JYX
JYVÄSKYLÄN YLIOPISTO
UNIVERSITY OF JYVÄSKYLÄ

This is a self-archived version of an original article. This version may differ from the original in pagination and typographic details.

Author(s): Myllyla, Juuso; Costin, Andrei

Title: Reducing the Time to Detect Cyber Attacks : Combining Attack Simulation With Detection Logic

Year: 2021

Version: Published version

Copyright: © The Authors 2021

Rights: CC BY-ND 4.0

Rights url: https://creativecommons.org/licenses/by-nd/4.0/

Please cite the original version:

Myllyla, J., & Costin, A. (2021). Reducing the Time to Detect Cyber Attacks : Combining Attack Simulation With Detection Logic. In S. Balandin, Y. Koucheryavy, & T. Tyutina (Eds.), FRUCT '29 : Proceedings of the 29th Conference of Open Innovations Association FRUCT (pp. 465-474). FRUCT Oy. Proceedings of Conference of Open Innovations Association FRUCT. https://fruct.org/publications/acm29/files/Myl.pdf

Reducing the Time to Detect Cyber Attacks - Combining Attack Simulation With Detection Logic

Juuso Myllylä, Andrei Costin
University of Jyväskylä
Finland
juuso.p.myllyla@student.jyu.fi, ancostin@jyu.fi

***Abstract*—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 operator. This raises a serious question, how did the attacks go undetected? The conducted research focuses on the most common pitfalls regarding late or even non-existent detection by defining the root cause behind the failed detection.**

The main goal of this work is 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.

I. 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 towards the product which then analyses the logs and creates automated alerts for the defenders. This is not effective, since the capability of threat detection lies in the hands of the vendor, not the defenders. By empowering the defenders with the right knowledge and tools, the defenders achieve better results and are independent of any vendor providing detection capabilities.

Relying on the defenders expertise in threat detection, the capability of creating novel use cases and correlations will return the investment multiplied. Defenders, who rely on automated detection from a third party often feel powerless compared to defenders, who independently modify and improve the existing detection capabilities and create new ones, generating more involvement for the latter group. All of the simulations are done with open source tools and test environment is also an open source project, 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 creating own detection capabilities without depending on external parties. The study also has empirical elements in the form of attack simulation, when attacks are conducted in the test environment. The log events generated from the simulations are then observed and evaluated, according to the detection framework.

A. Contributions

This research aims to contribute to the information security research field in describing the reasons behind failed threat detection, how threats can be detected and how security professionals can improve their understanding of how simulated attacks can help in understanding and creating more robust detection capabilities.

The findings of the study are targeted towards all the security teams trying to monitor their networks and find threats by introducing a simple framework, which helps eliminate weak detection. The research incorporates log analysis, threat hunting and hints of purple team exercises as methods to understand what the threats are, how different attacks can be seen from the vast amount of log data and how that data can be turned into actionable detection logic, ultimately trying to reduce the dwell time of an attacker being inside the network.

B. Organisation

The rest of this paper is organized as follows. In Section II we present the current work and research being done in regards of threat detection. We present our experimental setup in Section III. After defining the necessary components for meaningful threat detection capabilities, the threat detection framework is presented in Section IV. The last chapter V is focused on how the critical vulnerability Zerologon was exploited in the experimental setup.We conclude with Section VI.

II. Current work and research related to threat detection

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 be similar across different environments, machine learning can be applied by

ISSN 2305-7254

transforming said log into training data sets, where as 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.

A. 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 [1]. 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 [2]. 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 [3].

All the previously listed uses of machine learning were targeted against detection insider threats and compromised accounts, which seemed to be a rising trend from 2015 to 2017.

B. Machine learning in threat detection

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 ultimately confirmed by the analysts [4].

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.

C. 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 ATTCK 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 ATTCK framework requires sophisticated and mature environments and logging policies to detect most of the tactics, techniques, and procedures [5].

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 [6]. Although the publication is 14 years old at the time of authoring the research, most of the concepts 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 [7]. 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 [8].

The two main disciplines, machine learning and practical threat detection will form the scientific basis for this research. As the current trend in threat detection 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 [9].

D. Threat hunting versus threat detection

An important distinction between threat detection and threat hunting must also be made for clarification. The threat detection in this research is focused on generating search queries, which will constantly query the incoming log events in SIEM,

generating alerts for analysts. These search queries are generated with the proposed threat detection framework, producing high quality alerts and most importantly, reducing the amount of low value alerts.

Threat hunting is a proactive discipline, where the analysts form hypotheses and execute manual searches to actively search the logs for signs of potential misuse. A hypothesis is the basis of a threat hunt, and the aim is to prove the hypothesis as false. Such example could be that the monitored environment is known to run SMBv1, which is prone for distributing ransomware. The threat hunter then forms a hypothesis that SMB (Server message block protocol) is used to distribute ransomware. If no signs are found, the hypothesis is proven wrong and the hunt can be concluded and it can be said with certainty that SMBv1 was not used to distribute ransomware, based on the generated queries and their results.

Threat hunting is a newer discipline in cybersecurity field, where the threat detection focus is shifted towards analysts proactively creating potential scenarios and searching for those events. Once the proactive approach generated search query finds misuse, it can be then transformed to the detection baseline using the threat detection framework. While using the threat detection framework, the analyst is actually doing threat hunting by creating hypotheses by identifying the threat. Threat hunting is not however intended to only produce detection rules, but to actively search the environment for signs of compromise.

III. Experimental setup

The detection lab test environment is a very minimal and simplified version of an enterprise environment, 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 / duplicated to match an actual enterprise 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 configured, and the defenders do not need to spend additional time setting up a working Active Directory environment.

The domain controller and the workstation send their event 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 (Security information and event monitoring) system. 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. [10]. The last host is a Linux host running the Splunk instance and other various tools bundled with the detection lab package.

IV. Threat Detection 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. A diagram was created to demonstrate the cyclical approach in threat detection, starting from identifying the threat and ultimately reaching the communication of the results phase, where the detection logic is ready to be shared with the security community.

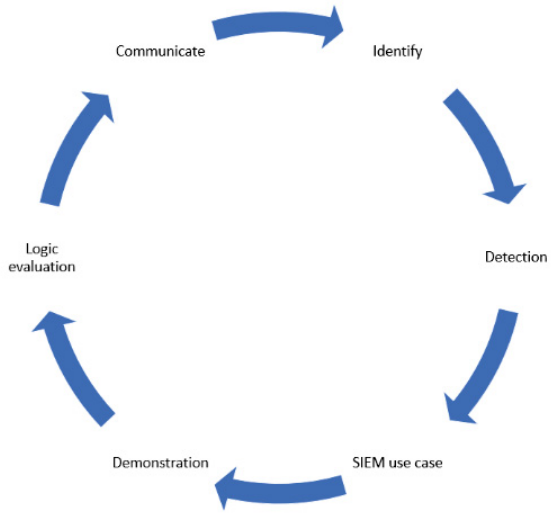

Fig. 1. Threat detection framework and the six phases

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.

A. Threat identification

In the first phase, the threat must be identified and defined, and motivation must be given why this problem needs to be identified. When creating detection logic, the more accurate the threat definition, the less there are false positives and alert fatigue originating from the detection logic [11]. To create an accurate description of the threat, tactics, techniques, and procedures should be considered.

A small thought process can be helpful to demonstrate the logic behind the accurate description and identification of the threat. As an example, the defenders want to detect possible data exfiltration to external destinations. In order to detect this,

the problem needs to be specified using tactics, techniques and procedures. Data exfiltration itself is a tactic, with nine techniques associated to it in the ATTCK framework, making a general detection for data exfiltration nearly impossible. The defender then chooses the most probable technique, data exfiltration to cloud services. Lastly after the defenders have narrowed down the original idea from detecting general data exfiltration to data exfiltration to third part cloud services, which in turn is (usually) large amounts of outgoing traffic from a single endpoint to an external cloud service. Detecting abnormal amounts of outgoing data from a single endpoint to any or specific cloud service is more detailed and specified threat than attempting to create a general rule to detect data exfiltration.

B. Detecting the identified threat

Once the threat has been identified and defined, and the motivation behind it has been stated, the defenders can follow the framework into the next phase, detecting the threat. This is also the second step of the design science research method, defining objectives for a solution [12]. During this threat detection phase, the defenders must now be able to identify the detection capabilities for the threat. The objective is to define and create detection capabilities to detect the threat identified in the previous phase.

When considering the possibilities of detecting the identified threat, the defenders must refer to their logging capabilities and audit policies and determine first and foremost whether the detection is even possible with the current policies and available event logs. The ability to detect threats with the detection logic framework is often related to being able to collect the necessary and relevant logs. As defined earlier in the research regarding threat definitions 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 be detected from firewall traffic generated inside the protected network, such as SMB traffic generated in the local network. Host threats are efficiently 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 [13].

C. SIEM Use case

The third phase of the framework is to create a SIEM use case on how the identified threat can be detected. The third step is also designing and developing the artefact, as stated in the design science research method [12]. 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, using the search query language specific to the SIEM. 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 trailing wildcard searches or simple keyword searches to see what log events are found.

An example of this would be to write a SIEM use case, where the identified threat is the need to detect any use of the popular hacking tool Mimikatz in the environment. As stated in the previous phase, it was determined that the most probable way of detecting Mimikatz use is from the Sysmon event logs, related to new process create events. 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.

D. Detection logic demonstration

Demonstrating the detection logic phase was already mentioned in the previous step, 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 search criteria.

By defining the detection logic clearly, the following step of logic evaluation is more effective, as the defender now begins to investigate 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.

E. Logic evaluation

The logic evaluation phase is heavily involved with the fourth phase, as the results of the detection logic demonstration are evaluated here, as stated in the design science research method. [12]. To evaluate the results, the defender 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 excluding some results that generate lots of false alarms. Iterating between the fourth and fifth phases of the detection logic framework 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 SIEM

use case 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 one 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.

F. 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 [12]. 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 documented in Sigma format, a universal SIEM search query syntax [14]. 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.

V. 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 [15]. 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 [16]. 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 [17].

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 sub-chapter, 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. [12] 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 [18]. 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.

A. 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 [19]. 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. [20]. 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 [21].

The following sub chapters will describe the simulated attacks and most importantly, what log events were left behind and how those were turned into actionable detection logic.

B. Detecting initial compromise and domain enumeration in attack simulation phase one

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 discover 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 [22].

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 most logical method 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 screenshot below:

C. Detecting Zerologon exploitation

After 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 scaling detection logic as possible, understanding the vulnerability and how it was exploited becomes more important. In this research, a total of three detection logic 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 logic 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 [23] 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 [24].

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 controllers machine account password change can be detected from Windows security auditing logs, by monitoring event ID 4742 [25]. Finally, using

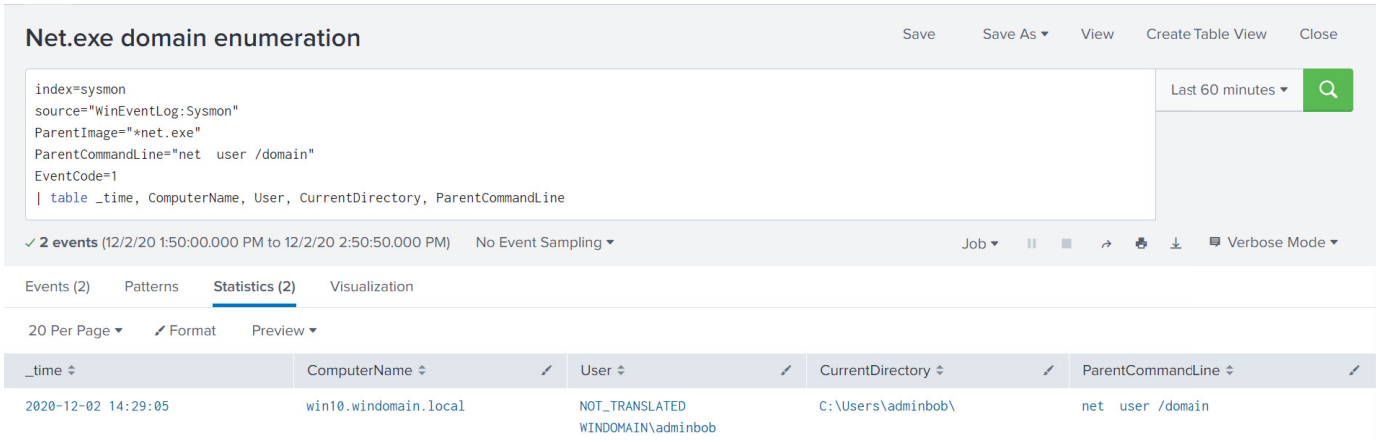

Fig. 2. Example of a SIEM use case - Detecting domain enumeration

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. The captured traffic showed 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* (the computer account of the Domain Controller) 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 based detection logic, like the ones created previously.

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 [26]. 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 computer 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.

The logic of detecting a sudden spike in Netlogon traffic originating from a single host was able to show the source IP address of the attackers machine creating the Netlogon traffic against the domain controller. 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.

The achieved results of the detection logic regarding Mimikatz 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 [27].

D. Detection 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 internal 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,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 [28], 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 [29] due to the sheer amount of log events. The attack originated 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 [30]. 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 “DRSGetNCChanges” 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.

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.

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 [31]. Below is a screenshot of the contents of the ntds.dit file that was used to compromise the whole domain, using the open source tool to extract the contents of the file and providing the attacker the keys to the kingdom.

VI. Conclusion

The main purpose of this work 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 life-cycle was documented and actionable detection logic was created upon the findings. The concrete end results of this work 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

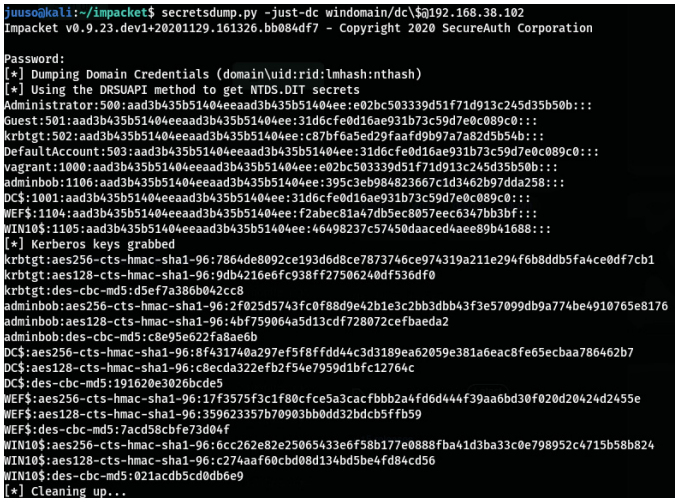

Fig. 3. Obtaining the password hash of the administrator account by exploiting the Zerologon vulnerability

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 life cycles, 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.

VII. Acknowledgements

Authors wish to thank the anonymous reviewers for their valuable comments and feedback that helped improve the quality of the paper.

References

[1] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," *CoRR*, vol. abs/1710.00811, 2017. [Online]. Available: http://arxiv.org/abs/1710.00811

[2] M. Mayhew, M. Atighetchi, A. Adler, and R. Greenstadt, "Use of machine learning in big data analytics for insider threat detection," in *MILCOM 2015-2015 IEEE Military Communications Conference*. IEEE, 2015, pp. 915–922.

[3] M. Shashanka, M.-Y. Shen, and J. Wang, "User and entity behavior analytics for enterprise security," in *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 2016, pp. 1867–1874.

[4] I. Ghafir, M. Hammoudeh, V. Prenosil, L. Han, R. Hegarty, K. Rabie, and F. J. Aparicio-Navarro, "Detection of advanced persistent threat using machine-learning correlation analysis," *Future Generation Computer Systems*, vol. 89, pp. 349–359, 2018.

[5] B. E. Strom, J. A. Battaglia, M. S. Kemmerer, W. Kupersanin, D. P. Miller, C. Wampler, S. M. Whitley, and R. D. Wolf, "Finding cyber threats with attck-based analytics," 2015. [Online]. Available: https://www.mitre.org/publications/technical-papers/finding-cyber-threats-with-attck-based-analytics

[6] K. Kent and M. P. Souppaya, "Sp 800-92. guide to computer security log management," 2006.

[7] D. Green, "Building cyber resilience by changing your approach to testing," 2020. [Online]. Available: https://www.f-secure.com/en/consulting/our-thinking/cyber-security-resilience

[8] C. Zimmerman, "Cybersecurity operations center," *The MITRE Corporation*, 2014.

[9] J. Kerwin, "Applying the scientific method to threat hunting," Tech. Rep., 2020. [Online]. Available: https://www.sans.org/reading-room/whitepapers/threathunting/applying-scientific-method-threat-hunting-39610

[10] Microsoft, *Use Windows Event Forwarding to help with intrusion detection*, Microsoft, 2019. [Online]. Available: https://docs.microsoft.com/en-us/windows/security/threat-protection/use-windows-event-forwarding-to-assist-in-intrusion-detection

[11] C. Wang, J. Clark, and D. Wilcox, "The state of the soc," 2018. [Online]. Available: https://fidelissecurity.com/resource/report/the-state-of-the-soc/

[12] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of management information systems*, vol. 24, no. 3, pp. 45–77, 2007.

[13] V. Mavroeidis and A. Jøsang, "Data-driven threat hunting using sysmon," in *Proceedings of the 2nd International Conference on Cryptography, Security and Privacy*, 2018, pp. 82–88.

[14] F. Roth and T. Patzke, *What is Sigma*, 2020. [Online]. Available: https://github.com/Neo23x0/sigma#what-is-sigma

[15] G. Couchard, W. Qimin, and T. L. Siew, "Catching lazarus: Threat intelligence to real detection logic - part one," 2020. [Online]. Available: https://labs.f-secure.com/blog/catching-lazarus-threat-intelligence-to-real-detection-logic/

[16] Q. Chen and R. A. Bridges, "Automated behavioral analysis of malware: A case study of wannacry ransomware," in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2017, pp. 454–460.

[17] Sqrrl, "Hunt evil - your practical guide to threat hunting," 2021. [Online]. Available: https://www.threathunting.net/files/hunt-evil-practical-guide-threat-hunting.pdf

[18] Microsoft, *Netlogon Elevation of Privilege Vulnerability - CVE-2020-1472*, 2020. [Online]. Available: https://msrc.microsoft.com/update-guide/vulnerability/CVE-2020-1472

[19] M. Simakov and Y. Zinar, "Zerologon (cve-2020-1472): An unauthenticated privilege escalation to full domain privileges," 2020. [Online]. Available: https://www.crowdstrike.com/blog/cve-2020-1472-zerologon-security-advisory/

[20] T. Tervoort, "Zerologon: Instantly become domain admin by subverting netlogon cryptography (cve-2020-1472)," 2020. [Online]. Available: https://www.secura.com/blog/zero-logon

[21] V. Vissamsetty, "Protection against targeted active directory ransomware," 2020. [Online]. Available: https://medium.com/attivotechblogs/protection-against-targeted-active-directory-ransomware-edf86fbbb389

[22] P. Berba, "Data analysis for cyber security 101: Detecting lateral movement," 2020. [Online]. Available: https://towardsdatascience.com/data-analysis-for-cyber-security-101-detecting-lateral-movement-2026216de439#9c29

[23] Secureworks, “Bronze union cyberespionage persists despite disclosures,” 2017. [Online]. Available: https://www.secureworks.com/research/bronze-union

[24] F. Amrouche, S. Lagraa, G. Kaiafas, and R. State, “Graph-based malicious login events investigation,” in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2019, pp. 63–66.

[25] Microsoft, *4742(S): A computer account was changed*, 2017. [Online]. Available: https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4742

[26] M. Felton, “Digging deep into the ad ds workstation logon process – part 3,” 2017. [Online]. Available: https://journeyofthegeek.com/2017/03/23/digging-deep-into-the-ad-ds-workstation-logon-process-part-3/

[27] Crowdstrike, “Nowhere to hide – 2020 threat hunting report – insights from the overwatch team,” 2020. [Online]. Available: https://go.crowdstrike.com/rs/281-OBQ-266/images/Report2020OverWatchNowheretoHide.pdf

[28] Microsoft, *Network security: Restrict NTLM: Audit NTLM authentication in this domain*, 2017. [Online]. Available: https://docs.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/network-security-restrict-ntlm-audit-ntlm-authentication-in-this-domain

[29] J. Petters, “Complete guide to windows file system auditing,” 2020. [Online]. Available: https://www.varonis.com/blog/windows-file-system-auditing/

[30] Microsoft, *https://www.varonis.com/blog/windows-file-system-auditing/*, Microsoft, 2017. [Online]. Available: https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4624

[31] Stealthbits, “Ntds.dit password extraction,” 2020. [Online]. Available: https://attack.stealthbits.com/ntds-dit-security-active-directory