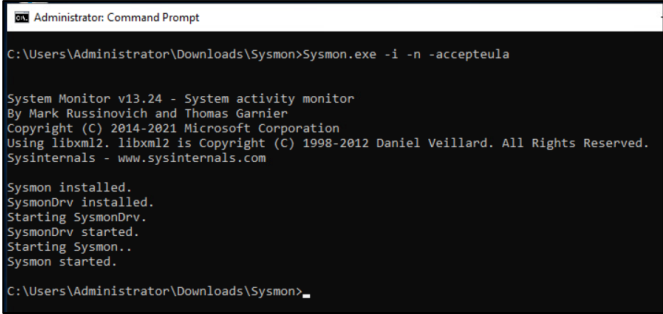


Figure 10 – Installing Sysmon on the Windows Server 2019 VM

² EvtxExplorer available to download at: <https://github.com/EricZimmerman/evtx>

The *EvtxExplorer* application does not need installing and is used as a standalone executable. A detailed explanation of how this application is used in conjunction with the deliverable is illustrated in chapter 4.2.3.3 of this dissertation.

4.2 C# .NET Threat Hunting Application

The C# application consists of different features that will assist a threat hunter in their role of carrying out a hunt on event logs. The core foundation of the application mainly features the event viewer with basic event type, source, and instance ID filtering capabilities. The hypothesis is that this event log viewer will be faster than the standalone *Windows Event Viewer* used currently as it does not need to load in all logs that may be unnecessary for a threat hunter. The secondary functionality of the application includes the machine learning aspect where the filtering can be used against a machine learning model to detect any anomalies that may occur in the logs of the *Sysmon* event logs in particular. Any anomalies detected by the machine learning algorithm will be highlighted to the user for further analysis.

4.2.1 Initial Layout and Design

The C# application consists of a basic *Winforms* GUI that features a menu bar, tool strip with various options, the main event log data grid view and a progress bar anchored towards the bottom of the form.

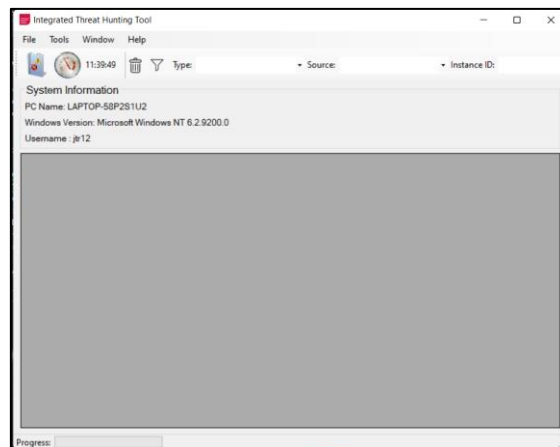


Figure 11 – Basic GUI layout for the primary WinForms deliverable application

The user can access the *Windows Event Viewer* and *Performance Monitor* directly from the main application as well as being able to filter based on type, source, and instance ID of the event they are trying to query. The event logs for that particular query will be displayed in a *DataGridView* in the lower three-quarters of the UI. Double-clicking on cells within the UI allows the user to view more information as well as auto-populating

the filter boxes in the tool strip for ease of convenience for the next filter they may want to conduct.

The second form within this application is the machine learning form. This form will be used in order to gather the *Sysmon* CSV file created using the *EvtxExplorer* application on the target infrastructure while processing and analysing it using the spike detection time-series anomaly detection algorithm. The results of the analysis are displayed in a similar format to the event logs retrieved in the main form, however, a final column ‘Anomaly Detected?’ will denote whether the algorithm believes an anomaly exists for that particular log or not.

4.2.2 Visual Studio and GitHub Setup

The *Visual Studio* version used for this application is *Visual Studio Community 2019* which includes the necessary tools to create an interactive GUI with the backend code to conduct the event log collection and analysis. For this application to work, there are a few libraries that need installing. The installation of the libraries can be done via the in-build *NuGet Package Manager* which keeps track of the versions and whether any updates are required to keep the application safe and up to date. The *NuGet* libraries required for the machine learning element of this application are:

- Microsoft.ML
- Microsoft.ML.CpuMath
- Microsoft.ML.DataView
- Microsoft.ML.Mkl.Redist
- Microsoft.ML.TimeSeries

The aforementioned frameworks contain the necessary classes, components and dependencies needed for the machine learning algorithms to work. The *Microsoft.ML.TimeSeries* specifically contains the algorithms for the anomaly detection whilst the other packages contain the core functionality of machine learning.

A *GitHub* repository, which is used for software development and version control, was also set up for this project in order to track changes and issues with ease. The repository allows for recovery of files in case of any bugs that may occur further down the line and keeps a record of the progress made. Not all saves are uploaded to the *GitHub* repository, but significant milestones are saved locally and are then committed and pushed to the repository. By using *GitHub*, it also allows for the source code to be shared with the public and is downloadable by anyone who has the correct access.

4.2.3 Core Functionality

This section is split into three parts in which the application functionality is outlined: the creation of the *Windows Event Logs* filterer and viewer, the addition of the *Sysmon* logs being collected to the filterer and viewer, and finally the implementation of the machine learning algorithm in the second form using *Sysmon*. Each section will cover aspects of the design and development of these core functionalities; however, this section is not exhaustive to the whole application. The source files for the application are available on *GitHub* for further analysis of the C# code which includes detailed comments for each functionality. The link for the source code is available at *Appendix B*.

4.2.3.1 Processing of Windows Event Logs

In order for the application to collect logs, it is required that the threat hunter use the filters provided in the tool strip to narrow down the search criteria. The reason these filters were chosen in this way is that they only allow the logs that the threat hunter requires to be loaded instead of loading all possible, often unnecessary, logs specifically for the hunt. The user can select the type of log: Application, System, Security and Sysmon, which will then auto populate the ‘Source’ filter based on the sources that those event logs are attached to.

The application iterates through the event logs of that type and will gather the sources and add them to the ‘Source’ filter upon the ‘Type’ filter being closed. This process takes a few seconds as it is an iterative step. There is no need for the user to use a source, however, this will narrow down the query and produce results faster and more focused. Finally, the last filter ‘Instance ID’ takes a numerical value which correlates to the event ID of an event log. This ‘Instance ID’ filter provides a more granular search to the query which in total provides three ways of retrieving event logs. As with the source filter, there is no requirement to set an instance ID for a query to take place.

Once the values for the filters have been set, the user can then press the ‘filter’ button to the right of the ‘bin’ button that will gather the latest event logs of that type, as well as source or instance ID if they have been set. This process takes a few moments and may take up to and over a minute in cases where there are many thousands of logs being queried. An example of the search process and subsequent populated results are shown in *figure 12*.

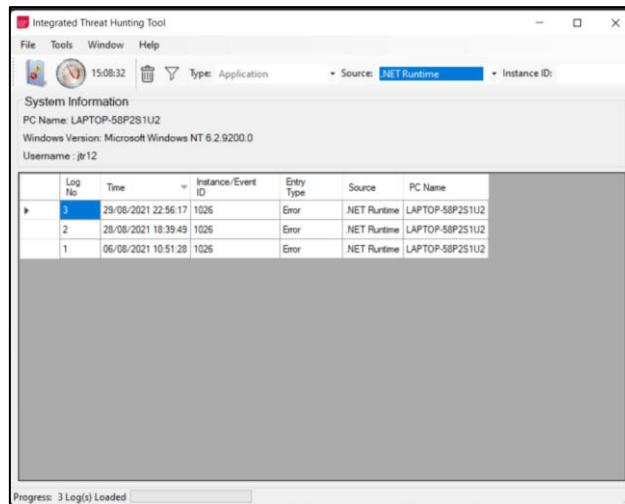


Figure 12 – Deliverable application showing event logs that have been filtered

Within the DataGridView, where the event log results are shown, it is possible to click on the cells for further functionality. Double-clicking on the ‘Log No’ will display further information about the log which cannot be stored within the table due to its length, while double-clicking on an ‘Instance/Event ID’ or ‘Source’ cell will auto populate the filter textbox with its value for the next search respectively. An example of the detailed information shown to the user after double-clicking a ‘Log No’ cell for a particular event log is outlined in *figure 13*.

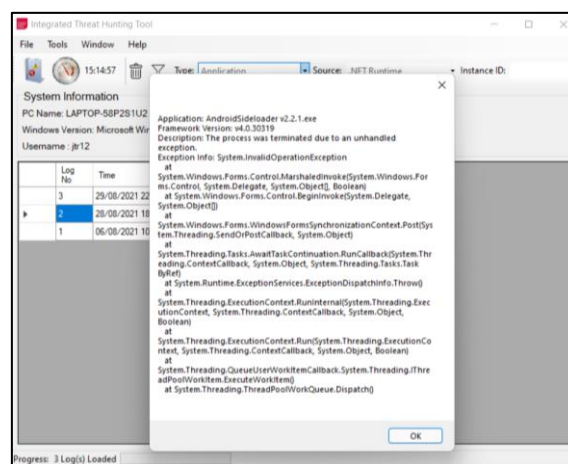


Figure 13 – Deliverable application showing more information for the specified event log

This simplified approach to filtering based on three variables provides new users and novices to threat hunting with a great opportunity to learn and discover the art of the field without being overwhelmed with logs and an array of different options in tools such as the *Windows Event Viewer*, which hasn’t had any considerable upgrades, especially to its UI and UX (User Experience) since *Windows Vista* (Petri, 2009).

As with the *Windows* logs, it is also possible for the threat hunter to double-click on the cell of the ‘Log No’ for *Sysmon* logs and retrieve further information regarding that particular event. All the log messages for each filter iteration and stored in a List<string> using and are then accessed when the user double-clicks on the cell as highlighted in *figure 16*.

The pop-up message which retrieves the string from the *logMessage* list contains information such as what sort of event is it (such as process creation, network creation, process access, driver loaded etc.), the time the event was logged, the unique GUID of the process, name of the file name executed, hashes as well as a description among other interesting features valuable to a threat hunter.

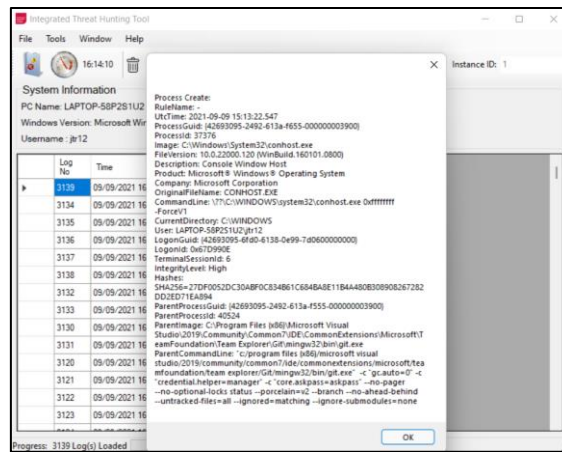


Figure 16 – Sysmon log showing more event information when double-clicked

The logs upon being filtered to the DataGridView are automatically hardcoded to be sorted by the time stamp hence, in *figure 16*, the last log number is shown first as it was the latest *Sysmon* event log of event ID 1 – Process Creation to be logged to the .evtx file.

4.2.3.3 Sysmon Machine Learning Anomaly Detection

The final section of the deliverable application is the use of anomaly detection to identify anomalous activity within the *Sysmon* events. The GUI of this form follows a very similar design as the event log retriever form covered in sections 4.2.3.1 and 4.2.3.2, however, there are notable features that are missing or have been added. Firstly, this window will only allow for the collection and analysis of *Sysmon* logs for this proof-of-concept. Secondly, the user must supply the CSV file using an external open-source tool. Thirdly, the filtering textboxes have been removed as they are not applicable for this section of the application.

The reason why this proof-of-concept deliverable has solely concentrated on the use of *Sysmon* logs for analysis is that allowing all logs to be analysed for a proof-of-concept would be exhaustive and would take a long time to implement correctly without any mistakes. The *Sysmon* logs only consist of twenty-six different events that can ever be logged, making the collection and analysis process much more straightforward. It is important to note however that although *Sysmon* only collects twenty-six events, it provides a lot of important information, especially for anomalous activity detection (Garnier & Russinovich, 2021). The majority of the time however, the most common logs that are collected for *Sysmon* are event IDs 1 – Process Creation, 3 – Network Connection and 5 – Process Terminated with the occasional error with event ID 5.

The threat hunter is required to use the previously mentioned tool, *EvtxExplorer*, to parse the .evtX event log files to the readable CSV format. The anomaly spike detection within ML.NET requires that the format be CSV and that there be two columns. This is part of the data processing stage required before using any machine learning algorithm. It is important that data is sanitised and of the correct format, this can be done by hand or within the program. Once the user presses the tool strip CSV button, as shown in *figure 17*, the application will run a sequence of events to collect, analyse and print the results, including any anomalous results, to the DataGridView.

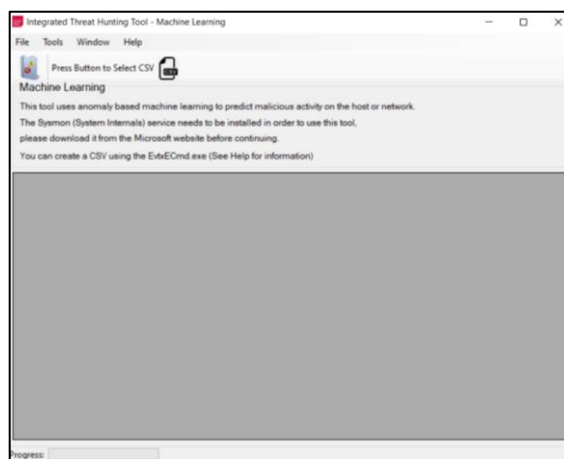


Figure 17 – GUI layout for the machine learning window of the WinForms deliverable application

The threat hunter can create a CSV of the *Sysmon* event logs by going to the directory of the *EvtxExplorer* executable and executing the following command in *figure 18*. This command was executed on the server in order to create the CSV file necessary for the deliverable application to process where -f is equal to the location of the *Sysmon*

.evtx log file, --csv is equal to the full path of the location to save and --csvf is equal to the name of the CSV file to save (Elshaer, 2020).

```
C:\Users\Administrator\Downloads\Code\EvtxExplorer>.EvtxECmd.exe -f "C:\Windows\System32\winevt\Logs\Microsoft-Windows-Sysmon%4Operational.evtx" --csv C:\Users\Administrator\Downloads\Code\EvtxExplorer\CSVs\ --csvf sysmon.csv
EvtxECmd version 0.6.5.0
Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/evtX
Command line: -f C:\Windows\System32\winevt\Logs\Microsoft-Windows-Sysmon%4Operational.evtx --csv C:\Users\Administrator\Downloads\Code\EvtxExplorer\CSVs\ --csvf sysmon.csv
```

Figure 18 – Generating a CSV file on the server using EvtxExplorer

The user can use this newly created and up-to-date CSV file to pass it into the application by pressing on the CSV button on the machine learning form. The user will be prompted to select the CSV file with an OpenFileDialog which will store the CSV contents in a variable called *csvFileName*.

Log No	Time	Alert	Score (Event ID)	P-Value	Anomaly Detected?
19289	2021-09-09 09:40:22.1139974	0	3	0.499707382607568	
19290	2021-09-09 09:40:22.1140343	0	3	0.499707382607568	
19291	2021-09-09 09:40:22.1140534	0	3	0.499707382607568	
19292	2021-09-09 09:40:23.1678439	0	3	0.499707382607568	
19293	2021-09-09 09:40:23.1679340	0	3	0.499707382607568	
19294	2021-09-09 09:40:23.9549655	1	1	0.0406093821182925	<- ANOMALY DETECTED (Spike - This log may be a cause for further investigation)
19295	2021-09-09 09:40:24.2132120	0	3	0.499707382607568	
19296	2021-09-09 09:40:24.2132892	0	3	0.499733980777015	
19297	2021-09-09 09:40:24.2133526	0	3	0.499707375181453	
19298	2021-09-09 09:40:24.2133764	0	3	0.499680768912684	
19299	2021-09-09 09:40:24.2134057	0	3	0.499680768912684	
19300	2021-09-09 09:40:24.2134464	0	3	0.499680768912684	
19301	2021-09-09 09:40:24.2134912	0	3	0.499707367765761	
19302	2021-09-09 09:40:24.2135319	0	3	0.49973396729392	

Figure 19 – Populated anomaly analysis results with Sysmon spike detection

The ‘Anomaly Detected?’ column will highlight if there is an anomaly detected by the algorithm, however, this is not to say that there is any malicious activity taking place, just that the logged event has deviated from the norm and required further human analysis. In order to use the *Microsoft.ML.TimeSeries* algorithms, in particular, Spike Detection, it is required that two classes are created: *SysmonData* and *SysmonPrediction*. Spike Detection requires a time series input data class to store the values of the CSV to be analysed (Alexander, et al., 2020). Within the *SysmonData* class are two properties: *public string TimeCreated*, and *public float EventID*. This algorithm only takes one input, which in this case will be the Event ID. This input is contrasted against the time stamp to predict anomalies. No other variables are used in calculating the anomaly, and as such,

this is a fairly straightforward algorithm containing only two features. Data preprocessing also occurs here, where the program will only select those two columns values and will convert the integer to a float, which is needed for this particular algorithm.

The second class, *SysmonPrediction*, contains a double array that stores the prediction results from the algorithm later as an object for adding to the *DataGridView* for viewing. Every machine learning project using ML.NET requires something which is known as an *MLContext*, which is a class that once instantiated allows for data preparation, feature engineering, training, prediction, analysis, and evaluation of the machine learning model among other things within its own environment (Microsoft, 2021). The data that is retrieved from the CSV file is stored and loaded as an *IDataView* interface, which is an efficient method of portraying tabular data for the algorithm to use.

Once the data has been stored in the interface, the application creates an empty *DataView* alongside the one which has the data, as it is used to create the schema for the model fitting. Model fitting is a measure used to determine how well a machine model matches real-world data (Khan & Jabbar, 2015). With all of the preparation mainly done to the variables and interfaces, the *DetectSpike* method is created which uses the aforementioned data to train the model on the same data that will be analysed. A transform is then made, which makes the CSV data useable and useful by joining it together, making it dimensionally modelled and de-normalised before prediction (Chan, 2019). This transformed data is then used to create a strongly typed *IEnumerable* called *predictions* that will store the results. The results stored in *predictions* includes the alert, score/Event ID and the P-Value. The alert is used to indicate whether a spike is at this event log, and is denoted by a 1 or 0, 1 being a spike. The score/Event ID will display the numerical value of the event ID while the P-Value will display the probability of an anomaly. The closer that this value is to 1 the likelihood that the entry is not an anomaly, where an entry nearer to 0 would indicate a higher probability of being an anomaly. The p-value is calculated by using the following equation:

$$pValue = 1 - \frac{confidence}{100}$$

Equation 1 - pValue prediction of an anomaly

Confidence in this equation is defined as the signature produced by the spike estimator that predicts the anomaly spikes. In this program, the confidence has been set

to 95, which means that the model requires a high degree of confidence before deciding and making the call if the event log is a spiked anomaly or not. Lowering this value would produce more spikes, therefore, it is good to tune the value depending on how many anomalies are being detected, that are known not to be anomalies.

Once the prediction has been made, the results are iterated through and are displayed to the DataGridView. The code for doing this is shown in *figure 20* below.

```

IDataView transformedData = iidSpikeTransform.Transform(sysmonResults);
MessageBox.Show("Predicting data");
var predictions = mlContext.Data.CreateEnumerable<SysmonPrediction>(transformedData, reuseRowObject: false);
toolStripProgressBar.Minimum = 1;
toolStripProgressBar.Maximum = docSize;
toolStripProgressBar.Step = 1;
//Create Table
DataTable table = new DataTable();
table.Columns.Add("Log No", typeof(int));
table.Columns.Add("Time", typeof(string));
table.Columns.Add("Alert", typeof(string));
table.Columns.Add("Score (Event ID)", typeof(string));
table.Columns.Add("P-Value", typeof(string));
table.Columns.Add("Anomaly Detected?", typeof(string));
int logNumber = 1;
int noOfAnomalies = 0;
foreach (var p in predictions)
{
    toolStripProgressBar.PerformStep();
    if (p.Prediction[0] == 1)
    {
        table.Rows.Add(logNumber, timeColumn[logNumber - 1], p.Prediction[0], p.Prediction[1], p.Prediction[2], " <-- ANOMALY");
        noOfAnomalies++;
    }
    else
    {
        table.Rows.Add(logNumber, timeColumn[logNumber - 1], p.Prediction[0], p.Prediction[1], p.Prediction[2], "");
    }
    logNumber++;
}
dataGridView1.DataSource = table;
MessageBox.Show("The algorithm found " + noOfAnomalies + " possible anomalies.", "Anomalies found", MessageBoxButtons.OK, Mess

```

Figure 20 - Iterating through predicted results and displaying logs with anomalies to DataGridView

5 Analysis

There are a few tests that will be carried out in order to verify the effectiveness of the deliverable. The first test is to check the speed of the deliverable in retrieving logs from the event log and comparing its time with the native *Windows Event Viewer*.

The first test will run five times, opening *Windows Event Viewer*, searching for *Sysmon*, sorting by Event ID and then repeating by closing the application and trying again. Then the deliverable application will do the exact same test. This test will measure the efficiency of lowering the dwell time, that is, which application can load the logs the fastest for the quickest analysis by a human threat hunter.

Running *Windows Event Viewer* five times and opening *Sysmon* via the folder tree takes around 2s to filter from lowest to highest Event ID on average. The same test with the deliverable takes around 5s to filter those logs based on lowest to highest Event ID on average. There is a slightly disappointing result in that the deliverable application takes longer than the *Windows Event Viewer* to load *Sysmon* results, however, this is not the

case for *Application, System and Security* event logs where the loading time is around the same or even faster in some cases. This is also dependent on the resources being currently consumed by the computer doing the task. This could also be due to the fact that the deliverable has not been compiled fully into an executable prior to testing. The same logs are loaded by both the *Windows Event Viewer* and the deliverable. The deliverable requires a single double-click to view all the log's descriptions and messages, whereas the standard *Windows Event Viewer* requires the user to scroll down to view more details about the log in question.

Windows Event Viewer does not include an anomaly detector, and as such, cannot be compared to the system build in the deliverable to detect *Sysmon* anomalies. The user using the conventional approach would have to scroll through all the logs and determine whether an anomaly may be taking place, this increases the dwell time and the time the attacker has to carry out their APT and attacks. The deliverable highlights which logs have an anomaly, and can be incorporated with the *Windows Event Viewer* to verify those specific logs with more detailed information.

Due to the very nature of this novel *Sysmon* log analyser using anomaly spike detection, there is no way to compare it against an existing tool that is available to the public, as such a tool does not exist in the current domain for *Sysmon* logs in particular. However, ELK using Ubuntu was also used to determine how practical it is to analyse *Sysmon* logs versus the anomaly detection provided by the deliverable. ELK is a logging tool for search and analysis which can ingest data from multiple sources.

The advantage of this deliverable is that it can accompany both *Windows Event Viewer*, ELK and other SIEMs to enhance the intelligence capabilities of a new threat hunter who has entered the field. The tool can be used as the first stop of a training threat hunter to understand logs, how they work, how they are sequenced and if there are any anomalies that could be present. ELK is incredibly complicated and contains many configurations, which is mainly used for large infrastructures and networks. ELK would be an overkill for many threat hunts where the network environments only contain a few computers.

6 Evaluation and Conclusion

The event log collector and analyser created in this dissertation is a great tool for newcomers to cyber threat hunting to understand the basic ideas and principles behind the techniques used for log collection and analysis. The tool provides novices to threat hunting with a way to analyse these logs in a visually appealing format without too much distraction from excessive options and numerous other miscellaneous logs that have no bearing on threat hunting. This application acts as a first-step for new threat hunters and can be used for training new SOC team members or those completely new to cyber or computing. The application that has been created limits the logs that are collected to the 3 core *Windows* logs as well as the custom *Sysmon* logs. These are the 4 main log sources that a threat hunter should be utilising and having them all in one place for quick analysis without having to worry about complicated filtering is a great tool for anyone starting off in the field. The usage of anomaly detection for determining spikes of anomalous logs in the *Sysmon* event logs is a powerful feature that will lower the dwell time of threat hunts by pointing threat hunters in the right direction towards which logs should be analysed and given priority before others.

With every project, however, there are limitations and lessons learned. The main limitation with this project is time. The time constraints put on this project do not allow for it to grow to its full potential if it had much longer to develop. The *Sysmon* anomaly detection uses a basic machine learning algorithm only and will not notify if a log is malicious or not, only anomalous. This is very useful, however, a longer development time could allow for more advanced machine algorithms to be utilised and adapted. It could also be possible for deep learning algorithms to play a substantial role in this application. Due to the very nature of machine learning needing large datasets to work at its most effective, another limitation with this approach is that the computing power to analyse in real-time the *Sysmon* logs of a larger environment from a commercial computer sold to the public is limiting and would require immense deep learning computational power the likes of only large corporations have. This application will continue to be developed on with time, more features will be added, more analysis will be made on non-*Sysmon* logs with different machine learning techniques and hopefully with more time this application can become a practical learning and instructional tool for teaching the techniques of threat hunting to a new audience with little experience.

7 LSEPI

For every research project, the **Legal, Social, Ethical and Professional Issues** (LSEPI) need to be discussed and addressed prior to conducting any studies. In some cases, there may not be any issues whatsoever, however, it is vital that all consideration is given in determining whether there are any LSEPI problems that may arise that need managing, no matter how small they are. The next 4 subchapters will discuss what issues this project will be dealing with and how those issues can be alleviated.

7.1 Legal

There are some legal aspects regarding this research project that needs addressing. In the UK, computer professionals are bound by legislation, guidelines and standards which must be adhered to. This research project will mainly need to comply with the *Computer Misuse Act 1990* (CMA 1990), *Data Protection Act 2018* (DPA 2018) and the *General Data Protection Regulation* (GDPR). Besides legislation, frameworks and codes of conduct of the British Computer Society (BCS) must also be followed, however, this can also be considered as part of the Professional aspect of LSEPI.

This project requires that data be used to test and analyse the effectiveness of the threat hunting programme that will be created against data logs and events. The data collected will be subject to the DPA 2018 as it may contain sensitive information such as names, addresses, websites etc, however, utmost attention has been taken to ensure that all data is pseudonymised before use in the project. Open-source datasets are used by threat hunters in order to simulate hunts and are therefore pseudonymised out-of-the-box. This project will also explore creating personalised threats and attack data logs and events using a virtual machine (VM) over a virtual network. This virtual network will be locked down from the internet and external network and will only be used for carrying out attack simulations in order to test and evaluate the program deliverable.

The CMA is also valid in this project as there must not be any unauthorised access, unauthorised access with intent to commit further crime or unauthorised modification to computers or networks during this research (Computer Misuse Act, 1990). There will be no unauthorised access or modification to any computer or network besides the modification conducted within the VMs themselves, which will be disconnected from the internet and cannot inadvertently harm other network users. The utmost care must be taken to ensure that the necessary legislation is adhered to, however, the case of breaking

any legislation in this research project is minimal due to the closed nature of the work which won't be exposed to anybody besides those working closely on the project.

While writing the report as well as conducting research, all care must be taken to ensure that all sources are referenced correctly. This is to prevent any accusations of plagiarism as well as adhering to the *Copyright, Designs and Patents Act 1988* while using other organisations or peoples work.

7.2 Social

Due to the nature of this project, there will be minimal social interaction compared to other projects i.e. there will be no surveys nor questionnaires collected with personal information.

7.3 Ethical

During this research, there must be no bias presented towards any authors, researchers, lecturers or organisations that have produced articles, journals or research papers. All material and sources must be treated fairly and critically analysed to remove any underlying bias from them. As mentioned above, data will be anonymised during collection and storage and will not be shared with third parties without prior consent by said individual.

7.4 Professional

The research project will adhere to the guidelines and codes of ethics, practice and conduct set out by the BCS and other relevant bodies.

8 References

- Alexander, J. et al., 2020. *Tutorial: Detect anomalies in product sales with ML.NET*. Available at: <https://docs.microsoft.com/en-us/dotnet/machine-learning/tutorials/sales-anomaly-detection> (Accessed: 29 July 2021)
- Betz, C., Caltagirone, S. & Pendergast, A., 2013. *The Diamond Model of Intrusion Analysis*, Hanover (MD): Center For Cyber Intelligence Analysis and Threat Research.
- Bridge, K., Sharkey, K. & Satran, M., 2018. *Event Sources*. Available at: <https://docs.microsoft.com/en-us/windows/win32/eventlog/event-sources> (Accessed: 23 August 2021)
- Bridge, K., Sharkey, K. & Satran, M., 2018. *Event Types*. Available at: <https://docs.microsoft.com/en-us/windows/win32/eventlog/event-types> (Accessed: 11 August 2021)
- Chan, D., 2019. *Why You Need Data Transformation in Machine Learning*. Available at: <https://www.datanami.com/2019/11/08/why-you-need-data-transformation-in-machine-learning/> (Accessed: 22 August 2021)
- Christou, E., Spring, J. M. & Al-Shaer, R., 2020. *Learning the Associations of MITRE ATT & CK Adversarial Techniques*. Avignon, IEEE, pp. 1-9.
- Computer Misuse Act, 1990. Available at: <http://www.legislation.gov.uk/ukpga/1990/18/contents>
- Elshaer, A., 2020. *Introduction to EvtxEcmd (Evtx Explorer)*. Available at: <https://isc.sans.edu/forums/diary/25858> (Accessed: 2 August 2021)
- Garnier, T. & Russinovich, M., 2021. *Sysmon v13.22*. Available at: <https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon> (Accessed: 1 July 2021)
- Gartner, Inc., 2014. *Threat Intelligence: What is it, and How Can it Protect You from Today's Advanced Cyber-Attacks?*, Broomfield (CO): Webroot.
- Hutchins, E. M., Cloppert, J. M. & Amin, R. M., 2011. *Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chain*, Bethesda (MD): Lockheed Martin Corp..
- Jøsang, A. & Mavroeidis, V., 2018. *Data-Driven Threat Hunting Using Sysmon*. Guiyang, ICCSP, pp. 82-88.
- Kahraman, Z. & Kocak, C., 2016. *Performance analysis of IP network using two-way active measurement protocol (TWAMP) and comparison with ICMP (ping) protocol in a saturated condition..* Antalya, 4th International Symposium on Innovative Technologies in Engineering and Science (ISITES2016).

- Khan, R. Z. & Jabbar, H., 2015. Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study). *Computer Science, Communication and Instrumentation Devices*, pp. 163-172.
- Landauer, M. et al., 2018. Time Series Analysis: Unsupervised Anomaly Detection Beyond Outlined Detection. *International Conference on Information Security Practice and Experience*, pp. 19-36.
- Lee, R. M. & Lee, R. T., 2018. *Threat Hunting Survey Results*, Bethesda (MD): SANS.
- Lippmann, R. et al., 2006. *Validating and restoring defense in depth using attack graphs*. Washington (D.C.), MILCOM IEEE Military Communications Conference.
- Lord, N., 2020. *What are indicators of compromise*. Available at: <https://digitalguardian.com/blog/what-are-indicators-compromise> (Accessed: 21 July 2021)
- McHugh, J., 2000. ACM Transactions on Information and System Security. *Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory*, 3(4).
- Microsoft, 2007. Troubleshooting. In: *Windows Vista: Beyond the Manual*. New York City (NY): Apress, pp. 439-448.
- Microsoft, 2018. *Event Logging*. Available at: <https://docs.microsoft.com/en-us/windows/win32/eventlog/event-logging> (Accessed: 20 August 2021)
- Microsoft, 2021. *IidSpikeEstimator Class*. Available at: <https://docs.microsoft.com/en-us/dotnet/api/microsoft.ml.transforms.timeseries.iidspikeestimator?view=ml-dotnet> (Accessed: 28 August 2021)
- Microsoft, 2021. *MLContext Class*. Available at: <https://docs.microsoft.com/en-us/dotnet/api/microsoft.ml.mlcontext?view=ml-dotnet> (Accessed: 25 August 2021)
- Mitchell, T. M., 1997. *Machine Learning*. 1st ed. New York: McGraw Hill.
- MITRE, 2021. *ATT&CK Matrix for Enterprise*. Available at: <https://attack.mitre.org/> (Accessed: 30 June 2021)
- Müller, A. C. & Guido, S., 2017. *Introduction to Machine Learning with Python*. Sebastopol(CA): O'Reilly.
- Palacín, V., 2021. *Practical Threat Intelligence and Data-Driven Threat Hunting*. 1st ed. Birmingham: Packt.
- Petri, D., 2009. *Working with Vista's new Event Viewer*. Available at: <https://petri.com/vista-event-viewer> (Accessed: 15 August 2021)

- Robbins, A., Schroeder, W. & Vazarkar, R., 2021.
<https://github.com/BloodHoundAD/BloodHound>.
Available at: <https://github.com/BloodHoundAD/BloodHound>
(Accessed: 2 August 2021)
- Royal Literary Fund, 2019. *What is a Literature Review*.
Available at: <https://www.rlf.org.uk/resources/what-is-a-literature-review/>
- Russinovich, M. & Garnier, T., 2021. *Sysmon v13.24*.
Available at: <https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon>
(Accessed: 28 July 2021)
- SANS, 2021. *A Practical Model for Conducting Cyber Threat Hunting*, Bethesda (MD): SANS.
- Sharafaldin, I., Lashkari, A. H. & Ghorbani, A. A., 2016. *Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization*, New Brunswick: Canadian Institute for Cybersecurity.
- Sqrrl Data, Inc., 2018. *A Framework for Cyber Threat Hunting*, Cambridge (MA): Sqrrl.
- Taschler, S., 2021. *What is Cyber Threat Hunting?*.
Available at: <https://www.crowdstrike.com/cybersecurity-101/threat-hunting/>
(Accessed: 9 August 2021)
- Topi, H. & Brown, C. V., 2005. *IS Management Handbook*. New York: Taylor & Francis.
- Xiong, W., Legrand, E., Åberg, O. & Lagerström, R., 2021. Cyber security threat modeling based on the MITRE Enterprise ATT&CK Matrix. *Software and Systems Modeling*, pp. 1-21.
- Youngjin, K., Kim, C., Maeng, S. & Huh, J., 2011. *Virtualizing performance asymmetric multi-core systems*. San Jose (CA), IEEE.

9 Appendices

Appendix A – AD user creation CSV file (*FakeNameGenerator.csv*)

Appendix B – *GitHub* source code (<https://github.com/joshua-richards/Integrated-Threat-Hunting-Tool>)