FH University of Applied Sciences TECHNIKUM WIEN

MASTER THESIS

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering at the University of Applied Sciences Technikum Wien - Degree Program IT-Security

A Methodology for the Automatic Generation of Realistic Log Datasets of a Windows Domain for Testing Cybersecurity Machine Learning Algorithms

By: Sebahattin Sahin, BSc

Student Number: 52003789

Supervisors: Ing. Daniel Kissler, MSc MA
Dr. Dr. Florian Skopik

Vienna, September 22, 2024

Declaration

“As author and creator of this work to hand, I confirm with my signature knowledge of the relevant copyright regulations governed by higher education acts (see Urheberrechtsgesetz /Austrian copyright law as amended as well as the Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien as amended).

I hereby declare that I completed the present work independently and that any ideas, whether written by others or by myself, have been fully sourced and referenced. I am aware of any consequences I may face on the part of the degree program director if there should be evidence of missing autonomy and independence or evidence of any intent to fraudulently achieve a pass mark for this work (see Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien as amended).

I further declare that up to this date I have not published the work to hand nor have I presented it to another examination board in the same or similar form. I affirm that the version submitted matches the version in the upload tool.“

Vienna, September 22, 2024 Signature

SIGNATURINFORMATIONEN	**Unterzeichner**	Sebahattin Sahin
	Datum/Zeit-UTC	2024-09-22T17:43:08+02:00
	Prüfinformation	Informationen zur Prüfung der elektronischen Signatur finden Sie unter: https://www.signaturpruefung.gv.at
Hinweis	Dieses mit einer qualifizierten elektronischen Signatur versehene Dokument hat gemäß Art. 25 Abs. 2 der Verordnung (EU) Nr. 910/2014 vom 23. Juli 2014 ("eIDAS-VO") die gleiche Rechtswirkung wie ein handschriftlich unterschriebenes Dokument.	

Kurzfassung

Intrusion Detection Systeme zählen zu den wichtigsten Security Assets in Enterprise-Umgebungen [1]. Ihre Stärke gegenüber anderen Security Appliances liegt in der Anomalie-Erkennung durch Einsatz von Machine Learning Algorithmen, wodurch sie Schutz vor bekannten und unbekannten Angriffen, wie etwa Zero-Day-Exploits, bieten, indem sie das Verhalten von Systemen und dem Netzwerkverkehr überwachen [2]. Die Effektivität eines IDS hängt jedoch stark von der Qualität der Trainings-Datensets ab, welche oft veraltet sind und den aktuellen Threat Landscape nicht widerspiegeln [3],[4]. Besonders im Windows-Bereich gibt es eine Forschungslücke sowie fehlende aktuelle Datensets [5]. Diese Arbeit setzt hier an und hat als Ziel die automatisierte Erstellung von Datensets für das Training von Machine Learning Algorithmen mit Fokus auf Windows-Enterprise Umgebungen.

Zu Beginn werden Anforderungen an Windows-Log-Datensets formuliert, die als Grundlage für das Framework-Design dienen. In einem iterativen Entwicklungsprozess wird in der ersten Iteration der Fokus auf die Evaluierung der notwendigen Komponenten für die Bereitstellung eines Security-fokussierten Testbeds gelegt. Dazu gehört das Log-Management, die Verarbeitung von Logs, die Simulation von Benutzeraktivitäten und die Tools für das automatisierte Deployment. Da der Schwerpunkt auf Windows Enterprise-Umgebungen liegt, werden spezifische Konfigurationen wie Group Policy Objects und Audit Regeln berücksichtigt. Zudem werden Angriffe durchgeführt, die eine Vielzahl von Taktiken und Techniken aus dem MITRE ATT&CK-Framework abdecken.

In der zweiten Iteration wird die technische Implementierung des AECID-Win-Testbeds präsentiert. Diese Implementierung folgte den MDE-Prinzipien, indem Modelle für das Testbed, die Benutzer-Simulation und die Angriffe erstellt werden. Die Integration von GHOSTS-NPC ermöglichte realistische Benutzer-Simulationen, während Attackmate für die automatisierte Ausführung einer Angriffskette eingesetzt wird. Ein konzeptionelles Modell der verschiedenen Layer, einschließlich Modellierung, Testbed, Datensammlung und Datenset, wird vorgestellt. Die Funktionalität des Frameworks wird getestet, und das generierte Datenset weiter evaluiert. Dabei erfolgt die Zuweisung von Attributen und Metriken wie Precision, Recall und F1-Score. Die Ergebnisse zeigen, dass das vom Framework erzeugte Datenset einsatzfähig ist und die Funktionsfähigkeit des AECID-Win-Testbeds insgesamt bestätigt wird.

Die Forschung folgt der Methodik des MDE und verdeutlicht deren Eignung für die automatisierte Generierung von Datensets für IDS. Die Konfigurationen, Skripte und das erzeugte Datenset werden parallel mit dieser Arbeit von der AIT veröffentlicht.

Schlagworte: IDS, Security Testbed, Cyberangriffe, Logdatenset

Abstract

Intrusion detection systems are among the essential security assets in enterprise environments, providing multiple layers of security and aligning with the principles of defense in depth [1]. Their strength, compared to different security appliances, lies in the detection based on anomalies achieved through machine learning algorithms, offering protection against known and unknown attacks, such as zero-day exploits, by monitoring system behavior and network traffic patterns [2]. However, the efficacy of IDS relies on the quality of the dataset used to train it, which often is outdated and does not reflect the current threat landscape [3],[4]. Especially in the area of Windows, there is a research and dataset gap [5]. This thesis addresses this gap by proposing the AECID-Win-Testbed framework, aiming for the automated generation of log datasets specific to Windows environments, built upon the principles of model-driven engineering.

Requirements were formulated for Windows log datasets, upon which the framework design was based. In an iterative development process, the initial iteration focused on evaluating components necessary for deploying a security-focused testbed. This included log management, downstream processing of logs, user simulation and the tooling for automated deployment. As the focus is on Windows enterprises, domain-specific configurations were reviewed, including group policy objects and audit rules. Additionally, attacks were executed which cover a wide range of tactics and techniques from the MITRE ATT&CK framework.

In the second iteration, the technical implementation of the AECID-Win-Testbed is presented. This implementation followed the principles of MDE, defining models for the testbed, user simulation and attack chain. The integration of GHOSTS-NPC ensured realistic user simulations and Attackmate was utilized for the automated execution of an attack chain. A conceptual model of the layers, including model, testbed, collection and dataset, was presented. Functional testing of the framework was conducted and the resulting dataset was evaluated. This evaluation involved assigning attributes and calculating metrics such as precision, recall and F1-score. The results demonstrated that the dataset generated from the framework is usable and overall confirmed the functionality of the AECID-Win-Testbed itself.

The research follows the proposed methodology of MDE and highlights its feasibility for the automated generation of datasets for IDS. Configurations, scripts and the dataset itself will be published by AIT simultaneously with this thesis.

Keywords: IDS, security testbed, cyberattacks, log dataset

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my wonderful wife Elif, for her unwavering support throughout this journey. Your encouragement has been a constant source of strength, and the gift of our daughter Asya, has opened my eyes to a new perspective on life. Through her, I have come to understand what truly matters and for that, I am forever grateful.

I would also like to thank my parents and brothers for their constant encouragement and belief in me. Your support has always pushed me to move forward, even in the most challenging moments.

A special thank you to my supervisor Daniel Kissler, for setting high standards and challenging me to push beyond my limits. Your guidance has been invaluable to my academic growth and personal development.

I also extend my gratitude to the AIT members — Florian Skopik, Max Landauer, Markus Wurzenberger and Wolfgang Hotwagner. Working with you has been a great joy and experience. Your insights and collaboration have fundamentally shifted the way I approach problems, for which I am truly thankful.

Contents

1 Introduction

Intrusion detection systems are an integral part of a company's security setup, fitting into the overall approach of employing multiple layers of security and aligning with the principle defense-in-depth strategy [1]. Their unique strength, in contrast to other security appliances, lies in their ability to detect anomalies using machine learning algorithms, thereby offering the best defense against unknown threats such as zero-day attacks, by monitoring system behavior and network traffic patterns [2]. However, the effectiveness of these algorithms is directly linked to the quality of the datasets used to train them [3]. The process of creating a dataset is time-consuming, involving the deployment of infrastructure, execution of various attacks, collection of logs and the application of labeling rules to determine ground truth [6]. This thesis aims to establish a framework for the automated generation of log datasets, with a specific focus on a Windows enterprise, for training host-based intrusion detection systems.

1.1 Motivation

As of 2024, AI and machine learning technologies have reached a state of the art and are significantly impacting our lives. This highlights the critical need for high-quality datasets to effectively train these algorithms. Furthermore, threat actors are also leveraging these technologies to enhance their malicious activities and even individuals with limited technical skills, often referred to as "script kiddies", can access harmful information through the use of AI [7]. Given these concerns, the significance of this thesis is underscored, as IDS remains a core defense mechanism utilized by security analysts [6].

The current state of automated log dataset generation is aligning with trends towards IoT, yet it lacks further enhancement for Windows enterprises as presented in Section 2.1. This is concerning because research conducted in the past may not reflect the circumstances of today, given that technology is constantly changing and evolving, as is the threat landscape. A study [4] evaluated the suitability of public datasets for current security environments and concluded that there is a lack of high-quality public datasets. Another statement is that the existing datasets are often outdated and no longer relevant for current threat landscapes. Therefore, it is necessary to further develop and re-evaluate older frameworks and approaches, as new technologies may offer improvements. A similar conclusion was presented in [8], which emphasized the evolving threat landscape as attackers leverage newer technologies, resulting in new patterns and artifacts. This underscores the need for datasets to encompass realistic scenarios and continuous development. The evolving nature of attacks also underscores the necessity for enhancements. For example, while Microsoft provided patches to mitigate "Kerberos Key Distribution Center (KDC) confusion" [9] and "SAM Account Name spoofing" [10], exploited through golden ticket attacks, they inadvertently introduced new attack vectors, such as the more sophisticated and stealthy diamond ticket attack [11]. This highlights the essential need for enhancements and further research in the area of automated dataset generation for anomaly detection across all domains.

Another aspect is the domain of existing Windows datasets. In the research sphere, the focus of Windows datasets is monitoring system calls [12], [13], [14], while in the technical domain, some datasets containing native Windows logs are publicly available on GitHub [15] and other websites [16]. The drawback of some of these datasets, created in a non-scientific manner, lies in their inappropriate methodological approach for the creation of those datasets. Additionally, there is minimal documentation regarding the underlying environment and limited to no information on how the attacks were performed. Consequently, an evaluation of the datasets is also absent, questioning their suitability for training and testing machine learning algorithms.

1.2 Related Work

This thesis was developed in collaboration with the Austrian Institute of Technology, following their request to expand their cyberrange capabilities by creating a security-focused testbed tailored to a Windows enterprise environment. The foundation of their testbed generation methodology is outlined in the paper "Have it your way" [6], which describes key design principles for generating log datasets and highlights the importance of a model-driven approach in testbed setup. A critical component of this approach involves leveraging model-driven engineering techniques to enable a higher level of abstraction, allowing the focus to remain on design rather than technical complexities. By defining input variables and using a transformation engine, this method facilitates the creation of multiple testbeds, processing the inputs to produce comprehensive testbed environments and diverse log datasets.

Through ongoing research, the AIT has released a subsequent paper titled "Maintainable Log Datasets for Evaluation of Intrusion Detection Systems" [17]. This publication outlines the requirements for high-quality log datasets. In addition, it gives insights into creation of synthetic data and provides arguments for it compared to the usage of real-world data. Further, aspects of user simulations are outlined to provide behavior based log patterns and strengthen robustness of IDSs. Nonetheless, the approach is facilitated through a state machine [18], which is linear and defines only a restricted range of interactions, thus limiting its ability to mimic real-world behavior comprehensively. This drawback is also addressed in this thesis.

The Kyoushi-Framework, as presented in "Quality improvement of labels for model-driven benchmarking of Intrusion Detection Systems" [19], integrates the fundamental principles outlined in [6] and builds upon them. Specifically tailored for a Linux environment, it also incorporates logs from the network infrastructure, distinguishing between the model layer, testbed layer, data collection layer and processing layer. The layers that include the testbed and scenario configuration provide a detailed representation of the simulated environment, guaranteeing reproducibility and establishing a single source of truth for the dataset. While the proof of concept for Kyoushi appears to be comprehensive, the framework itself displays considerable complexity, particularly in aspects such as log collection and integration into the "Kyoushi Dataset tool". Also, the state machines used for user simulations are also employed for conducting attacks which presents complexity as it interacts with an self developed Python module. This thesis addresses the challenges from Kyoushi and aims for overall simplification and optimization.

1.3 Methodological Approach

The creation of a framework for automated generation of log datasets was carried out through an initial analysis of requirements. These include design principles for log datasets, which were further detailed in Section 2.2.1. For a better understanding different papers were reviewed to gain insides on what requirements are genuinely set for different niches as Windows lacks on research duo it commercial manner as the requirements for datasets demands the use of open-sourced products [1], [17], [19].

In an iterative process, the three streams of testbed, user simulation and cyber attacks were handled accordingly. For the testbed, the necessary methodologies were evaluated to meet the defined requirements in Section 2.2.2 and address the research question described in Section 1.4. This involved researching common practices for Windows enterprise environments. In addition to the components and services used, the introduction of native log management was developed to derive further datasets for training machine learning algorithms. This also addresses the research question focusing on audit rules and evaluating suggested best practices, which were then compared. Another part of the environment determined by the requirements is the simulation of users, for which effective simulation methods were sought. The technologies used in related works have some shortcomings, prompting the discovery, evaluation and integration of a new framework. Besides infrastructure and user simulations, another significant aspect are the attack vectors on Windows enterprises. The attacks used in this thesis cover a broad range of the MITRE ATT&CK Framework, employing various tactics and techniques, particularly targeting Windows environments. The approach here was to perform the attacks based on pentestbooks and detect those subsequently based on IoCs. Consequently, the foundation for implementation was formed as the infrastructure and components were evaluated, users were effectively simulated and attacks were carried out, resulting in an initial prototype of the desired environment.

The second iteration, referred to as proof of concept phase, started by determining the framework's design and an implementation for the automatic generation of datasets was conducted based on the artifacts from the first iteration. The principles of model-driven engineering were applied to define a scenario, as specified in the requirements. By modeling the scenario, the use case and scope of the dataset can be defined. For reproducibility, the scripts, configurations, documentation and datasets will be published by the AIT. The intended framework, named AECID-Win-Testbed, includes the automated deployment of the Windows environment, user simulation and automated execution of attacks to further obtain an automatically generated dataset.

The results of the functional testing discuss the suitability of the AECID-Win-Testbed. In the next phase, further investigation into the requirements for quality data was conducted, focusing on machine learning. In this cycle, the evaluation of the dataset and its suitability for machine learning training was determined, utilizing metrics from the literature and visualizing patterns. In the final step, the dataset was validated by using it to train the AMiner [20]. The findings were further evaluated and compared with the labels, concluding in a feasible generated log dataset highlighting the usability of the developed framework AECID-Win-Testbed.

1.4 Research Questions

This thesis attempts to establish a framework for the automated generation of log datasets within a Windows environment. The objective is to facilitate the training and testing of host-based intrusion detection systems. Accordingly, the central research question posed is: "How can a framework be developed to automate the generation of log datasets tailored to a Windows enterprise, thereby enhancing the training of host-based intrusion detection systems?".

Following the formulation of this research question, subsequent inquiries raised:

- Q1 What methodologies enable the automated creation of a Windows testbed, specifically designed towards detecting or evaluating detection capabilities?
- Q2 What elements should the infrastructure contain to ensure its completeness, adaptability and adjustability?
- Q3 Which recommended practices for Windows audit policies ensure comprehensive logging for attack detection?
- Q4 How can realistic noise be achieved in the testbed based on simulations of users?

Q5 In what manner do the specified attacks manifest and are all attacks detectable in the Windows Event Logs?

Q6 What elements of the relevant work can be utilized, require adaptation, or need complete overhaul for the Windows enterprise scenario?

2 Methods

This chapter begins with a literature review to identify and establish the methodological foundation based on relevant works, focusing on a scenario-based approach, which consequently addresses Q1. Building on this research, general design principles for testbed methodologies are presented, followed by the definition of requirements for the desired framework. The development process is then carried out in iterations. The first iteration, also referred to as the prototype phase, outlines the procedure of discovering artifacts by answering the subsequent research questions Q2–Q5, with a focus on the three streams testbed, user simulation and cyberattacks. In the second iteration, also referred to as the proof-of-concept phase, the artifacts were further utilized and considered for the development of the framework "AECID-Win-Testbed." Here, the design and architecture were highlighted, including the presentation of the cyberattack automation.

This thesis was conducted in collaboration with the AIT to address their need for an additional scenario, specifically focusing on a security testbed for a Windows enterprise environment. Their current cyberrange scenarios contains "Videoserver," "Linux Malware," and "Lateral Movement" but doesn't contain any Windows machines [21]. In response to their request for a "Windows scenario", this thesis introduces the AECID-Win-Testbed as a framework for the automated deployment of a Windows enterprise environment. To support this effort, access to their private cloud infrastructure, OpenStack, was provided. The development was conducted independently of AIT and the integration into their infrastructure falls outside the scope of this work.

2.1 Literature Review

To initially gain an overall view of the current field of HIDS and log dataset generation, a general search was conducted in Google Scholar and IEEE using the keywords "host-based intrusion detection system" and "log dataset generation" for the years 2022 and 2023. The first three pages of each database revealed a clear trend toward IoT, with further publications on NIDS and Linux, but no recent publications on Windows enterprises.

Shifting the scope to cybersecurity testbeds, an evaluation of security focused testbeds was conducted in [22]. The systematic literature review analyzed 100 papers, with 94 of them providing details about scenarios. The second most common capability identified was management, with 91 instances found, followed by 86 with details about monitoring. Least classified capability contains details about scoring mechanism and was mentioned in 26 papers. The results show that cybersecurity testbeds typically operate based on a scenario-based approach, emphasizing the manageability of the cyber range. Another relevant paper on security centered testbeds can be found in [23]. This paper also emphasizes scenarios and achieves testbed manageability through technologies and techniques from DevOps, for the automated generation of testbeds which can be further utilized for cyber exercises, training and research.

After gaining insights from papers focusing on testbeds in a security context, the next research scope shifted towards automated log dataset generation, with a specific focus on Windows Event Logs. Several relevant methodologies have been identified and the studies within this domain primarily concentrated on either Windows API calls, as demonstrated in the well-known dataset presented in [12], or Windows logs, as discussed in [24],[25],[26]. In these referenced papers, the primary objective regarding Windows logs was to identify malware, leaving potential for further exploration into Windows Event Logs concerning malicious post-exploitation activities. A notable difference lies in the utilization of a sandbox environment for deploying malware in the referenced research, leading to a stronger emphasis on identifying traces of malware rather than examining the underlying environment, which is the scope of this thesis. Another method for generating Windows datasets is outlined in [27]. The approach emphasizes simulating Windows users and intruders, leaning towards a behavior-based strategy for masquerade detection. User simulation in this study involved participants, thereby generating real log data and classifying users based on their roles. However, the focus is not specifically on an enterprise environment but rather on a more generalized context, such as in the private sector. Consequently, event traces may vary due to differences in private and work-related usage patterns.

In addressing the literature gap within the context of Windows, the research extended to automated log dataset generation. The focus was on identifying papers that follow a model-based approach, as investigated in [22]. Consequently, the works from AIT [6],[17] were examined, with a focus on their utilization of a model-driven engineering approach within a scenario and their adherence to DevOps principles for infrastructure maintenance [23]. MDE was also a consideration in [19], where the framework Kyoushi was developed for the automated generation of labeled log datasets. A similar approach was used in the development of the CREME toolchain [28]. Automated dataset generation in CREME relied on various applications, but the underlying infrastructure was manually set up. VM image publication lacked flexibility, limiting environment customization. The infrastructure prioritized individual OS usage over integrated Windows enterprise systems, neglecting component interaction. Furthermore, user simulation wasn't considered, resulting in a gap in the dataset for capturing user-generated log patterns. Another framework for log data generation was found in [29] which prioritizes network traffic. The ID2T framework is presented in the referenced paper, where the technical implementation is based on a real network and malicious traffic is injected into benign traffic performed by the framework. A similar injection method is also found in [1],[30]. However, an injection based approach does not suit for endpoint related datasets due the challenges of time consistency and authenticity.

The result of the literature review aligns with the results in [22]. Furthermore, it has been noted that anomaly detection in native Windows logs is rare. Additionally, the references either address A) the automated creation of the testbed for dataset generation, B) the evaluation of a dataset, or C) a hybrid approach, which generates and evaluates the dataset, although the testbed is mostly created manually and does not further elaborate on the requirements for a dataset. To address Q1, this paper adopts the model-based approach and utilizes MDE as the methodology for developing the framework [6],[17],[19].

2.2 Requirements Analysis

A comprehensive analysis of the requirements and objectives was undertaken to understand the scope and demand. This was carried out through literature reviews.

2.2.1 Design Principles

Essential for deriving the requirements in the upcoming section are the design principles presented in [6], which provide general guidance for testbed generation methodologies.

Authenticity

Acquiring log data for comprehensive IDS assessment necessitates careful consideration of multiple aspects. Initially, it involves the selection and arrangement of all network elements, encompassing servers and clients, to mirror a network environment reflective of a particular usage scenario, while factoring the complexity of the environment. Subsequently, it requires the simulation of authentic behaviors displayed by network components and user interaction to generate different patterns within the logs. Thirdly, the simulation of attacks associated with recent vulnerabilities to accurately evaluate the IDS's efficacy, deploying software that is not outdated. Finally, it requires the authentic capture of log data, leveraging readily available log sources and configuring logging mechanisms in accordance with the specific usage context. [6]

Flexibility

To enhance the efficiency of setting up the testbed, it is essential to prioritize flexibility to facilitate adjustments and expansions, thus enabling iterative development. Important adaptations include scaling the network by adding or removing components, fine-tuning configurations and modifying dynamic behaviors and interactions among components. This approach fosters a customizable and reliable testing environment. [6]

Reproducibility

For reproducibility of log datasets, the ability to reset the testbed to a previous state is crucial, especially when investigating the effects of attacks on modified testbed versions. While it's challenging to guarantee identical reproduction due to communication latency's and unpredictable events, ensuring isolation from external sources, using open-source services and avoiding commercial products is essential to enhance the reliability of subsequent evaluations on captured data. Therefore, in this thesis Windows evaluation licenses were used.[6]

Availability

Publicly accessible datasets intended for IDS benchmarking must incorporate complete documentation, explaining the dataset's objectives, infrastructure configuration and description of typical and malicious behaviors. Such comprehensive documentation serves to facilitate meaningful comparisons and reproducibility of results. [6]

Utilizability

Simply possessing a dataset is insufficient for conducting a thorough IDS evaluation. A vital aspect is the establishment of a quantifiable ground truth that clearly defines malicious behavior, enabling meaningful comparisons. While basic labeling categorizes all log events during attack periods as malicious, a more precise evaluation involves labeling only those events that genuinely signify malicious activity. Enhanced outcomes are attained through labels that distinguish between different attack types or specific attack stages. [6]

2.2.2 Requirements

To evaluate the requirements for this thesis, it was derived from the design principles in the previous section in combination of related work [1], [17], [19], [28]. The following requirements were defined for HIDS and the methodological creation of log datasets for Windows domains.

R1 Scenario based
Regarding the authenticity design principle, the infrastructure model should be created to meet a particular scenario. This condition ensures the simulation's scope and helps to define the dataset's limitations. Also this requirement highlight the focus of real-world scenarios which strengthens realism.

R2 Synthetic data generation
In the context of synthetic data generation, the dataset is regarded as synthetic since it was generated within a simulated environment, offering the advantage of not raising privacy concerns.

R3 Collection of system logs
The primary focus of this thesis is on generation datasets based of Windows Logs, which are located in the Windows Event Viewer. This necessitates the collection of the relevant event logs for detecting malicious activities implicating the correct setup of audit policies and ensuring completeness of the data.

R4 Integration of user behavior
The integration of user behavior is crucial. This should contain periodic simulation patterns in form of different activities, which in its simplest form should reflect work hours from 9 to 5, including breaks.

R5 Performing relevant attacks
The evolving threat landscape introduces new attack vectors which are important factors in benchmarking IDSs. Outdated attacks may decrease the relevance of the generated dataset which would directly influence the usability of datasets.

R6 Attack repetitions
For the recreation of log datasets, it is essential to emphasize that the attacks must be repeatable. This aspect is also beneficial for anomaly-based algorithms, which classify attacks using clustering aggregation methods, as mentioned in [3].

R7 Attack variations
This requirement is established for the sake of flexibility. It is not only the environment itself that needs to be adaptable, but also the executed attacks. Therefore, performing the same attack with different parameters provides anomaly detection with a wider range of patterns to learn from, thereby enhancing robustness. Additionally, the modeling of attacks must be considered, as they vary according to the desired scenario.

R8 Labeling ground truth
To test and evaluate the efficiency of an IDS, it is crucial to label malicious logs, which is necessary for meaningful benchmarking. Additionally, supervised machine learning algorithms require labeled datasets for training to ensure robustness and accurate predictions. The process of labeling varies depending on the approach but must be performed with accuracy to achieve precise results.

R9 Documentation
Aligning with the design principle of availability, it is essential to provide documentation of the dataset, including aspects of data creation, infrastructure setup, user simulation and

executed attacks. This documentation is necessary for the public to gain a comprehensive understanding of the artifacts.

R10 Reproducibility of datasets
In scientific work, the reproducibility of results is crucial and the same importance applies to regenerating a dataset. Given that the applications used are periodically updated, it is necessary to be able to reproduce the dataset with similar results while keeping the environment, user and simulation unchanged and only updating components or services. This approach is essential to prevent the dataset from being marked as outdated.

2.3 Iterative Development - Iteration 1

The approached framework was developed iteratively in alignment with the principles of design science research. During this iteration, a prototype was developed to further evaluate the suitability and capability of the technologies and methodologies used. Subsequently, the core components for the approached framework were identified. Further, in the second subsection, the attacks were performed with the aim of understanding how to execute them and how they manifest in the logs. Based on these artifacts, a concept was designed in Section 2.4 and further a proof-of-concept implementation presented in Section 3.2.

2.3.1 Iteration 1 - Testbed Design and Infrastructure Evaluation

In the first iteration, a static testbed was set up using the recommended tooling outlined in [23]. As for the infrastructure setup, Terraform was reviewed and due the change of their license model from open-source to commercial, as alternate option OpenTofu was chosen [31]. Further abbreviated as Tofu, which is a fork from Terraform using the Mozilla license and being maintained by the community, it provides an accessible open-source IaaC tool which is employed in this thesis. Alternatives to Ansible, like Chef and Puppet were reviewed but not considered as one of the key aspects is the architecture, leading by Ansible with an agent-less approach compared to the master-agent architecture of the others [32]. Ansible, as Configuration-as-Code, is fundamentally situated in the Linux world. With Ansible, a central repository of configurations for Linux components is maintained. Playbooks and roles are used to manage the Linux devices from a centralized location. However, this work focuses on a Windows scenario, which is why harmonization was considered, facilitated by the presence of Windows modules in Ansible.

In the realm of Linux, the counterpart to Ansible are essentially Group Policy Objects on Windows. GPO serves as a central layer for configuring the Windows environment, much like Ansible does for Linux. Initially, differences in system architecture of Linux and Windows were taken into account, as previous work solely utilized Ansible for configuring system components. However, to effectively address the research question Q1 and provide authenticity of the dataset, it becomes crucial to establish a Windows environment containing a domain controller and centralized management through GPOs. Consequently, an additional configuration layer is integrated into the Infrastructure Model in Section 2.4.2, as relying solely on Ansible modules would limit configuration options to local policies, which does not align with the objective R1. However, domain preferences through Ansible were set by utilizing the script module allowing the execution of Powershell scripts and importing GPOs with so called migration tables using Powershell cmdlets.

For the logging aspect, related works were considered and it was decided to approach this point differently. In [6],[17],[19], logs are collected from all components through a script, whereas the consideration was made to establish centralized log management of Windows Event Logs. This simplifies further log processing by having a single point of source instead of establishing multiple connections to all Windows machines and extracting the logs. In the context of reproducibility, this eliminates the need for multiple external connections into the infrastructure, which would manifest as unwanted artifacts in the logs. Instead, a Windows-native solution called Windows Event Collector was evaluated. Windows Event Collector is relevant for the centralised logging and also mentioned by Palantir [33] and Crowdstrike [34], given common practices on preferred log shipping technologies.

In parallel with centralized logging are the audit rules. To answer the research question Q3 regarding the configuration of audit rules, a comparative analysis of best practices is presented in Table 2 and 3 in Appendix B. The examination revealed that the best practices recommended by Palantir offer more explicit policies compared to those recommended by Microsoft. These disparities were also taken into account in the context of the cyberattacks discussed in Section 2.3.2. Thus, the decision was to implement the practices recommended by Palantir as certain performed attacks would only manifest themselves in the Windows Logs, when the additional audit rules where set. To fully meet R3, the collection of system logs, Sysmon was also incorporated to enrich the Windows logs and detect attacks that do not manifest in the logs in native manner, as example process injection techniques.

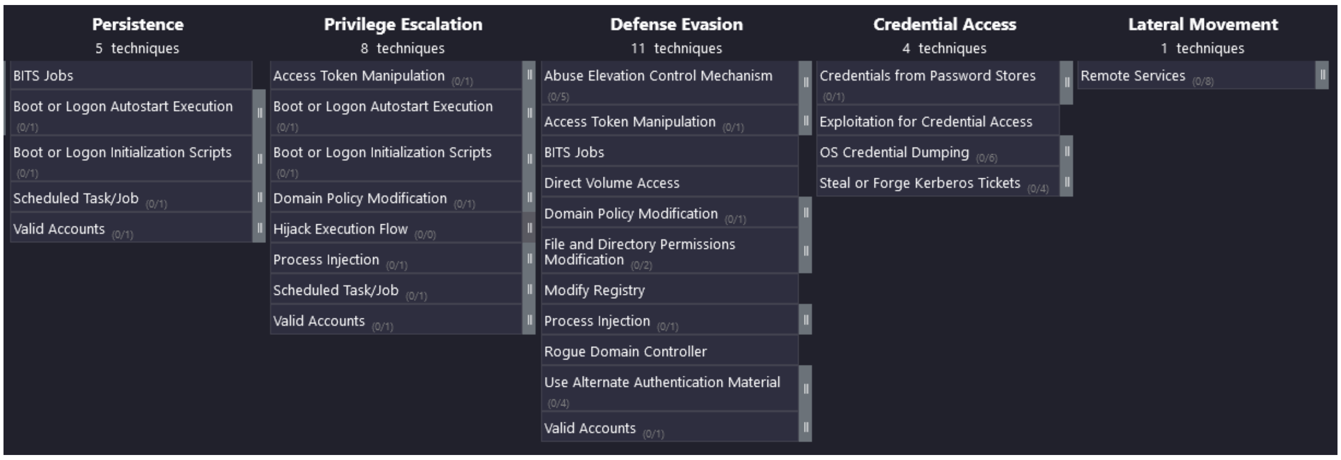
The integration of emulated users corresponds to requirements R2 plus R4 and was also formulated as in question Q4. In related works, this was achieved through so-called state machines, which provide a linear sequence of activities reflected as patterns in the logs. To make the state machines more realistic, timeouts were used. Additionally, the state machines were developed using a custom-developed Python library to also conduct automated attacks. However, in this thesis, the state machines [18] were replaced because of complexity and lack of documentation. Here, the integration of the Ghosts NPC Framework into the technical implementation was evaluated [35]. Ghosts was specifically developed for the purpose of emulating user, with agents on the Windows components acting as NPCs. Furthermore, it allows for running various user profiles on the agents to emulate different employees, thereby enhancing aspects of authenticity and improving user behavior patterns. The framework simplifies the integration of user simulation and also provides more possibilities to model the desired simulation behavior, offering a streamlined solution for generating user patterns in the dataset.

Moving forward towards dataset generation, it is necessary to prepare the collected Windows logs to fulfill R8, labeling ground truth. Since there is no native Windows solution for this purpose and the ELK stack represents increased complexity in previous works [19], Kafka as an alternative was evaluated [36]. Thus, was also mentioned by Crowdstrike [37] who also highlights the features of Kafka including storing events in an ordered sequence and providing fault-tolerance. Kafka, as a message queue, was designed for data streaming and allows further downstream processing, whereas the ELK stack is often used as a log management solution, which has already been established by the WEC. Due to the possibility of processing the incoming data, a simplified process is created here to label the dataset accordingly which is crucial for the training of machine learning algorithms. Another facet involves the transfer of Windows Logs from WEC to Kafka, facilitated by Winlogbeat [38]. One notable advantage is its capability to convert logs from EVTX format to JSON, coupled with its fault-tolerant design that prevents data loss during network interruptions and thereby supporting the completeness of the dataset. Notably, Winlogbeat is also referenced as the preferred log shipping tool by Crowdstrike [37].

2.3.2 Iteration 1 - Cyberattacks

This section covers cyberattacks and, in cooperation with AIT, a list of attacks, found in Appendix A, was discussed as part of R5, performing relevant attacks. The attacks covers many tactics from the MITRE Framework and include various techniques within the employed tactics, which is illustrated in Figure 1. The mapping provides a wide range of techniques which are also deployed through malicious actors with advanced variations. Based on the mapping, the scope of the modeling is well-defined. It is excluded from the scope that an initial intrusion into the infrastructure is needed or that a user's credentials must be guessed via brute force. This is predetermined in the scenario, which focuses on post-exploitation attacks carried out within the environment. This is also due to the fact that the dataset is based on Windows logs and accordingly, the goal is to detect anomalies in the Windows logs caused by the attacks.

As part of the first iteration, the formulated research question Q5, on how the specified attacks manifest in the Windows logs, was examined which also supports the decisions for audit policies of Q3. As part of this process, attacks were manually executed on the statically generated testbed. The focus was primarily on feasibility and detection. Herefore, literature was reviewed on how to perform the attacks and variations of it were tested. For the detection IoCs were reviewed and cross-checked with the logs generated through the attacks. This phase was also utilized to evaluate the research question regarding audit policies. Non-visible attacks were further analyzed to determine whether the policy lacked configuration or if they do not manifest within the Windows logs. The assessment of attacks is found in Apendix A. As the aim of this thesis is to automate the generation of log datasets for testing machine learning algorithms, adaptations had to be made to the environment to enable the execution of the malicious activities. Furthermore, attention was also given to the relevance of the attacks and variations were performed according to R7, Attack variations.



Persistence 5 techniques	Privilege Escalation 8 techniques	Defense Evasion 11 techniques	Credential Access 4 techniques	Lateral Movement 1 techniques
BITS Jobs	Access Token Manipulation (0/1)	Abuse Elevation Control Mechanism (0/5)	Credentials from Password Stores (0/1)	Remote Services (0/8)
Boot or Logon Autostart Execution (0/1)	Boot or Logon Autostart Execution (0/1)	Access Token Manipulation (0/1)	Exploitation for Credential Access	
Boot or Logon Initialization Scripts (0/1)	Boot or Logon Initialization Scripts (0/1)	BITS Jobs	OS Credential Dumping (0/6)	
Scheduled Task/Job (0/1)	Domain Policy Modification (0/1)	Direct Volume Access	Steal or Forge Kerberos Tickets (0/4)	
Valid Accounts (0/1)	Hijack Execution Flow (0/0)	Domain Policy Modification (0/1)		
	Process Injection (0/1)	File and Directory Permissions Modification (0/2)		
	Scheduled Task/Job (0/1)	Modify Registry		
	Valid Accounts (0/1)	Process Injection (0/1)		
		Rogue Domain Controller		
		Use Alternate Authentication Material (0/4)		
		Valid Accounts (0/1)		

Figure 1: MITRE Mapping

2.4 Iterative Development - Iteration 2

In the second iteration, the implementation of the framework was developed based on the artifacts gained from the first iteration. To accomplish this, the design phase was utilized to associate both the findings and related works, thus defining a fundamental concept accordingly. This was then expanded to include the architecture of the framework, which methodically follows a model-based approach. Furthermore, in this step, modeling was conducted in 4 layers. The technical implementation of the framework is presented in Section 3.2.

2.4.1 Framework Design

During the design of the framework for Windows enterprises, the technical implementation of related work was thoroughly examined. This assessment included an evaluation of the potential utilization of elements from previous work. Additionally, the findings from the initial iteration were leveraged to outline the essential methodologies while simultaneously conducting a comparative analysis of the technology stack in relation to [6],[17],[19]. As the primary objective of the framework is the automated generation of datasets containing Windows logs, a conceptual processing for creating the datasets, in the form of a Chevron process, is introduced in Figure

2.

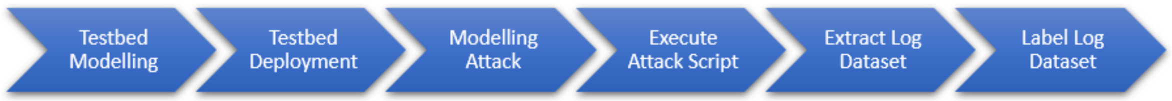


Figure 2: Log dataset generation procedure

The visual representation describes also the data processing process within the framework and was derived with adoptions from [17]. Starting from defined use cases, the respective models for the testbed, simulated user and attack chain are established. These models can be defined in parallel to each other and are further explained in Section 2.4.2. Subsequently, the provisioning, including configuration of the testbed, is performed. This is executed using the referenced scripts in Appendix C. The setup of the NPCs is done as part of the provisioning and therefore does not require a dedicated manual step but is modeled by adopting the configuration file of the NPCs. However, the attacks must be launched from the attacker machine, necessitating in manual execution of the playbook. Further details about the attack automation is described in Section 2.4.3. The collection of logs and their shipping to the Kafka broker is done automatically and integrated into the testbed. For downstream processing of the raw logs, the collected logs has to be exported where then labeling rule are applied to produce a labeled log dataset. The design supports rapid deployment and destruction of the testbed.

2.4.2 Framework Architecture

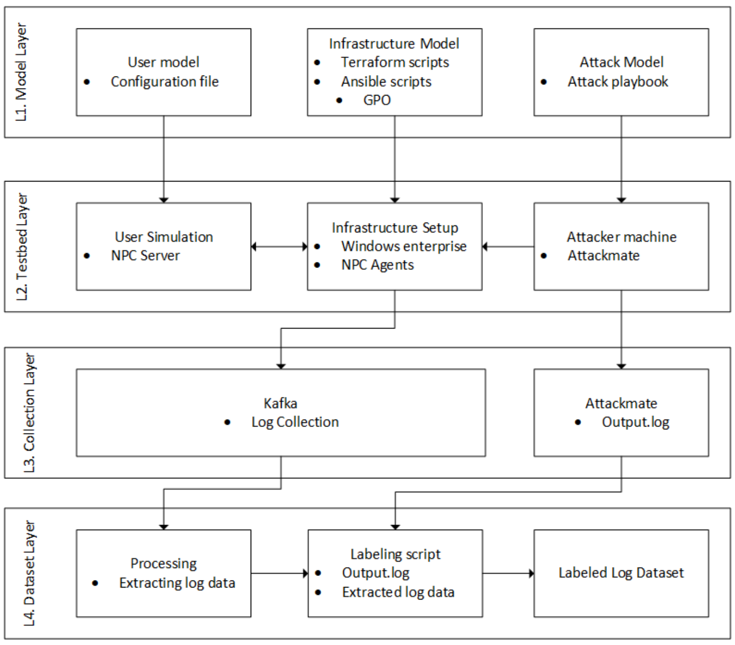


Figure 3: Testbed Generation Concept

The architecture was derived from related work [6],[17],[19] and adapted specifically for the Windows environment. The concept of dataset generation, illustrated in Figure 3, is structured into 4 layers. The top layer L1 comprises the models that function as testbed-independent models [6],[17]. In this layer, the infrastructure model, user behavior and attack model are designed and defined through variables and configurations. These models are then applied to layer 3 via a transformation engine, resulting in testbed-specific models [6],[17]. The testbed is generated here and the functions user simulation, log collection and log forwarding are already integrated. The playbook for automated attacks must be manually executed from a dedicated attacker machine and should be considered as an individual component within the environment. Although log collection is already integrated into the testbed, it is considered its own layer containing log data. The decision not to integrate L3 and L2 is based on the potential for further research when deploying a Kafka broker. As a message queue, it supports multiple input channels, similar to connectors of a SIEM, allowing the collection of not only Windows logs but also the consumption of network logs or syslogs. Therefore, it may be further utilized for different scenarios. In Layer 4, the preparation of log collections takes place by applying labeling rules and creating a log dataset. This is accomplished through the manipulation of the file using different queries through a script which adds the labels and finally results in a labeled log dataset.

Model Layer

The model layer is used to define the simulated environment and serves as the single source of truth for the dataset and limits its scope. Within this layer, the modeling of the infrastructure, user simulation and attacks is conducted by defining input variables and configuration files. This fulfills the design principle of reproducibility, as the generated dataset can be recreated based on these definitions. Furthermore, this layer is defined as testbed-independent models [6],[17], which will be abbreviated as TIM. These models can be adopted based on the defined scenario, offering templates for the designated models.

User Model The user model is defined through a configuration file which can be added to deployment scripts. In this section varieties of logs are produced based on the given scenario. Thus, it is necessary to have benign system behavior and user created log patterns for the enhanced training of supervised machine learning algorithms. In this thesis the user model is defined for the GHOSTS framework and also supports different user profiles, which makes it possible to define realistic user models based on a comprehensive analysis of characteristics of system usage. Through the GHOSTS management server, user interactions can be added during runtime.

Infrastructure Model The infrastructure model is responsible for defining the testbed. The size of the cyberrange can be set in configuration files of Terraform, while configurations of hosts are made through Ansible and domain preferences are defined as GPOs, also applied through Ansible. By using state-of-the-art technologies, the infrastructure setup is automated, facilitating infrastructure- and configuration-as-code technologies. This enables reproducibility while minimizing the time required to provision the environment. These technologies also allow for the design of more complex environments based on the required scenario. In contrast to [19], a less flexible approach has been implemented here, as otherwise it would bring high complexity and have a deterrent effect. Therefore, the creation of the infrastructure is to some extent static, but includes all elements necessary to generate a testbed designed for the creation of datasets. Furthermore, the framework provides basic functions in the form of Ansible roles to enable infrastructure extensibility in a simplified manner.

Attack Model This model is used to simulate malicious activities within the environment, which are reflected in the logs. Depending on the use case, different attack scenarios may be employed on the same infrastructure. By utilizing Attackmate [39], an attack orchestration tool that integrates various penetration testing software products, it is possible to execute a wide range of techniques and also deploying variations of it.

Testbed Layer

The testbed layer contains the orchestration of components in the environment for building the enterprise, the attacker machine facilitating the attack orchestration and a C2 server for the centralized configuration of the emulated users. The testbed layer is created through a transformation engine, which processes the TIMs defined in L1, the User Model layer, and creates a testbed-specific model, further abbreviated as TSM. The transformation is executed automatically by the designated scripts, allowing the creation of multiple testbeds isolated from each other, with different configurations, user simulation profiles and attack vectors. This provides a method for the rapid generation of testbeds. The limitation of the TSM lies in the design of the framework as it provides static templates containing necessary components and configurations for the testbed with focus on dataset generation but enhances flexibility, adoptability and extendability. Therefore, it is a crucial aspect to add TIMs based on the defined scenario.

Collection Layer

The collection layer serves as the central repository for log collection. In this thesis, it contains Windows Event Logs retrieved from Kafka and the output file of Attackmate. Within the raw logs, patterns for system and user behavior can be found, as well as malicious events. The output file generated by Attackmate through the execution of the attack chain is further utilized in the dataset layer. The collection layer is fully integrated into the framework and is stated as a core element of it.

Dataset Layer

The labeling layer in this concept is responsible for downstream processing the collected raw log data. As described in R8, labeling the ground truth is a crucial task for training supervised machine learning algorithms. To achieve this, a correlation is established between the executed attacks and the collected logs in order to trim the amount of logs. Powershell is then used to query the dataset and a example is given in Section 3.3.

2.4.3 Cyberattack Automation

Ensuring reproducibility necessitated the automated deployment of cyberattacks, enabling consistent results across multiple executions and facilitating framework validation. Leveraging Attackmate, a tool for attack orchestration, automated attacks were executed following the predefined attack chain written in a playbook. This tool was selected as it was developed by the AIT, offering the option of direct support. Additionally, the functionality of Attackmate for a Windows scenario was tested during this thesis, as it had not been conducted before given an unique setup for AECID-Win-Testbed. Additional advantages lie in reproducibility, R7 attack variation and R6 attack repetition. Also compared to the state machines, it offers a tool which utilizes well known offensive frameworks which are also employed by advisories.

The selection of attacks aimed to strike a balance between stealthiness and detectability, targeting identifiable traces in Windows Event Logs while mitigating log overload. An additional selection criterion for attacks is based on relevant attack vectors targeting Windows systems. Variations of techniques were also considered in this context. Considering these points, the subsequent goal is to manifest different patterns in the logs to ensure the suitability of the datasets and generate representative samples. The objective was to simulate domain compromise while maintaining visibility. By modeling the attack chain, the framework enables the creation of diverse datasets containing different malicious patterns with minimal manual intervention. This approach not only ensured reproducibility but also enhanced the framework's adaptability to evolving threat landscapes.

When using Attackmate, limitations were discovered and issues were submitted on Github. While using Evil-WinRM through Attackmate, the issue occoured that local interactive commands were not supported. This limitation manifested as commands being sent through WinRM but receiving no output, resulting in blind operation. This Github issue [40] was fixed during the finalization of the practical part. Another issue occurred when using the persistence module within Meterpreter. Troubleshooting the module within Attackmate was not possible as the verbose mode for msf-modules was not functioning [41]. As of the time of writing, this issue has not been resolved, resulting in commenting out the persistence mechanics in the playbook and the scenario of hiding the payload in the registry not being implemented. Another limitation was found when attempting to use Mimikatz in the Powershell version and loading it into the

Evil-WinRM session. The playbooks for Attackmate are written in YAML format, which requires double quotes and a symbolic enter command in the strings to properly translate into the interactive sessions. The issue emerged from the presence of nested single and double quotes within a string surrounded by double quotes, which couldn't be translated correctly due to the limitations of the YAML format.

3 Results

In this chapter, the results are presented in four sections. First, the results of the prototype phase were highlighted, followed by the introduction of the proof-of-concept "AECID-Win-Testbed" including its implementation, with consideration of the predefined requirements and dependencies. In the third section, a functional test was conducted to verify the framework's capability to automatically deploy a testbed, execute attacks and export the raw logs. Further processing involved applying labels to the data. The resulting dataset was then reviewed and annotated with attributes in the final section, where an evaluation of the dataset was performed using a machine learning algorithm and metrics were calculated based on the findings.

3.1 Prototype

In the prototype phase several subsequent research questions were answered and the results were integrated in the automation phase. To achieve the goal of generating log datasets, completeness was ensured by capturing all relevant Windows logs, which were further enriched by deploying Sysmon. Using a WEC, all logs were collected in a centralized location and then forwarded to a Kafka broker using Winlogbeat. These components have built-in fault tolerance mechanisms that ensure all relevant logs are collected and correctly transmitted. Adaptability was achieved through a modular design that allows for simplified deployment of VM instances and configurations using Terraform and Ansible. The template-based setup of the AECID-Win-Testbed provides the core configurations and code to ensure completeness, with the possibility for extensions by extending the script collection. Thus, adjustability is also achieved through the design and the use of variables, enabling the creation of arbitrage numbers of Windows enterprise testbeds.

As of audit policies Q3, the recommendation from Microsoft is distinguished in default, baseline recommendation and strong recommendation were lastly was used for the comparison. In Appendix B two tables were presented consisting of audit policy settings for member servers from Microsoft and Palantir. A notable difference is the auditing of the category “Object Access,” which is left out by Microsoft, whereas Palantir has additional settings in this category for file share, file system, object access events and registry. As for the file share, more insight is gained into which share is accessed by whom, providing information about the source host and user accessing it. During a malicious GPO modification, the network share containing the Active Directory objects on the DC was accessed and manipulated, which is reflected in Event ID

5143, indicating a network share object was modified. Another example in the “Object Access” category is "Registry" which, when enabled, audits object access attempts. In a practical example, the Meterpreter reverse shell accessed lsass.exe to remove the process protection and generates Event ID 4663,which occurs only when "Registry" auditing is enabled. The results of the audit rule comparison highlights that Palantir’s recommendation is more effective for attack detection than Microsoft’s.

As of Q4 user simulation, the evaluation of the GHOSTS NPC Framework was conducted. As an virtual user agent is provides a rich set of features to emulate users by browsing, creating Microsoft Office documents, accessing file shares, writing mails, accessing workstations through RDP and specially allowing execution of instructions through the management server. Thus, the GHOSTS Agent can generate network traffic, applications logs and system logs resulting in diversified log patterns within the dataset. In comparison to the state machines, which mainly simulates web browsing activities in linear manner, it enhances the possibilities to generate user simulated artifacts within the logs. Another notable difference and advantage is that the GHOSTS agent can be controlled during runtime through the management server, allowing the integration of different user activities, which has the potential to create unique log artifacts, thereby challenging machine learning algorithms. The complexity of the state machines is further increased by the lack of documentation within the repository[18], whereas GHOSTS provides comprehensive documentation of the framework, including the API for the management server [35]. Additionally, GHOSTS is less complex, as only the configuration file needs to be modified after installation. The integration into the AECID-Win-Testbed was accomplished by creating two Ansible roles, one for the GHOSTS agent and the other for the GHOSTS server. Modeling of user behavior can be achieved by modifying the configuration file in the Ansible role for the GHOSTS agent and further fine-tuned through the management server.

For the Q5 cyberattacks, Windows Defender was disabled using a GPO, which is integrated into the role for the DC, allowing the execution of malicious binaries. The results of the attacks, including the corresponding IoCs are located in Appendix A.

3.2 AECID-Win-Testbed

At this point, I introduce the AECID-Win-Testbed, a framework designed for the automated creation of a Windows enterprise environment, complete with mechanisms to generate log datasets for benchmarking IDSs. Utilizing Terraform and Ansible, a testbed is deployed and configured on OpenStack. A GHOSTS management server is set up, which is tasked with integrating user models into the GHOSTS agent. Another integral component is Attackmate, which is responsible for executing the attack chain model. Through the WEC, logs are forwarded to the Kafka broker, where downstream processing occurs by applying labeling rules. Subsequently, a log dataset is produced, which can be utilized for training machine learning algorithms and benchmarking IDSs. Thus highlights on how a framework can be developed to automate the generation of log datasets.

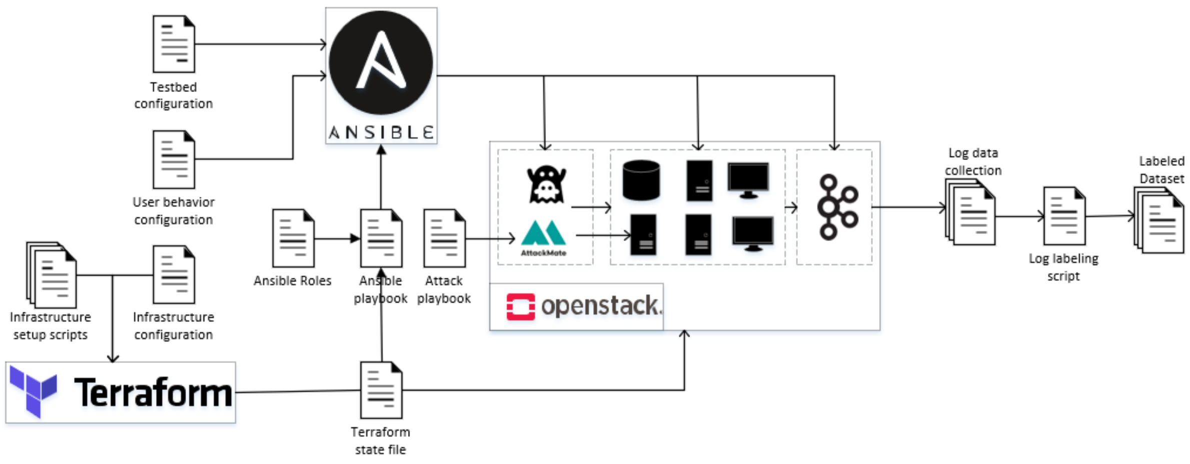


Figure 4: Framework AECID-Win-Testbed

The implementation of the framework follows the methodological approach of Model-Driven Engineering, which, in correlation with the scenario-based approach from R1, determines the use case and further narrows the scope of the desired environment while defining the dataset's limitations. The deployment of the testbed is performed on OpenStack, a private cloud hypervisor platform, aligning with the generation of synthetic log data in R2. The collection of system logs is incorporated in R3 and fully integrated by applying GPOs for audit policies and centralized log management through the WEC. Additionally, Sysmon is deployed on the Windows machines as part of the base configuration to enhance auditing capabilities and ensure log completeness. The integration of user behavior is outlined in R4 and realized through the integration of the GHOSTS NPC framework, which is described in Section 3.1. The evaluation of attacks in Appendix A was performed due to R5, which focuses on executing relevant attacks. Therefore, the cyber kill chain, discussed in Section 3.3, was selected based on the relevance of the attacks, such as the preference for the "diamond ticket" attack. During the establishment of the cyber kill chain, an evaluation of Attackmate for Windows was also conducted. This evaluation involved multiple iterations, where the playbook was extended task by task, and the destination was reset to a previous state after each iteration. This process is in flavor with R10, dataset reproducibility, and also aligns with R6, which focuses on attack repetitions, demonstrating that Attackmate is a feasible tool for meeting this requirements. Attack variations are addressed in R7, which is also fulfilled by using Attackmate for attack orchestration. Due to a YAML limitation, the execution of Mimikatz binaries was replaced by the kiwi module in Metasploit, showcasing that variations of Mimikatz can be utilized, a principle that applies to other techniques as well. This justifies the integration of Attackmate into the AECID-Win-Testbed. Labeling ground truth, as stated in R8, is achieved by querying the raw data. The output of Attackmate assists in narrowing down the size of the logs that need to be queried. Using Winlogbeat, which sends logs in JSON format, along with a Kafka broker to consume the logs, streamlines the extraction process to a designated location. Since JSON allows for easy manipulation through scripts, the queries do not require domain knowledge specific to an ELK stack, as discussed in [19]. Therefore, any preferred scripting language can be used to apply labels. An outlook on the labeling process in this thesis can be found in Section 3.3. Requirement R9 focuses on documentation, which is necessary for the public to understand the artifacts in the dataset. This is addressed by describing the models used in Section 3.3. Additional documentation is provided in each Ansible role, including comment blocks within the scripts to offer guidance for reproducibility. The final requirement, R10, concerns the reproducibility of datasets. This is ensured through the use of code for streamlined provisioning, deployment, configuration, user simulation and attack execution.

Distinction The notable difference in the technical implementation, compared to [6], [17] and [19], lies in the processing approach. While there are similarities, TIMs in this thesis serve as templates and act as repositories of multiple models represented as code. The so-called transformation engine in this thesis handles the correct processing of variables and configurations, which then lead to TSMs. Furthermore, utilizing PowerShell through Ansible was required for configuration of domain preferences which, unlike related work, adds an additional task to the model and testbed layer. The deployment of the environment is also handled differently as Packer was not used because of the difference in the underlying environment, requiring the sequential deployment of machines rather then parallel deployment as for Windows enterprises. Regarding Q6, which elements of related work can be reused, the concept and architecture follow a similar approach. However, the code used in the "as-a-code" tooling was self-developed and throughout this thesis, replacements were made for the user simulation and attack automation aspects. Although AIT also recognized the benefits of the GHOSTS NPC framework, they integrated it into their project and published the role "atb-ansible-ghostserver" [42] forehand.

3.2.1 Dependencies

The Ansible-roles in Appendix C has to be executed in the correct order to establish the environment, which necessitated a clear visualization of the dependencies within the environment, as illustrated in Figure 5. In Ansible, roles can be executed in a play, offering the flexibility to further decompose the server roles into member servers and services. The initial domain setup must be carried out with domain admin credentials, including a username and password. These credentials will also be used for the domain joins of servers and clients, as well as for the promotion of any additional domain controller. An additional note regarding the GPOs is that some may need to be converted using migration tables if the new environment has a different domain name. This process is simplified by passing a variable to the Ansible role. Furthermore, the IP address of the Kafka broker must be known before the WEC can begin shipping logs. The same applies to workstations with an active GHOSTS Agent, which must be configured with the

IP address of the GHOSTS server in its configuration file if an dynamic interaction during run is desired.

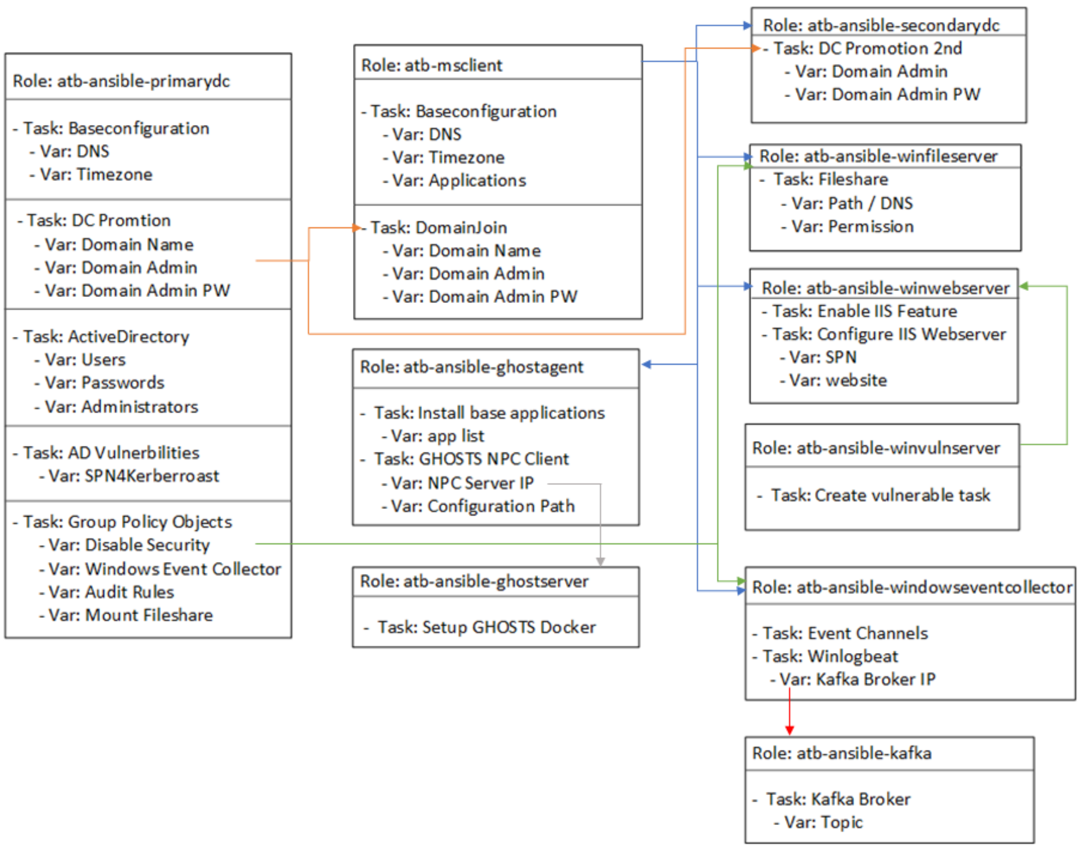


Figure 5: Dependencies

3.3 Functional Testing of the Framework

In this section, the framework "AECID-Win-Testbed" underwent functional testing. Building upon the concept outlined in Figure 3, initial infrastructure modeling, user simulation and attacks modeling were designed following the methodology of MDE. Subsequently, infrastructure provisioning was performed, followed by manual review to ensure proper functionality. The subsequent step involved executing the automated attack chain, followed by extracting the resulting log data.

Infrastructure & User Model The topology was built using the core components of the framework, consisting of a domain controller, Windows Event Collector, Kafka broker and clients with an active NPC agent. This infrastructure was expanded to include a web and file server. The rationale behind the web server is that the initial basic function of the NPC agent is web browsing. This configuration was expanded and the web server was also designated as a predefined destination for the NPCs. The same rationale applies to the file server, as NPCs also have the functionality to work on office documents and save them to file shares. Another reason is the active use of shares to generate the corresponding logs. Otherwise, attacks targeting the file share of the domain controller would represent an anomaly based on their existence and would be detectable without much logic. Accordingly, algorithms must be challenged and it is necessary to generate a benign log entry for every malicious log entry for that particular scenario [1].

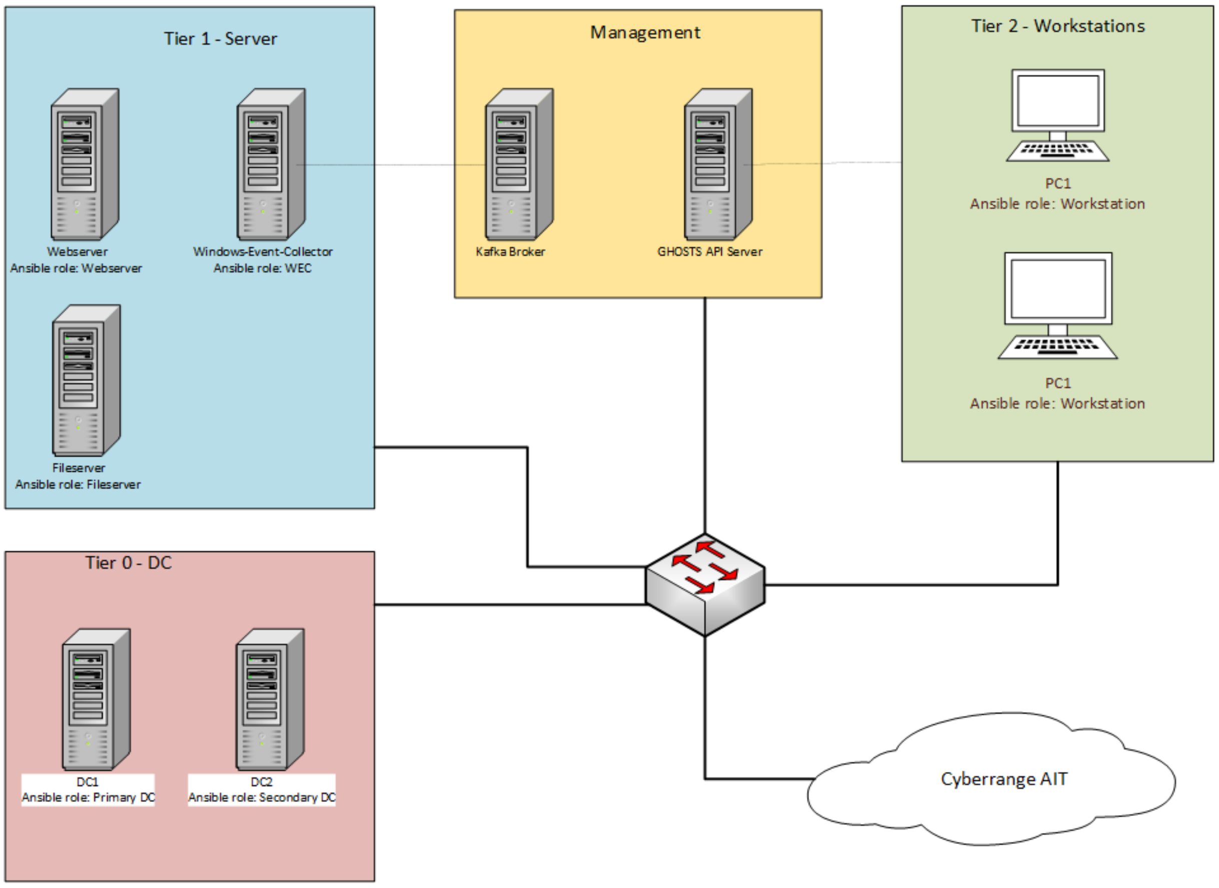


Figure 6: Topology Model

Attackchain Model Initially, it is defined that the credentials of a non-privileged user are already known and the attacker's machine has existing access to the infrastructure. Consequently, the exploitation of a public-facing service is considered beyond the scope of this model. The attack begins by compromising a user account for which the credentials are already known. The initial foothold to the target system is gained through WinRM. This method is partially chosen due to the necessity of an active WinRM session for configuring Windows hosts via Ansible. The target system is then subjected to various privilege escalation and persistence techniques, executed through tools commonly used by adversaries. These techniques are complemented by the extraction of operating system credentials and thorough enumeration of the target host. Following this, the attack strategy involves lateral movement to penetrate the dc within the network, leading to the exfiltration of the Active Directory database. Persistence at the domain level is achieved by spawning a rogue domain controller into the environment. Furthermore, conventional attacks on the Kerberos authentication protocol are conducted to round out the attack scenario by gaining domain admin privileges.

The provisioning of the infrastructure was then carried out using Terraform and Ansible. Scripts and configurations can be found in Appendix C. Despite the scripts executing flawlessly, each component was manually verified for functionality. Subsequently, Attackmate was employed to conduct the attacks based on the scenario outlined above. The playbook used is available in Appendix C and also executed without errors. Finally, the dataset was extracted from Kafka. Thus, the functional testing of the framework was successfully completed. In the following chapters, the extracted dataset from this chapter will be further evaluated to determine if it meets the requirements for machine learning algorithms.

Labeling Labeling the log data is defined in R8 and displays the ground truth of the dataset. In the subsequent script block, a template is found for log labeling. The process itself is divided into two steps where in the first step the labeled logs were copied in a new file and the corresponding event record IDs in another file. With the record IDs, the original file is labeled subsequently. This procedure gives the advantage of having the malicious logs labeled in the dataset file and an additional file containing only labeled logs for review.

As for the queries, different approaches were utilized to archive precise labeling. Based on the IoC's in Appendix A, logs were queried for example by event ID or by values of an attribute. This task is simplified because the attacks were already analyzed and evaluated. The logic behind the labeling for attributes is based on known executable names, logon sessions and GUID's, copied files based on file name, object manipulations based on source image and running the vulnerable task. By filtering the unique record IDs of these events, in a second script the raw data is labeled by matching the IDs from the queries and adding the label "malicious".

```powershell
### query and filter raw data
# Define variables
$jsonPath = "$($PSScriptRoot)\Dataset_Payload.json"
$jsonQueried = $($PSScriptRoot)\QueryX.json
$recordID = "$($PSScriptRoot)\RecordIDs.txt"

# Read JSON file content
$jsonContent = Get-Content -Path $jsonPath | ConvertFrom-Json

# Filter content | example: query all events with ID 11
$eventX = $jsonContent | Where-Object { ($_.winlog.event_id -eq '11') }

# Write the filtered and labeled events in a new file
Add-Content -Path $jsonQueried -Value ($eventX | ConvertTo-Json)

# Add the Event record ID's to an additional file
Foreach ($event in $eventX) {
    Add-Content -Path $recordID -Value $event.winlog.record_id
}
```

```powershell
### label Dataset based on query results
# Define variables
$jsonPath = "$($PSScriptRoot)\Dataset_Payload.json"
$jsonLabeled = "$($PSScriptRoot)\Dataset_Labeled.json"
$recordID = "$($PSScriptRoot)\RecordIDs.txt"

# Read the file contents
$jsonContent = Get-Content -Path $jsonPath | ConvertFrom-Json
$recordIDs = Get-Content -Path $recordID

# Add label "malicious" as property
$jsonContent | Where-Object {
    if ( $recordIDs -contains $_.winlog.record_id ) {
        $_ | Add-Member -MemberType NoteProperty -Name "malicious" -Value
    $true
    }
}

# Output multi root JSON file
foreach ($event in $jsonContent) {
    Add-Content -Path $outputPath -Value ($event | ConvertTo-Json -Compress)
}
```

3.4 Dataset Evaluation

Evaluating the dataset started by splitting the log files into a training and testing set. The trainings dataset contains 223k logs which were recorded over 24 hours with benign system interactions. 2865 logs were collected during the attack phase in a time span of 15 minutes which considering the time along with the quantity of logs, the characteristics of equal distribution is shown. The dataset contains 13 unique log providers and 17 different tags, which specify the event channel of the log source. As of the nature of event log records, it may appear that the event record ID may not be unique if the collection of logs is over a longer time period. Therefore, for unique assignment of logs the values of hostnames and timestamps may be included [43].

An assessment of the automated log dataset generated from the AECID-Win-Testbed was performed using the logdata-anomaly-miner [20], further abbreviated as AMiner, a supervised machine learning algorithm developed by AIT. It contains the capability to parse the log data and provides analysis pipelines for anomaly detection with configurable detectors which is utilized to train the algorithm including checking for efficiency and resulting in the feasibility of the dataset itself. For the configuration of the parser, a feature engineering phase was conducted also considering the different detection capabilities by AMiner.

3.4.1 Detectors

As for the the detectors, the AMiner was setup with three different detectors. The first detector used is the analysis component "NewMatchValueComboDetector" which detects new value combination in in specified paths. The corresponding features used here is event ID and hostname. Therefore, for the first analysis component the combination of event IDs per host was performed, with the intention to detect anomalies based on whenever a host generates an event which it didn't do in the training phase. The anomalies found was adding Alice to the administrator group, the network connection created by the Meterpreter reverse shell, the installation of services for privilege escalation and Mimikatz. The same component was used with the features of source- and target-image, parsed from Sysmon Event ID 8, to detect process injection.

The second detector utilized is named "NewMatchValueDetector" which learns about the value of single variables and creates anomalies based on new occurrences. The analysis component was setup to identify anomalous IP addresses which were pre-processed to only contain internal IPs. Intending to identify all events generated which the IP of the attacker machine, this detector had a 100% true positive rate and found the initial foothold but also any additional connection established from the attacker for example the access of the network share during dcsync. The same component was used with the feature "CommandLineLength" which was processed from the raw data and the length of process executions was determined using an integer value. All invocations from the location of C:\\Temp were found as the value is very unlikely to match does from locations like program files which also had high accuracy in form of true positives.

Lastly as third detector the "EntropyDetector" was used which calculates the sequence of characters and estimates which characters are likely to follow. As feature the unparsed process invocation was considered to effectively find anomalous executions using obfuscated strings. Here the value of the attribute "CommandLine" was extracted from Sysmon Event ID 1. All anomalies found with AMiner were plotted into the training-dataset and is displayed in figure 7.

To summarize the findings, they demonstrate the efficacy of the AMiner algorithm in detecting anomalies within the dataset. The use of the "NewMatchValueComboDetector" revealed significant anomalies, such as unauthorized access and suspicious network activities. The "NewMatchValueDetector" successfully identified malicious IP addresses and anomalous command line executions. The "EntropyDetector" effectively detected obfuscated process invocations.

3.4.2 Findings

In this section, the findings from AMiner are correlated with the labeled logs and further divided into correct, incorrect and missing detections. First, during the labeling process an amount of 153 logs were labeled. By omitting similar entries a final number of 57 malicious logs are used for further calculations. The findings of the detectors are presented in Table 1 which found 38 log entries, again after omitting similar entries. Further dividing the logs shows 23 true positive and 15 false positive events which also results in 34 false negative and leaving 2213 true negative entries. By applying those metrics into the calculation used for dataset evaluations [44],[45], the precision 1, recall 2 and F1-Score 3 is presented.

Detector	True Positiv	False Positive
Name_Event_Combo	6	7
Source_Target_Combo	1	0
EntropyDetector	5	2
CMD_Length_Detector	10	6
IP_Address_Detector	1	0

Table 1: Findings per detector

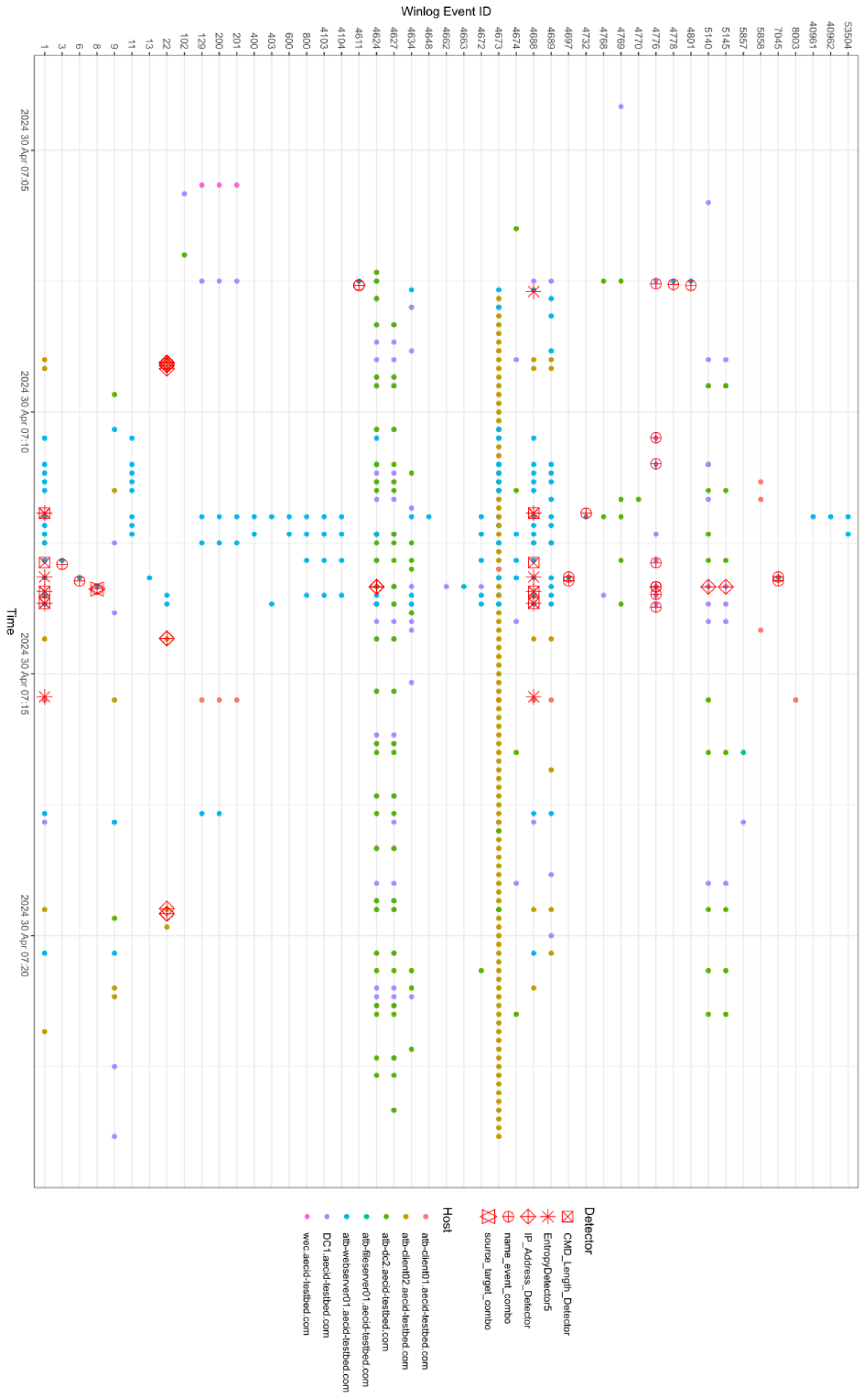


Figure 7: Anomaly plot

$$Precision = \frac{TP}{TP + FP} \approx 59.5\% \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \approx 38.6\% \tag{2}$$

$$F1Score = \frac{2 * Precision * Recall}{Precision + Recall} \approx 46.8\% \tag{3}$$

The scores are promising including potential enhancements by implementing additional detectors and refined feature engineering of raw logs. This highlights the usability of the AECID-Win-Testbed for the creation of datasets for the training of supervised machine learning algorithms.

4 Discussion

This thesis presents a set of design principles and requirements for creating a framework for the deployment of a security focused testbed to generate log datasets in an automated manner specific for Windows enterprises. Building upon these principles and requirements, the AECID-Win-Testbed is introduced as a proof of concept for the automated deployment of a Windows environment, utilizing user simulations, including automated attacks and generating log datasets. The framework illustrates that the input is defined by creating testbed-independent models through planning a model and defining the variables and configurations for it. This models are then translated through a transformation engine, allowing the creation of a testbed-specific model for the deployment of the desired cyberrange. Additionally, playbooks for attacks on a Windows environment are published, showcasing how an automated attack chain is deployed using Attackmate. Through the integration of different methodologies, all Windows logs from different machines are collected at a designated location and then shipped to the Kafka broker using Winlogbeat, which offers the advantage of forwarding the logs in JSON format and according to the elastic-schema. Subsequently, further processing of the logs takes place to label the ground truth, resulting in a log dataset for the training of supervised machine learning algorithms. The dataset was assigned with attributes for documentation purpose and examined in the evaluation phase. Further, during this phase, it was used to train an algorithm with benign logs and the tested with malicious content. The algorithm was setup with five different configurations and the evaluation of the executions were presented, showing that the generated dataset by the AECID-Win-Testbed framework is utilizable.

4.1 Main Findings

For Q1, MDE was applied as the methodology in this work and was a suitable approach. The modeling of the testbed, user simulation and attacks implies fulfilling the predefined requirements in Section 2.2.2 and also follows the general approach of generating scenario-based datasets outlined in [22]. The transformation engine from MDE is utilized in this work through dynamic configuration of the implementation to correctly translate variables and configuration files. However, in terms of infrastructure, the framework only provides a template for configuring the infrastructure, consisting of core components and domain configurations, which are outlined in Section 3.1.

This leads to Q2. The infrastructure can be considered as static but can be extended and adjusted according to the defined use cases. Therefore, regarding the infrastructure, configurations can be applied through the framework to deploy a server with the necessary setup, but the services on it must be configured independently by an individual. Consequently, the self-created Ansible role as example can be included as an additional role in a playbook alongside the roles from this framework. Furthermore, the responsibility for ensuring completeness lies with the individual defining the modeled scenario, in regards of the underlying environment.

The configuration mentioned in the previous paragraph includes the audit rules for Q3, which were evaluated through a comparison of best practices. The chosen practise is confirmed by the executed attacks to be valid. Sysmon was also deployed, providing more security-relevant event logs and ensuring completeness of auditing.

For Q4, the GHOSTS-NPC framework was considered and integrated into the AECID-Win testbed. Therefore, the deployment of both NPC agents and the NPC server occurs automatically and the NPCs are modeled by adjusting the configuration file in the corresponding Ansible role. An additional advantage arises from the client-server architecture of GHOSTS. This allows for the adaptation of simulated users even after the deployment of the testbed. Thus, a new deployment of the testbed is not necessarily required for adopting the user simulation behavior and can be configured through the NPC server.

For Q5, during the evaluation of the attacks the manifestation was analyzed and is found in Appendix A 2.3.2. Here, IoCs for the respective attacks were researched. Furthermore, following the principles of MDE, Attackmate was used to model a cyber kill chain and execute it automatically. The use of Attackmate also aligns with the defined requirements in Section 2.2.2.

For Q6, the methodological approach of MDE and the recommended DevOps tooling were adopted and applied to the Windows scenario. Further improvements were made in the areas of user simulation and attack automation by utilizing GHOSTS and Attackmate. The elements design and architecture were further optimized towards a Windows enterprise environment [6],[17], [19].

4.2 Comparison with Previous Work

This implementation represents a further development of "Have it your Way" [6], "Maintainable log datasets" [17] and "Kyoushi" [19] and aligns with the model-driven engineering approach. In comparison to the related work, simplifications and optimizations have taken place. One component in Kyoushi is the "Kyoushi Simulation", which is a state machine execution library written in Python responsible for user and attack simulation. The same state machines were used in [6] and [17]. For the user simulation, the state machine approach was replaced with the GHOSTS NPC Framework, adding more dynamic user behavior to the testbed, resulting in more realistic interactions within the environment. This is also reflected in the logs, aligning with the production of user emulated event patterns and enriches them by adding multiple artifacts.

For the attack simulation in this thesis, the tool Attackmate is used. Unlike the state machines, it isn’t fully integrated into the testbed and requires an additional deployment of an Ubuntu VM including the execution of an Ansible role but simplifies the execution of the attacks, given advantage of simplified attack repetition and variations. Another component that was further improved is the Kyoushi Dataset. In its design, an ELK-Stack was deployed. As stated in the paper, "Internally it uses the log aggregation and parsing tool Logstash to parse unstructured log data and the search and analytics engine Elasticsearch as its processing database" [19, p. 45]. Due to the complexity of that solution, in the AECID-Win-Testbed, a Kafka broker is deployed, which simplifies the log collection and provides an output in JSON format where further labeling rules are applied. Thus, the labeling can be employed based on domain knowledge as manipulating JSON files is streamlined in scripting languages. Additionally, Kafka supports multiple message queues, providing an opportunity to facilitate the log collection of multiple log sources, mimicking SIEM functionalities. However, the labeling approach in the previous works was given more priority and utilizing a multi-labeling approach rather the binary classification as in this thesis.

Apart from improvements in user simulation and attack orchestration, a significant difference lies in the infrastructure setup. In related work, provisioning is completed in one go using additional technologies like Packer, which offers Imaging-as-Code. In this thesis, VMs are deployed through Tofu, and the configuration of instances must be done separately for each one. This is due to dependencies that arise when deploying a Windows enterprise environment, where a domain join can only be executed once the domain already exists. This approach results in a moderately higher workload during provisioning but allows for greater extensibility. Ansible roles can either be derived and executed from their repository or developed independently based on the specific scenario. In summary, a trade-off was made to achieve simplification and extensibility, resulting in slightly more workload for deployments.

4.3 Implications

The AECID-Win-Testbed is built upon the pre-defined requirements and implemented using the approach of model-driven engineering, which implies a practical framework for the automated creation of log datasets for Windows enterprises with a focus on realistic scenarios. Thus, the framework offers a template-based approach to deploy various Windows environments and create diverse datasets through a predefined scope. The configuration allows for the definition and representation of various scenarios. Furthermore, the templates could be extended in the future through a community-based approach and maintained in a repository.

Another implication is the intended use of the framework. The provisioning of the Windows environment can also influence the setup of testbeds for security analysts. These analysts can utilize the documented scripts from this work to automatically set up labs, which can be used for training purpose and testing detection capabilities. Consequently, an evaluation of IDS can also be performed by deploying the testbed, leveraging the capability for rapid generation of multiple testbeds with varying IDS solutions. This enables a comparative analysis of their performance and effectiveness by provisioning multiple identical testbeds isolated from each other, deploying an IDS in each testbed and measure their capability to detect attacks executed. Additionally, centralized log management enables exercises between Red and Blue teams to be conducted. Therefore, it can be generally employed for training purposes.

4.4 Limitations

The major limitation of this framework is the aspect of patching. Applying software patches may cause some services or processes to behave differently and the challenge is that it is difficult to foresee exactly what furture updates will contain. Thus, in the future, it may be necessary to troubleshoot the environment. During the evaluation of the components, for example, it was observed that the latest Chrome browser was available in the Chocolatey repository, but the corresponding Chrome driver, which is facilitated by the GHOSTS framework, was not. While this was resolved by defining the software versions in the responsible Ansible role, similar situations may arise and outlines the continues development of the framework itself. Another consideration directly linked to patching is the aspect of closing security vulnerabilities where the executed playbook for automated attack execution may have to be adopted.

Another shortcoming of this thesis is the precision of the labeling approach. During the evaluation of the dataset, some events were discovered by AMiner and tagged as malicious which are in the context true positives and weren't labeled before. This on one hand shows the efficacy of AMiner for anomaly detection but on the other hand highlighting the challenges of precise labeling. Sequential logging is the nature of Windows Event Logs and an interaction like process execution creates multiple entries in different log providers which raises the difficulty to label all relevant logs for an malicious interaction on a Windows host. However, like in a SIEM, the most important aspect of labeling is to mark logs which identify an IoC which is given in this thesis.

Another area for improvement is the user simulation component. While a robust framework was identified and integrated, its current usage is restricted to fundamental functions like web browsing. However, it also facilitates the utilization of machine learning and the formation of various profiles, potentially enhancing simulation behavior significantly and establishing a more realistic environment. Further enhancements could involve integrating administrative tasks, enabling remote connections, incorporating a mail server component and utilizing email communication within the environment. Ultimately, it allows also the representation of insider threats. This would diversify network traffic with various protocols and introduce additional user generated log patterns.

4.5 Future Work

The evaluation of the dataset is conducted to the best of knowledge and ability but also requires analysis by specialists. While the corresponding logs for the attacks were found and narrowed down to some specific logs, the additional log artifacts that emerged were not reflected in the identified IoCs. However, due to the use of AMiner, further log artifacts related to the attacks were discovered, some of which cannot be confidently attributed to certain traces and would require review by a specialist. For following projects the labeling approach my be refined by developing a method to auto correlate the automated attacks with those from the Windows Logs and following system traces for more precise labeling. Another consideration involves anomaly detection utilizing log sequences, which necessitates feature engineering. Windows logs follows a sequence-based structure, delineating the processes that produced the logs and subsequent processes invoked. This series of sequences requires comprehensive analysis to appropriately configure machine learning algorithms. This outlines the necessity for a method dedicated to log labeling.

Also for future work I suggest to further enhance the framework by connecting the testbed to cloud services provided by Microsoft. As cloud technologies are state of the art and continuously increasing in services and capabilities, most enterprises run in a hybrid environment. Thus, more scenarios can be covered ultimately reflecting in further datasets containing mixed patterns with logs collected from cloud and on premise environments. A drawback of this recommendation is the aspect of costs occurring by using cloud services.

Bibliography

[1] I Sharafaldin, A Lashkari, and A Ghorbani. “Toward generating a new intrusion detection dataset and intrusion traffic characterization.” In: *Intl Conf. on Information Systems Security and Privacy (ICISSP)*. Vol. 1. Jan. 2018, pp. 108–116. DOI: https://doi.org/10.5220/0006639801080116.

[2] Waqas Haider et al. “Windows Based Data Sets for Evaluation of Robustness of Host Based Intrusion Detection Systems (IDS) to Zero-Day and Stealth Attacks”. In: *Future Internet* 8.3 (2016). ISSN: 1999-5903. DOI: 10.3390/fi8030029. URL: https://www.mdpi.com/1999-5903/8/3/29.

[3] Max Landauer et al. “Dynamic log file analysis: An unsupervised cluster evolution approach for anomaly detection”. In: *Computers & Security* 79 (2018). PII: S0167404818306333, pp. 94–116. ISSN: 01674048. DOI: 10.1016/j.cose.2018.08.009. URL: https://repositum.tuwien.at/handle/20.500.12708/145265?mode=full.

[4] A. Kenyon, L. Deka, and D. Elizondo. “Are public intrusion datasets fit for purpose characterising the state of the art in intrusion event datasets”. In: *Computers & Security* 99 (2020), p. 102022. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2020.102022. URL: https://www.sciencedirect.com/science/article/pii/S0167404820302959.

[5] Nour Moustafa et al. “Federated TON_IoT Windows Datasets for Evaluating AI-Based Security Applications”. In: *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. 2020, pp. 848–855. DOI: 10.1109/TrustCom50675.2020.00114.

[6] Max Landauer et al. “Have it Your Way: Generating Customized Log Datasets With a Model-Driven Simulation Testbed”. In: *IEEE Transactions on Reliability* 70.1 (2021), pp. 402–415. DOI: 10.1109/TR.2020.3031317.

[7] CyberArk. *EP 44 - The Rise of Prompt Engineering: How AI Fuels Script Kiddies*. Sept. 2024. URL: https://www.cyberark.com/podcasts/ep-44-the-rise-of-prompt-engineering-how-ai-fuels-script-kiddies/ (visited on 09/17/2024).

[8] Ankit Thakkar and Ritika Lohiya. “A Review of the Advancement in Intrusion Detection Datasets”. In: *Procedia Computer Science* 167 (2020). International Conference on Computational Intelligence and Data Science, pp. 636–645. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2020.03.330. URL: https://www.sciencedirect.com/science/article/pii/S1877050920307961.

[9] *CVE-2021-42287 - Security Update Guide - Microsoft - Active Directory Domain Services Elevation of Privilege Vulnerability*. Mar. 2024. URL: https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-42287 (visited on 03/01/2024).

[10] *CVE-2021-42278 – Leitfaden für Sicherheitsupdates – Microsoft - Active Directory Domain Services Elevation of Privilege Vulnerability*. Sept. 2024. URL: https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-42278 (visited on 03/01/2024).

[11] Charlie Clark. "A Diamond Ticket in the Ruff". In: *Semperis* (July 2022). URL: https://www.semperis.com/blog/a-diamond-ticket-in-the-ruff/ (visited on 03/01/2024).

[12] Gideon Creech. "Developing a high-accuracy cross platform Host-Based Intrusion Detection System capable of reliably detecting zero-day attacks". In: 2014. URL: https://api.semanticscholar.org/CorpusID:12873377.

[13] MIT Lincoln Laboratory DARPA. *KDD Cup 1999 Data*. The Fifth International Conference on Knowledge Discovery and Data Mining. 1999. URL: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (visited on 04/15/2024).

[14] StrGenIx | Laurens D'hooge. *CIC-MalMem2022*. Apr. 2023. URL: https://www.kaggle.com/datasets/dhoogla/cicmalmem2022 (visited on 08/12/2024).

[15] GitHub. *sbousseaden/EVTX-ATTACK-SAMPLES: Windows Events Attack Samples*. 2024. URL: https://github.com/sbousseaden/EVTX-ATTACK-SAMPLES/tree/master (visited on 04/15/2024).

[16] Roberto Rodriguez. *Windows — Security Datasets*. Sept. 2022. URL: https://securitydatasets.com/create/windows.html (visited on 04/15/2024).

[17] Max Landauer et al. "Maintainable Log Datasets for Evaluation of Intrusion Detection Systems". English. In: *IEEE Transactions on Dependable and Secure Computing* (2022), pp. 1–17. ISSN: 1545-5971. DOI: 10.1109/TDSC.2022.3201582.

[18] GitHub. *ait-aecid/kyoushi-statemachines: User and attacker statemachines for simulation in testbeds*. Feb. 2024. URL: https://github.com/ait-aecid/kyoushi-statemachines (visited on 02/17/2024).

[19] Maximilian Frank. "Quality improvement of labels for model-driven benchmark data generation for intrusion detection systems". Technische Universität Wien, 2021. URL: 10.34726/hss.2021.82646.

[20] GitHub. *ait-aecid/logdata-anomaly-miner: This tool parses log data and allows to define analysis pipelines for anomaly detection. It was designed to run the analysis with limited resources and lowest possible permissions to make it suitable for production server use*. 2024. URL: https://github.com/ait-aecid/logdata-anomaly-miner (visited on 04/15/2024).

[21] GitHub. *ait-testbed/attackbed: The AttackBed is a simulated enterprise network with numerous vulnerabilities. Attacks in this testbed are executed automatically and cover a variety of tactics and techniques of the MITRE ATT&CK enterprise framework*. Aug. 2024. URL: https://github.com/ait-testbed/attackbed/tree/main (visited on 08/19/2024).

[22] Muhammad Mudassar Yamin, Basel Katt, and Vasileios Gkioulos. “Cyber ranges and security testbeds: Scenarios, functions, tools and architecture”. In: *Computers & Security* 88 (2020), p. 101636. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2019.101636. URL: https://www.sciencedirect.com/science/article/pii/S0167404819301804.

[23] Maria Leitner et al. “AIT Cyber Range: Flexible Cyber Security Environment for Exercises, Training and Research”. In: *Proceedings of the 2020 European Interdisciplinary Cybersecurity Conference*. EICC ’20. Rennes, France: Association for Computing Machinery, 2021. ISBN: 9781450375993. DOI: 10.1145/3424954.3424959.

[24] Konstantin Berlin, David Slater, and Joshua Saxe. “Malicious Behavior Detection using Windows Audit Logs”. In: *CoRR* abs/1506.04200 (2015). arXiv: 1506.04200. URL: http://arxiv.org/abs/1506.04200.

[25] Ferhat Ozgur Catak et al. “Data augmentation based malware detection using convolutional neural networks”. In: *PeerJ Computer Science* 7 (Jan. 2021), e346. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.346. URL: https://doi.org/10.7717/peerj-cs.346.

[26] Markus Ring et al. “Malware detection on windows audit logs using LSTMs”. In: *Computers & Security* 109 (2021), p. 102389. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2021.102389. URL: https://www.sciencedirect.com/science/article/pii/S0167404821002133.

[27] J. Benito Camiña et al. “The Windows-Users and -Intruder simulations Logs dataset (WUIL): An experimental framework for masquerade detection mechanisms”. In: *Expert Systems with Applications* 41.3 (2014). Methods and Applications of Artificial and Computational Intelligence, pp. 919–930. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2013.08.022. URL: https://www.sciencedirect.com/science/article/pii/S0957417413006349.

[28] Huu-Khoi Bui et al. “CREME: A toolchain of automatic dataset collection for machine learning in intrusion detection”. In: *Journal of Network and Computer Applications* 193 (2021), p. 103212. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2021.103212. URL: https://www.sciencedirect.com/science/article/pii/S1084804521002137.

[29] Carlos Garcia Cordero et al. “On generating network traffic datasets with synthetic attacks for intrusion detection”. In: *CoRR* abs/1905.00304 (2019). arXiv: 1905.00304. URL: http://arxiv.org/abs/1905.00304.

[30] W. Haider et al. “Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling”. In: *Journal of Network and Computer Applications* 87 (2017), pp. 185–192. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2017.03.018. URL: https://www.sciencedirect.com/science/article/pii/S1084804517301273.

[31] *OpenTofu*. Aug. 2024. URL: https://opentofu.org/ (visited on 08/15/2024).

[32] “Chef vs. Puppet vs. Ansible: a side-by-side comparison for 2024”. In: *Better Stack* (Feb. 23, 2023). URL: https://betterstack.com/community/comparisons/chef-vs-puppet-vs-ansible/ (visited on 04/15/2024).

[33] *Windows Event Forwarding Guidance*. original-date: 2017-09-05T19:54:17Z. Oct. 19, 2023. URL: https://github.com/palantir/windows-event-forwarding (visited on 10/20/2023).

[34] crowdstrike.com. *Windows Logging Guide Part 4: Centralizing Logs*. Aug. 2024. URL: https://www.crowdstrike.com/guides/windows-logging/centralizing-logs/ (visited on 04/11/2024).

[35] Dustin Updyke. *GHOSTS*. Version 7.0.0. URL: https://github.com/cmu-sei/GHOSTS (visited on 02/17/2024).

[36] Apache Kafka. *Apache Kafka*. Apr. 2024. URL: https://kafka.apache.org/documentation/#gettingStarted (visited on 04/15/2024).

[37] crowdstrike.com. *Windows Logging Guide: Advanced Concepts - CrowdStrike*. Oct. 2023. URL: https://www.crowdstrike.com/guides/windows-logging/advanced-concepts/ (visited on 04/11/2024).

[38] *Winlogbeat Overview | Winlogbeat Reference [8.13] | Elastic*. Mar. 2024. URL: https://www.elastic.co/guide/en/beats/winlogbeat/current/_winlogbeat_overview.html (visited on 04/15/2024).

[39] GitHub. *ait-aecid/attackmate: AttackMate is an attack orchestration tool that executes full attack-chains based on playbooks*. Feb. 2024. URL: https://github.com/ait-aecid/attackmate (visited on 02/17/2024).

[40] GitHub. *Support for local interactive command · Issue #65 · ait-testbed/attackmate*. 2024. URL: https://github.com/ait-testbed/attackmate/issues/65 (visited on 04/08/2024).

[41] GitHub. *msf-module verbose mode · Issue #70 · ait-testbed/attackmate*. 2024. URL: https://github.com/ait-testbed/attackmate/issues/70 (visited on 04/08/2024).

[42] GitHub. *Release v1.0.0 · ait-testbed/atb-ansible-ghostserver*. Aug. 2024. URL: https://github.com/ait-testbed/atb-ansible-ghostserver/releases/tag/v1.0.0 (visited on 08/28/2024).

[43] Karl-Bridge-Microsoft. *Event Log Records - Win32 apps*. Feb. 2024. URL: https://learn.microsoft.com/en-us/windows/win32/eventlog/event-log-records (visited on 07/26/2024).

[44] Ilhan Firat Kilincer, Fatih Ertam, and Abdulkadir Sengur. “Machine learning methods for cyber security intrusion detection: Datasets and comparative study”. In: *Computer Networks* 188 (2021), p. 107840. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2021.107840. URL: https://www.sciencedirect.com/science/article/pii/S1389128621000141.

[45] Siamak Layeghy, Marcus Gallagher, and Marius Portmann. “Benchmarking the benchmark — Comparing synthetic and real-world Network IDS datasets”. In: *Journal of Information Security and Applications* 80 (2024), p. 103689. ISSN: 2214-2126. DOI: https://doi.org/10.1016/j.jisa.2023.103689. URL: https://www.sciencedirect.com/science/article/pii/S2214212623002739.

[46] jpcertcc. *WinRM*. Dec. 2017. URL: https://jpcertcc.github.io/ToolAnalysisResultSheet/details/WinRM.htm#DestinationDetails (visited on 02/17/2024).

[47] *PEASS-ng/winPEAS at master · carlospolop/PEASS-ng*. Feb. 2024. URL: https://github.com/carlospolop/PEASS-ng/tree/master/winPEAS (visited on 02/17/2024).

[48] *HackTool - winPEAS Execution*. Feb. 2024. URL: https://detection.fyi/sigmahq/sigma/windows/process_creation/proc_creation_win_hktl_winpeas/?query=winpeas (visited on 02/17/2024).

[49] GitHub. *PowerSploit/Privesc/README.md at master · PowerShellMafia/PowerSploit*. Feb. 2024. URL: https://github.com/PowerShellMafia/PowerSploit/blob/master/Privesc/README.md (visited on 02/17/2024).

[50] *CVE-2019-1069 - Security Update Guide - Microsoft - Task Scheduler Elevation of Privilege Vulnerability*. Feb. 2024. URL: https://msrc.microsoft.com/update-guide/en-US/advisory/CVE-2019-1069 (visited on 02/17/2024).

[51] Alvinashcraft. *Access control lists - Win32 apps*. Feb. 2024. URL: https://learn.microsoft.com/en-us/windows/win32/secauthz/access-control-lists (visited on 02/17/2024).

[52] Vinaypamnani-msft. *File System (Global Object Access Auditing) - Windows Security*. Feb. 2024. URL: https://learn.microsoft.com/en-us/windows/security/threat-protection/auditing/file-system-global-object-access-auditing (visited on 02/17/2024).

[53] Jai Minton. *MITRE ATT&CK™ Analysis - T1037.001 Logon Script (Windows)*. 2019. URL: https://www.jaiminton.com/Mitreatt&ck/T1037-001# (visited on 02/17/2024).

[54] MITRE. *Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder, Sub-technique T1547.001 - Enterprise | MITRE ATT&CK®*. Jan. 2024. URL: https://attack.mitre.org/techniques/T1547/001/ (visited on 02/17/2024).

[55] GitHub. *gentilkiwi/mimikatz: A little tool to play with Windows security*. Mar. 2024. URL: https://github.com/gentilkiwi/mimikatz (visited on 03/01/2024).

[56] GitHub. *impacket/examples/secretsdump.py at master · fortra/impacket*. Jan. 2024. URL: https://github.com/fortra/impacket/blob/master/examples/secretsdump.py (visited on 02/17/2024).

[57] HackTricks. *Mimikatz*. Mar. 2024. URL: https://book.hacktricks.xyz/windows-hardening/stealing-credentials/credentials-mimikatz (visited on 03/01/2024).

[58] Super User. *Active Directory Hacking: Angriffe mit mimikatz*. Mar. 2024. URL: https://www.whitehat.de/active-directory-hacking-angriffe-mit-mimikatz (visited on 03/01/2024).

[59] MITRE. *Process Injection, Technique T1055 - Enterprise | MITRE ATT&CK®*. Feb. 2024. URL: https://attack.mitre.org/techniques/T1055/ (visited on 03/01/2024).

[60] Red Canary. *Process Injection - Red Canary Threat Detection Report*. Mar. 2024. URL: https://redcanary.com/threat-detection-report/techniques/process-injection/ (visited on 03/01/2024).

[61] MITRE. *Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder, Sub-technique T1547.001 - Enterprise | MITRE ATT&CK®*. Feb. 2024. URL: https://attack.mitre.org/techniques/T1547/001/ (visited on 03/01/2024).

[62] Bima Fajar Ramadhan. *How to Use Metasploit for Post-Exploitation and Maintaining Access*. Feb. 2024. URL: https://linuxhint.com/metasploit-post-exploitation-maintaining-access/ (visited on 03/01/2024).

[63] GitHub. *SwiftOnSecurity/sysmon-config: Sysmon configuration file template with default high-quality event tracing*. Feb. 2024. URL: https://github.com/SwiftOnSecurity/sysmon-config (visited on 02/17/2024).

[64] MITRE. *BITS Jobs, Technique T1197 - Enterprise | MITRE ATT&CK®*. Feb. 2024. URL: https://attack.mitre.org/techniques/T1197/ (visited on 03/01/2024).

[65] GitHub. *GitHub - 3gstudent/bitsadminexec: Use bitsadmin to maintain persistence and bypass Autoruns*. Jan. 2024. URL: https://github.com/3gstudent/bitsadminexec (visited on 02/17/2024).

[66] Penetration Testing Lab. *Persistence – BITS Jobs*. 2019. URL: https://pentestlab.blog/2019/10/30/persistence-bits-jobs/ (visited on 02/17/2024).

[67] Raj Chandel. “Windows for Pentester: BITSAdmin”. In: *Hacking Articles* (Jan. 2020). URL: https://www.hackingarticles.in/windows-for-pentester-bitsadmin/ (visited on 02/17/2024).

[68] *Tracker | LOLBAS*. Feb. 2024. URL: https://lolbas-project.github.io/lolbas/OtherMSBinaries/Tracker (visited on 03/01/2024).

[69] STRONTIC. *Tracker.exe | Tracker*. Feb. 2022. URL: https://strontic.github.io/xcyclopedia/library/Tracker.exe-566303DFC036D3B99B62B158CE5636C5.html (visited on 03/01/2024).

[70] MITRE. *Steal or Forge Kerberos Tickets: Kerberoasting, Sub-technique T1558.003 - Enterprise | MITRE ATT&CK®*. Feb. 2024. URL: https://attack.mitre.org/techniques/T1558/003/ (visited on 03/01/2024).

[71] HackTricks. *Kerberoast*. Mar. 2024. URL: https://book.hacktricks.xyz/windows-hardening/active-directory-methodology/kerberoast (visited on 03/01/2024).

[72] Joe Helle. “Kerberoasting Domain Accounts - TCM Security”. In: (July 2021). URL: https://tcm-sec.com/kerberoasting-domain-accounts/ (visited on 03/01/2024).

[73] Vinaypamnani-msft. *4769(S, F) A Kerberos service ticket was requested. - Windows 10*. Mar. 2024. URL: https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-10/security/threat-protection/auditing/event-4769 (visited on 03/01/2024).

[74] GitHub. *EmpireProject/Empire: Empire is a PowerShell and Python post-exploitation agent*. Mar. 2024. URL: https://github.com/EmpireProject/Empire (visited on 03/01/2024).

[75] Elastic. *PowerShell Invoke-NinjaCopy script | Elastic Security Solution [8.12] | Elastic*. Feb. 2024. URL: https://www.elastic.co/guide/en/security/current/powershell-invoke-ninjacopy-script.html (visited on 03/01/2024).

[76] iainfoulds. *Audit Policy Recommendations*. Aug. 3, 2023. URL: https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/plan/security-best-practices/audit-policy-recommendations (visited on 12/03/2023).

[77] Vinaypamnani-msft. *4662(S, F) An operation was performed on an object. - Windows 10*. Mar. 2024. URL: https://learn.microsoft.com/de-de/previous-versions/windows/it-pro/windows-10/security/threat-protection/auditing/event-4662 (visited on 03/01/2024).

[78] 0xdeaddood. *Forging Tickets in 2023 - 0xdeaddood*. May 2023. URL: https://0xdeaddood.rocks/2023/05/11/forging-tickets-in-2023/ (visited on 03/01/2024).

[79] GitHub. *GhostPack/Rubeus: Trying to tame the three-headed dog*. Mar. 2024. URL: https://github.com/GhostPack/Rubeus (visited on 03/01/2024).

[80] Charlie Bromberg. *Diamond tickets*. Mar. 2024. URL: https://www.thehacker.recipes/a-d/movement/kerberos/forged-tickets/diamond (visited on 03/01/2024).

[81] GitHub. *FSecureLABS/SharpGPOAbuse: SharpGPOAbuse is a .NET application written in C# that can be used to take advantage of a user's edit rights on a Group Policy Object (GPO) in order to compromise the objects that are controlled by that GPO*. Sept. 2024. URL: https://github.com/FSecureLABS/SharpGPOAbuse (visited on 09/12/2024).

[82] GitHub. *Release v1.0.0 · ait-testbed/atb-ansible-primarydc*. Aug. 2024. URL: https://github.com/ait-testbed/atb-ansible-primarydc/releases/tag/Latest (visited on 08/28/2024).

[83] GitHub. *Release v1.0.0 · ait-testbed/atb-ansible-msclient*. Aug. 2024. URL: https://github.com/ait-testbed/atb-ansible-msclient/releases/tag/latest (visited on 08/28/2024).

[84] GitHub. *Release v1.0.0 · ait-testbed/atb-ansible-secondarydc*. Aug. 2024. URL: https://github.com/ait-testbed/atb-ansible-secondarydc/releases/tag/latest (visited on 08/28/2024).

[85] GitHub. *Release v1.0.0 · ait-testbed/atb-ansible-windowseventcollector*. Aug. 2024. URL: https://github.com/ait-testbed/atb-ansible-windowseventcollector/releases/tag/latest (visited on 08/28/2024).

[86] GitHub. *Release v1.0.0 · ait-testbed/atb-ansible-winwebserver*. Aug. 2024. URL: https://github.com/ait-testbed/atb-ansible-winwebserver/releases/tag/latest (visited on 08/28/2024).

[87] GitHub. *Release v1.0.0 · ait-testbed/atb-ansible-winvulnserver*. Aug. 2024. URL: https://github.com/ait-testbed/atb-ansible-winvulnserver/releases/tag/latest (visited on 08/28/2024).

[88] GitHub. *Release v1.0.0 · ait-testbed/atb-ansible-winfileserver*. Aug. 2024. URL: https://github.com/ait-testbed/atb-ansible-winfileserver/releases/tag/latest (visited on 08/28/2024).

[89] GitHub. *Release v1.0.0 · ait-testbed/atb-ansible-ghostagent*. Aug. 2024. URL: https://github.com/ait-testbed/atb-ansible-ghostagent/releases/tag/latest (visited on 08/28/2024).

[90] GitHub. *Release v1.0.0 · ait-testbed/atb-ansible-kafka*. Aug. 2024. URL: https://github.com/ait-testbed/atb-ansible-kafka/releases/tag/v1.0.0 (visited on 08/28/2024).

[91] GitHub. *Seb1n1h0/AECID-WIN-Testbed_Script-Library*. Aug. 2024. URL: https://github.com/Seb1n1h0/AECID-WIN-Testbed_Script-Library (visited on 08/29/2024).

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
AIT	Austrian Institute of Technology
AMiner	Logdata-Anomaly-Miner
BITS	Background Intelligent Transfer Service
CaaC	Cognition as a Code
DC	Domain Controller
DPAPI	Data Protection Application Programming Interface
ELK	Elasticsearch Logstash Kibana
GUID	Globally Unique Identifier
HIDS	Host-based Intrusion Detection System
IaaC	Infrastructure as a Code
ID	Identifier
IDS	Intrusion Detection System
IoC	Indicator of Compromise
IoT	Internet of Things
IP	Internet Protocol
IPSec	Internet Protocol Security
KDC	Key Distribution Center
LSASS	Local Security Authority Subsystem Service
MDE	Microsoft Defender for Endpoint
MPSSVC	Microsoft Protection Service
NPC	Non-Playable Character

NTLM	NT LAN Manager
PAC	Privilege Attribute Certificate
PID	Process Identifier
RC4	Rivest Cipher 4
RPC	Remote Procedure Call
SAM	Security Account Manager
SIEM	Security Information and Event Management
SPN	Service Principal Name
TGT	Ticket Granting Ticket
TIM	Testbed independent model
TSM	Testbed specifc model
VM	Virtual machine
WEC	Windows Event Collector
WEF	Windows Event Forwarding
WinRM	Windows Remote Management

A Appendix A

A.1 Evaluation of attacks

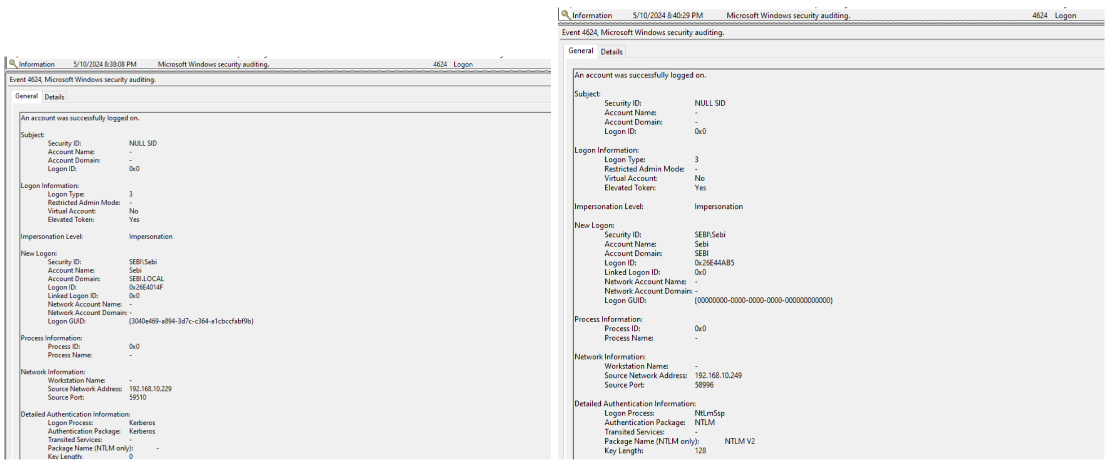
A.1.1 List of attacks

1. Attacker knows user/pass and logs into windows-system using WMI (T1047)
2. Attacker hides payload using process injection (T1055)
3. Attacker uses Mimikatz to dump credentials (T1003)
4. Attacker escalates privileges using
   - win-tasks (T1053.005)
   - logon scripts (1037)
   - modify system process (T1543)
5. Attacker persists malware using
   - BITS Jobs (T1197)
   - Autostart (T1547)
   - DLL-Hijacking (T1574)
   - Windows-Task (T1053.005)
6. Attacker moves to domain controller using
   - pass-the-hash(T1550)
   - rouge domain controller(T1207)
   - Golden Ticket(T1649)
   - kerberroast (T1482)
7. Attacker modifies GPO on domain-controller (T1484)
8. Attacker uses NinjaCopy to read out data from volume (T1006)
9. Attacker modifies registry for hiding payload (T1112)
10. Attacker reads out registry (T1012)

A.1.2 Initial Access - WinRM

Initial access was achieved through WinRM. For access to a Windows host via WinRM, the specified user must belong to either the administrators or remote management users group. To justify such a misconfiguration within an enterprise context, it is conceivable that an individual, acting as the product owner of a particular application, might procure a server. In this scenario, the service on the server is in the responsibility of the requester.

In the analysis, a minimal distinctions between a connection established through WinRM and the Enter-PSSession cmdlet in PowerShell was observed. The primary difference lies in the network information section, specifically in the representation of the source address. Furthermore, disparities were identified in the authentication information, with NTLM employed via Kali and Kerberos utilized for Windows-to-Windows connections through PowerShell. It is crucial to note that these log entries lack the specificity required to formulate IoCs, as the IP address can be manipulated through spoofing and connections via Evil-WinRM may appear legitimate by leveraging authentication with a valid Kerberos ticket. [46]



```
Information    5/10/2024 8:38:08 PM    Microsoft Windows security auditing.    4624  Logon
Event 4624, Microsoft Windows security auditing.

An account was successfully logged on.

Subject:
        Security ID:                NULL SID
        Account Name:               -
        Account Domain:             -
        Logon ID:                   0x0

Logon Information:
        Logon Type:                 3
        Restricted Admin Mode:      -
        Virtual Account:            No
        Elevated Token:             Yes

Impersonation Level:                Impersonation

New Logon:
        Security ID:                SEBI\Sebi
        Account Name:               Sebi
        Account Domain:             SEBI.LOCAL
        Logon ID:                   0x26E4014F
        Linked Logon ID:            0x0
        Network Account Name:       -
        Network Account Domain:     -
        Logon GUID:                 {3040e469-a894-3d7c-c364-a1cbccfabf9b}

Process Information:
        Process ID:                 0x0
        Process Name:               -

Network Information:
        Workstation Name:           -
        Source Network Address:     192.168.10.229
        Source Port:                59510

Detailed Authentication Information:
        Logon Process:              Kerberos
        Authentication Package:     Kerberos
        Transited Services:         -
        Package Name (NTLM only):       -
        Key Length:                 0
```

```
Information    5/10/2024 8:40:29 PM    Microsoft Windows security auditing.    4624  Logon
Event 4624, Microsoft Windows security auditing.

An account was successfully logged on.

Subject:
        Security ID:                NULL SID
        Account Name:               -
        Account Domain:             -
        Logon ID:                   0x0

Logon Information:
        Logon Type:                 3
        Restricted Admin Mode:      -
        Virtual Account:            No
        Elevated Token:             Yes

Impersonation Level:                Impersonation

New Logon:
        Security ID:                SEBI\Sebi
        Account Name:               Sebi
        Account Domain:             SEBI
        Logon ID:                   0x26E44AB5
        Linked Logon ID:            0x0
        Network Account Name:       -
        Network Account Domain:     -
        Logon GUID:                 {00000000-0000-0000-0000-000000000000}

Process Information:
        Process ID:                 0x0
        Process Name:               -

Network Information:
        Workstation Name:           -
        Source Network Address:     192.168.10.249
        Source Port:                58996

Detailed Authentication Information:
        Logon Process:              NtLmSsp
        Authentication Package:     NTLM
        Transited Services:         -
        Package Name (NTLM only):       NTLM V2
        Key Length:                 128
```

Figure 8: WinRM connection from trusted (left) and untrusted source (right)

A.1.3 Read Registry

After gaining foothold on the target system, basic enumeration is needed to gain information about the client and domain. There are many tools which provide automated scanning of the system to identify weaknesses. For this task the application winPEAS [47] was used to identify potential vulnerabilities.

Concurrently with the execution of winPEAS, procmon was deployed with a filter targeting the executable to detect the specific registry keys and files being accessed. Procmon filtered around 12,500 events related to the "userinfo" parameter, 15,000 events for "systeminfo," and approximately 1,135,000 events pertaining to the "domain" parameter. This voluminous data necessitated a more streamlined IoC. As a result, careful monitoring of the Windows event 4688 and 1 "process creation", along with the associated target image and command parameters, is sufficient. [48].

A.1.4 Privilege Escalation

In this section variety of privilege escalations techniques were tested for feasability and suitability.

Windows Tasks

In this section, the vulnerability associated with escalating privileges is based on a misconfiguration on the server. The underlying issue lies in a scheduled task that executes a PowerShell script, located in the ProgramData directory, running with elevated user privileges but open for modification by any domain user. This flaw provides an opportunity for an attacker to modify the script's content.

For the purpose of this attack, an already established connection was leveraged and engage in enumeration activities. PowerShell modules like PowerUp [49] have been tested, which offer task enumeration functionalities, but resulted in no favorable outcomes. Microsoft has addressed this vulnerability through the mitigation of CVE-2019-1069 [50]. As an alternative approach, using legacy binaries that still exist within Windows for compatibility reasons were used, specifically abusing the schtasks binary to execute a living off the land attack which was successfully used to enumerate tasks. Initially, attempting to misconfigure the task or accessing the task scheduler via PowerShell cmdlets or the aforementioned module is blocked by insufficient permissions. However, by executing schtasks, manually enumerating tasks is possible.

During the execution of this procedure, various steps were undertaken. Initially, investigations whether access to the task scheduler was achieved. The only log entry for this action is provided by Sysmon Event ID 1, which captures the process creation event with the image path "C:\Windows\System32\schtasks.exe". Subsequently, the modification of the PowerShell file was performed. However, due to its absence from the system access control list in Windows, this modification is not logged. Otherwise, Event ID 4663 would be generated, containing information regarding the accessed object, but in this particular scenario, it is not applicable. The third step in this attack involved the execution of the modified task. By examining Event IDs 129 (Task launched with ID x), 200 (Action launched in the instance of task) and 201 (Completed task and action), it is logged when the specific task is initiated with the corresponding timestamp. The vulnerable task is triggered during startup, thereby triggering the task during runtime indicates an anomaly. Upon completion of the task, the user is added to the local administrator group. This event can be identified by Event ID 4732, which signifies the addition of a member to a security-enabled local group. This serves as a more precise IoC. [51] [52]

Logon Scripts

Technique T1037.001, as specified by MITRE, involves the utilization of logon scripts in Windows. In this context, the registry key HKCU:SOFTWARE\UserInitMprLogonScript is particularly relevant. During the user logon process on Windows systems, the winlogon process is responsible for invoking userinit.exe, which is tasked with loading the user's environment. Consequently, by configuring the registry key with the path to an executable file, it can be automatically executed during logon, as explained in [53], where the calculator application is launched following logon. MITRE classifies this technique as both privilege escalation and persistence [54].

To carry out this attack, the Set-ItemProperty Powershell cmdlet can be utilized. Multiple batch files and Powershell scripts were employed in attempts to achieve privilege escalation. However, due to the token elevation type for logon scripts being set to 3, the corresponding process is assigned a limited token without elevated privileges. Nonetheless, persistence is achieved by leveraging the registry key, which is directed towards an executable that establishes a connection with the attacker's machine, thereby executing the establishment of a Meterpreter reverse shell.

Analyzing this attack reveals a chain of system calls that initiates with winlogon.exe invoking userinit.exe and further in the execution of the payload referenced in the UserInitMprLogonScript. By monitoring the registry key itself and detecting modifications made to its hive, it is possible to identify potential IoCs. Furthermore, Event ID 4688 and Event ID 1 may serve as an anomaly indicator, as instances where the creator process name "userinit.exe" invokes another process that deviates from the expected behavior could signify a suspicious combination of events.

System Process Modification

This aspect will be addressed within the context of the Mimikatz attack outlined in Section A.1.5. The presence of the Mimikatz driver on the system signifies a modification to the system processes as it will be used to remove protection from the LSASS process.

A.1.5 Mimikatz

Mimikatz is a widely recognized post-exploitation tool with focus on dumping credentials. Developed by Benjamin Delpy [55], it has gained significant attention within the security community due to its efficacy in extracting authentication information from compromised systems, particularly credentials stored in Windows operating systems.

To execute this attack scenario, Mimikatz binaries were obtained from the repository hosted on GitHub [55]. Establishing a remote shell to the target client was achieved using PsExec.py from the Impacket collection due to compatibility issues with evil-winrm, where executing Mimikatz resulting in shell crashes [56]. Prior to initiating the attack, privilege escalation was conducted, enabling the installation of mimidriver to access LSASS and remove process protection. Subsequently, stored credentials on the system were extracted. This process required the setting of prerequisite registry keys to enable the storage of passwords in cleartext [57],[58].

Each stage of the attack was analyzed individually. Initially, inspecting traces associated with connection attempts using PsExec.py. Beside the logon event with ID 5140, logs for accessing a network share object (Event ID 5145) and installing a service on the system (Event ID 7045) was found. PsExec.py was observed to enumerate a share where it drops a malicious binary, subsequently executing the payload and activating the remote shell through service installation. Upon session termination, all artifacts were purged from the target system.

For Mimikatz, the initial IoC presents as Event ID 4688 or Sysmon Event ID 1, indicating the creation of a new process where the process name or image concludes with "mimikatz.exe". Similarly, Event ID 4689 signifies the exit of a process. However, this IoC is vulnerable to evasion through simple file renaming. Upon initiating the attack and launching Mimikatz, the driver mimidrv.sys was loaded to gain kernel space access. Detection may occur via Windows Event ID 7045 or Sysmon Event ID 6, both indicating driver loading. While both logs include the loaded image, Sysmon events additionally provide file hashes and signatures. Subsequent steps involve removing process protections from LSASS, signifying a system process modification, captured in logs as Event ID 4663. This event requires kernel object auditing, providing information about the source process and accessed object, including access request details.

A.1.6 Process Injection

Process injection is a tactic employed by adversaries to implant code into a process, to evade defensive measures and potentially escalate privileges. This involves injecting arbitrary code into the memory space of a currently active process. Such infiltration creates pathways for accessing the memory, system and network resources associated with the targeted process. [59]

In the context of the process injection, a payload was generated using msfvenom to facilitate the deployment of a Meterpreter shell. The payload was transferred to the target system through Evil-WinRM, where it was subsequently activated on the target after the establishment of a listener on the attacker machine. Leveraging the newly established Meterpreter session, the process currently in execution was injected into explorer.exe, accomplished through the command "migrate <PID>".

Detecting process injections necessitates the utilization of a properly configured Sysmon, as native Windows Event Logs do not provide detailed information necessary to identify process injection activities [60]. Sysmon Event ID 8 is capable of detecting remote thread creation and provides details about both the source and destination images involved. This procedure occurs organically on Windows hosts and therefore, instances where the source image and target image combination is unfamiliar may indicate a compromise.

A.1.7 Hide Payload

This scenario will be exemplified in subSection A.1.8, where Metasploit was employed alongside the persistence module, facilitating the complete installation of the payload within the registry.

A.1.8 Persistence

In this section different persistence techniques will be demonstrated.

Autorun

The registry contains run keys that initiate the launch of applications upon system startup. Enterprises commonly utilize these keys to enable automatic startup of messaging applications such as Microsoft Teams or OneDrive, for instance. The specified application is executed during user logon within the user's context any may be abused by threat actors. [61]

Having employed Metasploit in the prior attack (Section A.1.6), the utilization of this offensive tool was continued. Metasploit offers a persistence module that installs a payload within the registry, executing it through the autorun key upon system startup. This module was utilized for persistence and also covers the section labeled as "HidePayload" in Section A.1.7, whereby the binary is entirely written into the registry. [62]

During the attack conducted using Metasploit, two registry keys were created. The first registry key, detected through Sysmon Event ID 13 (registry value set), is a run key. The second key is written in the HKCU:Software\DM21aGxU hive, where the payload is obfuscated as a string. However, the same event was not visible for the second key, as the Sysmon configuration lacked a rule for the HKCU:Software hive[63] which would lead overloading the logs.

Another point of detection occurs when the target machine undergoes a reboot and the user logs on. At this point, the run key automatically executes the payload, resulting in the generation of Event ID 4688 showing the path to the obfuscated binary. Since the run key involves the use of Powershell for invocation, additional traces are left in the Powershell logs. These include Event ID 400 (Engine state is changed from None to available), Event ID 4104 (executing a remote command) and Event ID 800 (pipeline execution details). All of these entries provide information about the executed command.

BITS

This section explores the potential exploitation of the Windows Background Intelligent Transfer Service, further abbreviated as BITS, by malicious actors. BITS, a low-bandwidth, asynchronous file transfer mechanism, finds legitimate use in background operations like updates and messaging for its efficient resource utilization. However, its inherent characteristics can be weaponized to achieve persistent code execution and perform various clandestine tasks. [64]

By leveraging administrative privileges on the target system, it becomes possible to execute BITS cmdlets in Powershell. For non-priviliged accounts this is not feasible but the backward compatibility of the Windows operating system may be abused, specifically through the use of bitsadmin enabeling a living-off-the-land technique. Consequently, the choice was made to employ bitsadmin and establish a BITS job with a duration of 90 days, ensuring that the job regularly triggers a scheduled payload download with execution, thereby functioning as a loader. [65][66]

BITS has its own designated channel in the event log, simplifying the analysis process [33]. Starting with Event ID 3 (Network connection detected), the BITS service is observed creating a new job and assigning it a specific job ID [67]. By correlating this job ID with Event 16403, additional information about the job can be obtained. This allows the identification of any abnormalities, particularly in cases where the external network connection is unrecognized and the executed payload is classified as malicious.

Tracker.exe

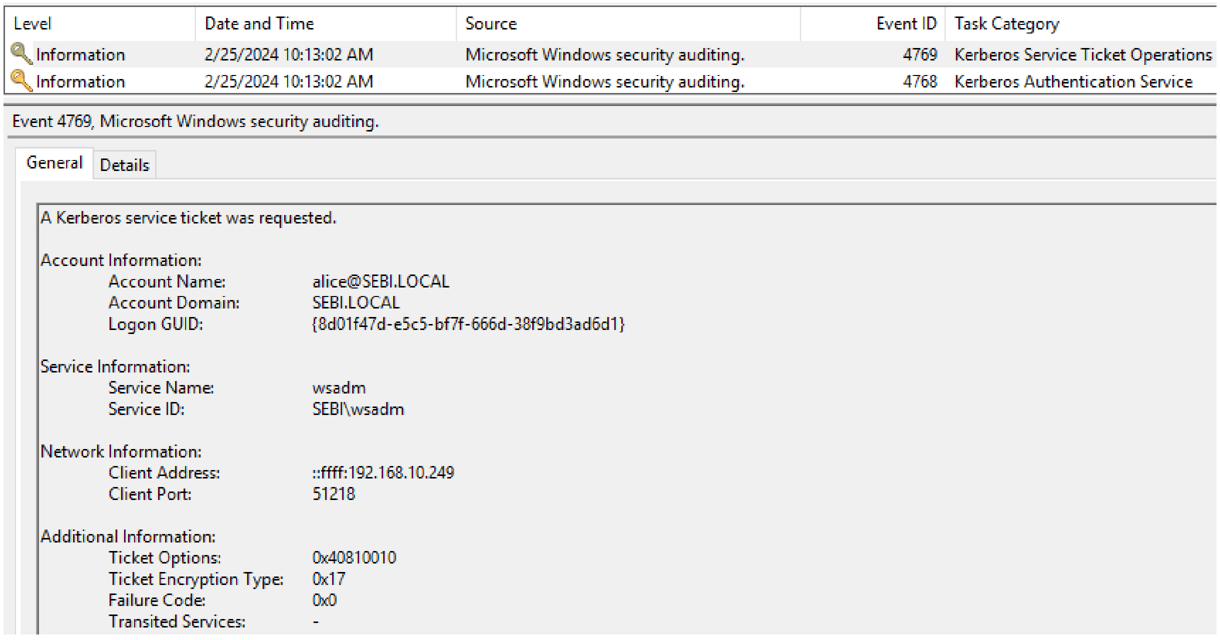
This attack appears to be infeasible. Despite some IOCs in [68],[69] and limited instructional resources, attempts to execute it have proven unsuccessful. The procedure contained the creation of a malicious DLL using msfvenom and attaching the DLL to a running process.

A.1.9 Kerberroasting

Kerberoasting is an attack technique commonly employed by threat actors to compromise Active Directory environments. This method targets service principal names, which are unique identifiers for services running in an enterprise. By exploiting vulnerabilities in the Kerberos authentication protocol, attackers can extract encrypted credentials associated with SPNs and then attempt to crack these hashes offline, potentially gaining unauthorized access. [70]

To prepare for the Kerberoast attack, the initial procedure contains verifying the presence of a configured user SPN within the designated environment. This validation process was performed through the utilization of the GetUserSPNs.py script sourced from the Impacket collection [56]. The functionality of this script streamlines the enumeration of user SPN-related data within the specified domain, additionally enabling the extraction of associated hashes from service accounts when run with the "-request" parameter. With success running GetUserSPNs.py, access to an account with domain admin privileges was gained. [71][72]

In order to detect kerberroasting, the encryption type of the ticket becomes crucial. The script employed from the Impacket collection utilizes RC4 encryption, which offers the advantage of faster offline hash cracking. In Event ID 4768, a request is made for a Kerberos authentication ticket, with the ticket encryption type being displayed as 0x17, representing the use of RC4 encryption. This observation also applies to Event ID 4769, where a Kerberos service ticket is requested [73]. The service name associated with the Kerberos ticket is displayed in the event, as illustrated in Figure 9.



Level	Date and Time	Source	Event ID	Task Category
Information	2/25/2024 10:13:02 AM	Microsoft Windows security auditing.	4769	Kerberos Service Ticket Operations
Information	2/25/2024 10:13:02 AM	Microsoft Windows security auditing.	4768	Kerberos Authentication Service

Event 4769, Microsoft Windows security auditing.

```
A Kerberos service ticket was requested.

Account Information:
        Account Name:             alice@SEBI.LOCAL
        Account Domain:           SEBI.LOCAL
        Logon GUID:               {8d01f47d-e5c5-bf7f-666d-38f9bd3ad6d1}

Service Information:
        Service Name:             wsadm
        Service ID:               SEBI\wsadm

Network Information:
        Client Address:           ::ffff:192.168.10.249
        Client Port:              51218

Additional Information:
        Ticket Options:           0x40810010
        Ticket Encryption Type:   0x17
        Failure Code:             0x0
        Transited Services:       -
```

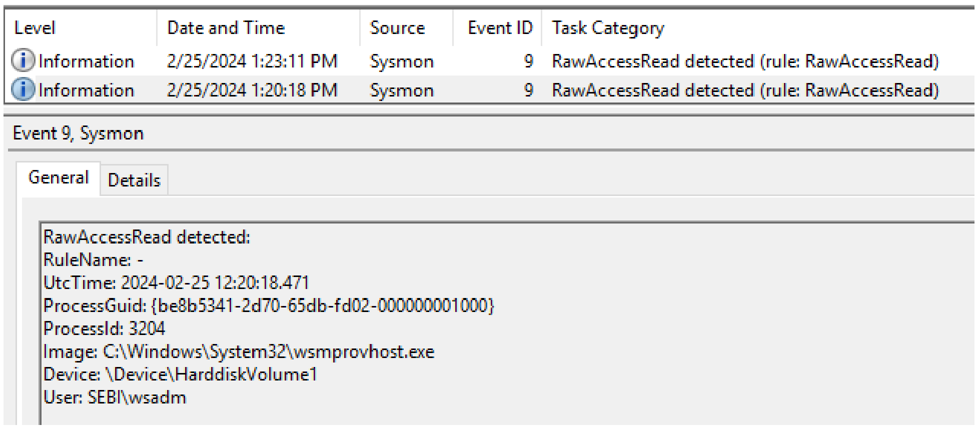
Figure 9: Event ID 4769 - A Kerberos serivce Ticket was requested

A.1.10 NinjaCopy

Ninjacopy, a script found within the PowerSploit collection, is utilized for data exfiltration by circumventing file locks, allowing it to copy files directly from the NTFS volume. [49]

The initial step involved in the attack was gaining access to the domain controller by abusing the SPN, as achived in Section A.1.9. Subsequently, a connection was established using Evil-WinRM and proceeded to load the Invoke-NinjaCopy script into the shell. While the original script sourced from PowerSploit proved not to be functional at is was outdated, a more advanced version available in Empire [74] proved to be functional. The objective of the attack was to parse the NTDS.dit file, which serves as the Active Directory database. In addition, the SYSTEM and SECURITY hive were exported from the registry using NinjaCopy, as they were necessary for accessing the database file. Once all three files were successfully downloaded from the domain controller, the NTDS.dit file was locally read using secretsdump. [56] This action resulted in the display of all Active Directory objects including hashes and private keys.

The detection of NinjaCopy can only be accomplished by correlating various Event IDs, as the script itself does not leave any binaries on the targeted system through the use of Evil-WinRM functionalities. Sysmon Event ID 9 (Raw Access Read), plays a crucial role in this process as it detects if a volume has been accessed and provides information about the source image accessing it. In this case, the presence of the "wsmprovhost" PowerShell hosting process indicates that the access was achieved through a remote PowerShell session. By correlating this with Event ID 4624, which signifies a successful account login and Event ID 91, which reveals the establishment of a WSMan shell on the server, it becomes evident that the shell was established from the attacker's machine, as indicated by the displayed source IP address. [75]



Level	Date and Time	Source	Event ID	Task Category
Information	2/25/2024 1:23:11 PM	Sysmon	9	RawAccessRead detected (rule: RawAccessRead)
Information	2/25/2024 1:20:18 PM	Sysmon	9	RawAccessRead detected (rule: RawAccessRead)

Event 9, Sysmon

```
RawAccessRead detected:
RuleName: -
UtcTime: 2024-02-25 12:20:18.471
ProcessGuid: {be8b5341-2d70-65db-fd02-000000001000}
ProcessId: 3204
Image: C:\Windows\System32\wsmprovhost.exe
Device: \Device\HarddiskVolume1
User: SEBI\wsadm
```

Figure 10: Sysmon Event ID 9 - Raw Disk Access

A.1.11 DCSync

To archive the same goal as in the previous attack, an alternate approach is using the DCSync attack to imitate a domain controller and sync the Active Directory data. This attack was also performed using secretsdump.py from the Impacket collection [56]. By executing secretsdump with the "-just-dc" parameter, the script performs a dcsync and retrieves the same informations as in subsection A.1.10.

In order to have visibility in the audit logs, it is necessary to enable directory services access and object access. The first anomaly is manifested in event ID 4624 (Group membership evaluation has started) [76], which indicates a successful account login achieved through NTLM authentication. Following the logon event, event ID 4627, which provides group membership information, reveals the permissions assigned for the session. Notably, this includes attributes such as Pre-Windows 2000 Compatible Access, Domain Admins, Denied RODC Password Replication and a high mandatory label. The combination of these attributes, which has not been observed in the environment before, serves as an additional indicator of potential malicious activity. Subsequently, events with ID 4662 (An operation was performed on an object) [77] are recorded, indicating an operation performed on an object. Of particular interest is the presence of an access mask value of 0x100, signifying the use of extended rights to carry out the operation.

Diamond Ticket

The diamond ticket represents a advancement from the well-known golden ticket attack, which is recognized for its ability to compromise a domain through the exploitation of the Kerberos authentication protocol. In an effort to address this vulnerability, Microsoft introduced a patch in CVE-2021-42287, which aimed to mitigate Kerberos KDC confusion. The patch effectively makes it more challenging to forge golden tickets within the environment. However, this mitigation also inadvertently introduced new attack vectors, known as diamond and sapphire tickets. [9, 78]

In order to generate a diamond ticket, the prescribed methodology outlined by Charlie Clark in [11] was performed through the Rubeus application [79]. The exploitation process begins by initiating a service ticket request, which is subsequently decrypted to extract the Privileged Access Control field. The PAC is then modified and digitally signed before being encrypted once again. Following the creation of the service ticket, a ticket granting service ticket is requested, incorporating the desired services as specified in Rubeus. This sequence of actions grants authorized access to the domain controller via the CIFS service. [80]

The detection of a diamond ticket at the host level is not entirely sufficient. This is because an original ticket is requested initially, resulting in Windows Event Logs that do not exhibit significant differences in the log entries themselves. A potential anomaly was nevertheless found in the analysis, as the attack generated four logs, whereas a legitimate access through another server produced only three logs related to Kerberos. Additionally, on the workstations, there was a discrepancy of one ticket in the Kerberos cache, which also constituted an anomaly, as the "Kdc Called:" field was found to be empty. This was analysed after the cached Kerberos tickets were purged and before the attack was initiated. Another deviation was observed in the time frame between the generation of Event ID 4769 and Event ID 4768. Other indicators may include the creation of processes or the occurrence of PowerShell events, where the string "Rubeus" is present in the source image.

A.1.12 GPO Modification

The final phase within the attack chain encompasses the manipulation of Group Policy Objects. As for an enterprise to establish and enforce user and machine configurations throughout the entire domain, GPOs are acting as a centralized repository for policies and settings. Adversaries can abuse these settings in various ways, including the elevation of privileges at the domain level, the establishment of persistence within the environment, or the deployment of run keys and scheduled tasks, as demonstrated in previous attack instances.

In this attack scenario, the tool SharpGPOAbuse [81] is utilized to manipulate an existing GPO. This tool enables the inclusion of user rights, computer scripts, user scripts and tasks, while also facilitating modifications to local group membership, such as the administrator group. Each type of attack provides additional execution parameters which must be appropriately configured. In order for the environment to function correctly, a GPO named "Add SeBatchLogonRight" was configured, because of Ansible and subsequently modified using SharpGPOAbuse to add specified users to the local administrator group. In our specific scenario, the user "WSADM" was added using the following command:

"C:\Temp\SharpGPOAbuse_4.7_4_x86.exe" –AddLocalAdmin –UserAccount bob –GPOName SeBatchLogonRight –Domain sebi.local –DomainController winserver19.sebi.local –TargetUsername bob

The manifestation of the attack is once again evident through the occurrence of specific Windows Event IDs. Event ID 1 and Event ID 4688 indicate that the source image contains the application SharpGPOAbuse. The functioning of this application involves establishing a connection with the domain controller and accessing the GPO files. This step is further detected through Event ID 5145 (access to a network share object), which provides information about the share path and the relative target name, which contains the GUID of the targeted policy. Lastly, the modification of the GPO is observed in Event ID 5136 (modification of a directory service object), which provides details about the GPO object itself.

B Appendix B

Table 2: Comparison of audit rules for server (1/2)

Setting	Microsoft	Palantir
Account Logon		
Audit Credential Validation	S | F	S | F
Audit Kerberos Authentication Service	S | F	S | F
Audit Kerberos Service Ticket Operations	S | F	S | F
Audit Other Account Logon Events	S | F	S | F
Account Management		
Audit Application Group Management		
Audit Computer Account Management	S | F	S | F
Audit Distribution Group Management		S | F
Audit Other Account Management Events	S | F	S | F
Audit Security Group Management	S | F	S | F
Audit User Account Management	S | F	S | F
Detailed Tracking		
Audit DPAPI Activity	S | F	S | F
Audit Process Creation	S | F	S | F
Audit Process Termination		S | F
Audit RPC Events		
DS Access		
Audit Detailed Directory Service Replication		S | F
Audit Directory Service Access	S | F	S | F
Audit Directory Service Changes	S | F	S | F
Audit Directory Service Replication		S | F
Logon and Logoff		
Audit Account Lockout	S	S | F
Audit User/Device Claims		S | F
Audit IPsec Extended Mode		
Audit IPsec Main Mode		
Audit IPsec Quick Mode		
Audit Logoff	S	S | F
Audit Logon	S | F	S | F
Audit Network Policy Server		
Audit Other Logon/Logoff Events	S | F	S | F
Audit Special Logon	S | F	S | F

Table 3: Comparison of audit rules for server (2/2)

Setting	Microsoft	Palantir
Object Access		
Audit Application Generated		
Audit Certification Services		
Audit Detailed File Share		F
Audit File Share		S | F
Audit File System		S | F
Audit Filtering Platform Connection		F
Audit Filtering Platform Packet Drop		
Audit Handle Manipulation		
Audit Kernel Object		
Audit Other Object Access Events		S | F
Audit Registry		S | F
Audit Removable Storage		S | F
Audit SAM		
Audit Central Access Policy Staging		
Policy Change		
Audit Audit Policy Change	S | F	S | F
Audit Authentication Policy Change	S | F	S | F
Audit Authorization Policy Change		
Audit Filtering Platform Policy Change		
Audit MPSSVC Rule-Level Policy Change	S	S | F
Audit Other Policy Change Events		S | F
Privilege Use		
Audit Non Sensitive Privilege Use		F
Audit Other Privilege Use Events		
Audit Sensitive Privilege Use		S | F
System		
Audit IPsec Driver	S | F	
Audit Other System Events		S | F
Audit Security State Change	S | F	S | F
Audit Security System Extension	S | F	S | F
Audit System Integrity	S | F	S | F

C Appendix C

C.1 Ansible roles

- atb-ansible-primarydc [82]
- atb-ansible-msclient [83]
- atb-ansible-secondarydc [84]
- atb-ansible-windowseventcollector [85]
- atb-ansible-winwebserver [86]
- atb-ansible-winvulnserver [87]
- atb-ansible-winfileserver [88]
- atb-ansible-ghostserver [42]
- atb-ansible-ghostagent [89]
- atb-ansible-kafka [90]

C.2 Scriptcollection

- Terraform [91]
- Attackmate [91]
- Python [91]
- R [91]