

AMERICAN UNIVERSITY OF BEIRUT

LEVERAGING AI FOR CONFIDENT CLASSIFICATION AND  
PRIORITIZATION OF INTRUSION DETECTION SYSTEM  
ALERTS

by  
ALI MOHAMMAD MUSTAFA

A thesis  
submitted in partial fulfillment of the requirements  
for the degree of Master of Engineering  
to the Department of Electrical and Computer Engineering  
of the Maroun Semaan Faculty of Engineering and Architecture  
at the American University of Beirut

Beirut, Lebanon  
August 2024

AMERICAN UNIVERSITY OF BEIRUT

LEVERAGING AI FOR CONFIDENT CLASSIFICATION AND  
PRIORITIZATION OF INTRUSION DETECTION SYSTEM  
ALERTS

by  
ALI MOHAMMAD MUSTAFA

Approved by:

---

Dr. Ali Chehab, Professor and Chairperson Department of Electrical and Computer Engineering	Advisor
--	---------

---

Dr. Youssef Tawk, Associate Professor Department of Electrical and Computer Engineering	Member of Committee
--	---------------------

---

Dr. Hadi Sareddeen, Assistant Professor Department of Electrical and Computer Engineering	Member of Committee
--	---------------------

Date of thesis defense: August 14, 2024

## ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Dr. Ali Chehab, my advisor, for his invaluable guidance, unwavering support, and insightful feedback throughout the process of completing this thesis. His expertise, patience, and encouragement have been instrumental in shaping my research and academic journey.

I am also deeply thankful to Dr. Youssef Tawk and Dr. Hadi Sareddeen for serving as members of my thesis committee. Their constructive criticism, expertise in their respective fields, and commitment to academic excellence have greatly enriched this work. I am also grateful to the staff of the ECE department for their massive support throughout my journey, and my MSFEA colleagues for their unwavering encouragement.

# ABSTRACT OF THE THESIS OF

Ali Mohammad Mustafa

for

Master of Engineering

Major: Networks and Security

Title: Leveraging AI for Confident Classification and Prioritization of Intrusion Detection System Alerts

The increasing complexity and volume of cybersecurity alerts significantly challenge threat detection efforts, particularly within Security Operations Centers (SOCs), where the high rate of false positives can obscure real and dangerous threats. This burden not only strains resources but also increases the risk of overlooking genuine security breaches. Leveraging advanced machine learning techniques, particularly Large Language Models (LLMs), this thesis introduces a novel methodology aimed at enhancing the precision of alert classifications from Windows endpoints' security logs.

This study extracted approximately 700 false and real threat cases from a real enterprise network. The proposed approach involves creating an Execution Graph for each alerting Windows process, which is then processed by a "Graph Contextualizer" block. This block transforms complex process interactions into structured, analyzable formats suitable for training and inference in large language models.

The transformed data points are subsequently fed into several locally fine-tuned LLMs designed to classify the alerts accurately. Preliminary evaluation of this pipeline shows excellent metrics, achieving high levels of precision and recall, thereby substantiating the effectiveness of our approach. The methodology not only improves the operational efficiency of SOCs by reducing the investigative overhead of false threats and assisting in the detection of real threats but also contributes significantly to the broader field of cybersecurity, offering a scalable model for integrating machine learning into existing security infrastructures.

Keywords - Security Operations Center (SOC), Host Intrusion Detection Systems (HIDS), False Positives, Large Language Models (LLM), Alert Classification, Audit Logs

# TABLE OF CONTENTS

ACKNOWLEDGEMENTS .....	1
ABSTRACT .....	2
ILLUSTRATIONS .....	6
ABBREVIATIONS .....	7
INTRODUCTION .....	8
1.1 Motivation/Background .....	8
1.2 Contribution .....	9
LITERATURE REVIEW .....	10
2.1 Intrusion Detection Systems (IDS) advancements .....	10
2.2 HIDS (Host-based Intrusion Detection Systems) .....	13
2.3 AI in HIDS .....	15
2.4 Provenance Graphs .....	17
2.5 LLMs in HIDS .....	18
OVERVIEW OF LOG AUDITING .....	21
3.1 Windows Event Log .....	21
3.2 Endpoint Monitoring Architecture .....	24
3.2.1 Basics of Endpoint Monitoring .....	25
3.2.2 Beats: Data Collection Mechanisms .....	25
3.2.3 Elasticsearch and ElastAlert: Storage and Alert Management .....	26

3.2.4 Elastic Rules: Creation and Application Examples .....	26
3.2.5 TheHive: Incident Management Workflow .....	27
3.2.6 Investigation Process: From Alert Detection to Resolution .....	27
3.2.7 Incident Example .....	28
3.3 Large Language Models (LLMs):.....	29
3.3.1 Text Classification in Cybersecurity.....	30
3.3.2 Anomaly Detection Techniques.....	30
3.3.3 Enhancements in Security Applications .....	31
3.3.4 Data and Computational Demands .....	31
<b>PROPOSED METHODOLOGY .....</b>	<b>33</b>
4.1 Testbed and Baseline Data.....	33
4.2 Attack Simulation .....	34
4.3 Graph-Based Process Analysis .....	35
4.4 Graph Contextualizer .....	38
4.5 LLM-Based Classification.....	40
4.5.1 Creating a Dataset.....	40
4.5.2 Creating a Tokenizer.....	40
4.5.3 Building the LLM with a Classification Head.....	41
4.5.4 Finetuning the Model.....	41
4.5.5 Inference and Performance Metrics .....	42
<b>EXPERIMENTATION AND RESULTS .....</b>	<b>43</b>
5.1 Experimentation Dataset.....	43
5.2 Experimentation Metrics.....	43
5.3 Experimentation Results and Analysis .....	46
5.4 Future Work.....	48

5.5 Conclusion .....	49
<b>REFERENCES .....</b>	<b>51</b>

## ILLUSTRATIONS

### Figure

1. Sysmon Log Example .....	24
2. Overview of the Alerting Flow .....	29
3. Generated Graph of an example alert .....	38
4. General Overview of the Proposed Pipeline.....	42
5. Experiment Confusion Matrix defining metrics of interest .....	44
6. Chosen Models in addition to their Confusion Matrixes.....	46

## ABBREVIATIONS

AI	Artificial Intelligence
IDS	Intrusion Detection Systems
HIDS	Host-based Intrusion Detection Systems
NIDS	Network-Based Intrusion Detection Systems
GNNs	Graph Neural Networks
CNNs	Convolutional Neural Networks
RNNs	Recurrent Neural Networks
LLMs	Large Language Models
GPT	Generative Pre-trained Transformer
SOCs	Security Operations Centers
EDR	Endpoint Detection and Response
LSTM	Long Short-Term Memory
RDP	Remote Desktop Protocol
GRU	Gated Recurrent Unit
APTs	Advanced Persistent Threats
BERT	Bidirectional Encoder Representations from Transformers
IT	Information technology
ID	Identity Document
JSON	JavaScript Object Notation
IOC	Indicators of Compromise
URL	Uniform Resource Locator
PID	Process Identifier
MD5	Message Digest Method 5
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative

# CHAPTER 1

## INTRODUCTION

### **1.1 Motivation/Background**

In the digital era, cybersecurity is not merely an option but a fundamental necessity for protecting data, privacy, and infrastructures from a growing array of cyber threats. Among the myriad technologies deployed to safeguard our digital assets, Intrusion Detection Systems (IDS) stand out as critical barriers against malicious activities [1]. These systems scrutinize network traffic and system activities to identify patterns that may indicate a security breach. However, the effectiveness of IDS is often compromised by a significant challenge: the prevalence of false positives—incorrect identifications of benign activities as threats. This issue not only burdens security personnel with unnecessary alerts but also diverts attention away from real and potentially damaging threats, thereby straining resources and reducing the operational efficacy of Security Operations Centers (SOCs).

The issue of false positives is particularly significant as it can lead to alert fatigue, a critical problem where the sheer volume of alerts causes security personnel to overlook or miss actual cyber threats. In sectors where cybersecurity is crucial, such as in protecting critical infrastructure or sensitive data, enhancing the precision of threat detection systems is paramount. This thesis addresses this essential challenge by incorporating advanced machine learning techniques, specifically Large Language Models (LLMs), to refine the alert classification process within IDS, aiming to reduce the frequency of false positives significantly.

This research introduces an innovative approach that integrates state-of-the-art ML technologies with traditional security data analysis methodologies. The objectives of this study are dual: first, to devise and implement a methodology capable of substantially reducing false positive rates in IDS alerts while not allowing real threats to pass; second, to assess the efficacy of this methodology within an operational SOC environment. Achieving these objectives will contribute to the cybersecurity field by enhancing IDS operational performance and demonstrating a scalable model for applying ML to improve security protocols.

## **1.2 Contribution**

The contributions of this thesis can be summarized as follows:

1. Development of a SOC Workload Reduction Pipeline that significantly decreases false positive alerts while ensuring accurate detection of true positives, reducing analyst workload.
2. Innovative automation and integration of execution graphs into security analysis, enhancing the contextual understanding of system interactions.
3. Creation of a Graph Contextualizer that transforms complex execution graphs into data suitable for Large Language Model processing, facilitating advanced AI-driven security analysis.
4. Conducting extensive attack simulations across an enterprise network to validate the model's robustness and ensure its effectiveness against a wide range of real-world cybersecurity challenges.

## CHAPTER 2

### LITERATURE REVIEW

In the realm of cybersecurity, anomaly detection is pivotal for maintaining the integrity and safety of data communication systems. Leveraging system logs, particularly EDR (Endpoint Detection and Response) logs, allows for a granular understanding of execution processes and the mapping of execution graph networks.

#### **2.1 Intrusion Detection Systems (IDS) advancements**

The field of Intrusion Detection Systems (IDS) has undergone significant evolution since its inception. Initially, IDS were primarily based on signature detection, where predefined rules and known patterns of attacks were used to identify threats. These signature-based systems, while effective against known threats, struggled with detecting new or unknown threats due to their reliance on pre-existing knowledge. As cyber threats grew in sophistication and frequency, the limitations of traditional IDS became increasingly apparent [2].

The advent of Artificial Intelligence (AI) marked a pivotal shift in the capabilities of IDS. AI, with its ability to learn from data, adapt to new situations, and improve over time, brought transformative changes to threat detection mechanisms. Machine learning, a subset of AI, enabled IDS to analyze vast amounts of data, identify patterns, and detect anomalies without explicit programming for each possible threat scenario.

AI-based IDS leverages various ML algorithms, including supervised, unsupervised, and reinforcement learning. Supervised learning models are trained on

labeled datasets containing both normal and malicious activities [3]. These models learn to classify new data based on patterns observed during training. Unsupervised learning, on the other hand, does not require labeled data. Instead, it identifies anomalies by learning the normal behavior of a system and flagging deviations from this norm. Reinforcement learning, though less common in IDS, can adapt to dynamic environments by learning optimal strategies through interactions with the environment.

The integration of AI into IDS has significantly enhanced their effectiveness and efficiency. One of the key advantages of AI-based IDS is their ability to perform real-time analysis of network traffic and system activities. Traditional IDS often faced challenges in processing and analyzing the massive volumes of data generated in modern networks. AI algorithms, particularly those based on deep learning, are capable of handling these large datasets efficiently.

Deep learning models, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have been particularly effective in enhancing IDS capabilities. CNNs are adept at recognizing spatial patterns in data, making them suitable for analyzing network traffic patterns. RNNs, with their ability to handle sequential data, are well-suited for analyzing sequences of system calls or log entries. These models can detect complex and subtle anomalies that might be missed by traditional methods [4].

AI also plays a critical role in reducing the false positive rates of IDS. High false positive rates can overwhelm security analysts and reduce the overall effectiveness of an IDS. ML models, through techniques such as anomaly detection and clustering, can distinguish between benign anomalies and actual threats, thereby reducing false positives.

Recent advancements in AI, such as Graph Neural Networks (GNNs) and Large Language Models (LLMs), are further pushing the boundaries of what IDS can achieve. GNNs, which operate on graph-structured data, are particularly effective for modeling relationships and interactions in network data. This makes them ideal for detecting sophisticated attack patterns that involve multiple nodes and connections within a network. GNNs can learn intricate dependencies and leverage contextual information to improve the accuracy and robustness of threat detection [5].

LLMs, on the other hand, have shown great potential in log analysis and anomaly detection. These models, trained on extensive corpora of text data, can understand and generate human-like text. In the context of IDS, LLMs can be used to analyze log entries, detect anomalies, and even generate explanatory reports that help security analysts understand the nature of detected threats. The ability of LLMs to process natural language data and extract meaningful patterns adds a new dimension to IDS capabilities, enabling more comprehensive and intuitive threat detection [6].

In summary, the integration of AI into IDS has revolutionized threat detection mechanisms. AI models, with their ability to learn from data, adapt to new threats, and analyze complex patterns, have significantly enhanced the capabilities of IDS across various platforms. From improving detection accuracy and reducing false positives to handling large and diverse datasets in real-time, AI has transformed IDS into more robust, adaptive, and effective security solutions. This evolution continues as AI technologies advance, promising even greater enhancements in the future.

## **2.2 HIDS (Host-based Intrusion Detection Systems)**

In the domain of Host-based Intrusion Detection Systems (HIDS), the core functionality revolves around meticulously monitoring and analyzing data derived from system operations, which include system calls, application logs, and file-system modifications. These data sources are critical as they serve as conduits through which programs request specific services from the operating system, or record events that might indicate security breaches. For instance, an excessive frequency of specific system calls could signal an attempt at privilege escalation, while unusual patterns in application logs, such as repeated failed login attempts, might suggest unauthorized access attempts. Similarly, unauthorized changes in file permissions or the creation of new files can often signal a compromise. The integration of these systems into the host's infrastructure allows them to have a comprehensive overview of the host's state and activities, enhancing their ability to detect anomalies.

The technology behind HIDS employs sophisticated detection techniques broadly categorized into two types: signature-based and anomaly-based detection. Signature-based detection relies on predefined patterns of known threats, enabling HIDS to effectively identify and mitigate recognized malware and attacks. This method is known for its precision with known threats but is limited by its inability to detect new, unknown attacks that do not match any existing signatures. On the other hand, anomaly-based detection does not depend on prior knowledge of threats. Instead, it models what is considered normal behavior within the system and flags deviations from this baseline as potential threats. This allows for the detection of novel or zero-day exploits that signature-based methods might miss, though it often suffers from higher rates of false positives, which can lead to alert fatigue among security personnel [7].

In addition to these advanced AI-driven methods, a substantial reliance on classical techniques remains integral to the functionality of HIDS. These traditional methods, such as n-grams, rule-based detection, and stateful protocol analysis [8], provide a foundational layer of security by utilizing simpler, yet effective, analytical approaches to identify potential threats based on historical data and predefined rules. N-grams, borrowed from text analysis, are used to detect anomalies in system call sequences by identifying unusual patterns that deviate from common or benign sequences, effectively pinpointing anomalies within large datasets. Rule-based detection, similar to signature-based but broader, incorporates rules that apply to system configurations, file changes, and network connections that should not occur under normal operating conditions. Stateful protocol analysis, meanwhile, involves inspecting and maintaining a record of the state of network protocols during a session to detect deviations that suggest misuse or an attack.

These classical methods are vital in a HIDS environment, providing the first line of defense based on well-understood, traditional techniques. They offer the advantage of lower complexity and often require less computational power than more advanced AI-driven methods. Moreover, because these techniques are based on established knowledge and behaviors, they tend to generate fewer false positives when dealing with known scenarios. However, their major limitation is their diminished effectiveness against novel or sophisticated attacks that do not match the predefined patterns or rules, underscoring the necessity of integrating these with more dynamic, learning-based approaches in modern HIDS.

The operational mechanics of HIDS are complex and involve real-time monitoring and analysis to provide effective protection. This necessitates robust

computational capabilities, as the system must process and analyze vast amounts of data continuously. The effectiveness of HIDS hinges on its ability to seamlessly integrate with the host system, its sophisticated analysis of incoming data, and the robustness of its detection algorithms. The detailed understanding and continuous improvement of these aspects are crucial for enhancing the capability of HIDS to protect against the ever-evolving landscape of cybersecurity threats.

### **2.3 AI in HIDS**

Going in-depth into host-based intrusion detection systems (HIDS), recent research has demonstrated significant advancements by incorporating sophisticated AI methodologies, particularly through the analysis of system event logs and provenance graphs. Ring et al. [9] employed Long Short-Term Memory (LSTM) networks to enhance malware detection from Windows audit logs. This research utilized a feature extraction process that adapted the sequential and textual nature of log data into a format suitable for neural network analysis, highlighting the effectiveness of LSTMs in capturing temporal dependencies among log entries.

Further exploring the utility of neural networks in log analysis, Bai et al. [10] focused on detecting lateral movements within networks by leveraging the remote desktop (RDP) session logs. They successfully applied various ML techniques to classify session activities, achieving superior performance in identifying unauthorized access attempts, a critical component of early intrusion detection and response strategies.

Another notable approach in the realm of network security explores the use of Graph Neural Networks (GNNs) for detecting lateral movements in enterprise

networks. The Jbeil framework [11], adapts to the dynamic nature of modern network environments by employing inductive graph learning. This approach not only addresses the limitations of transductive learning in static graphs but also enhances the detection of stealthy, sophisticated threats by focusing on the evolving interactions within the network. Jbeil's capability to dynamically track lateral movements, despite the inherent latency issues in real-time data analysis, represents a significant leap in adapting graph-based methodologies to practical cybersecurity applications.

Simultaneously, the ATLAS framework [12] marks a substantial development in handling the complexities of Advanced Persistent Threats (APTs). ATLAS utilizes a sequence-based learning model constructed from audit logs to trace and reconstruct the steps of APTs. By analyzing attack symptoms and constructing attack narratives, this framework aids cyber analysts in piecing together disjointed attack signals, thereby addressing the challenge of alert fatigue and enabling more effective response strategies.

More sophisticated architectures are used in research, for example, Studiawan et al. [13] introduced a sentiment analysis approach using Gated Recurrent Unit (GRU) networks to detect anomalies in operating system logs. This method effectively addressed class imbalances in log data, which are common in real-world scenarios, by identifying negative sentiments indicative of potential security breaches. Similarly, [14] introduced DeepLog, a framework that models system logs as natural language sequences, allowing for dynamic learning of log patterns and real-time anomaly detection. Another innovative approach, GLAD-PAW [15], utilized graph-based techniques to analyze log data, demonstrating that embedding log entries in a graph structure could significantly enhance anomaly detection capabilities. This method used

a position-aware weighted graph attention network to identify unusual patterns in system logs, offering a new dimension to log analysis by integrating structural and content-based insights.

These studies collectively underscore the evolving landscape of HIDS, where the integration of deep learning and graph-based analytics into log analysis not only enhances the detection capabilities of traditional systems but also offers new methodologies for understanding complex patterns and interactions within system data. This shift towards more advanced analytical techniques in HIDS promises to improve the precision and efficiency of cybersecurity measures in response to the increasing sophistication of cyber threats.

## **2.4 Provenance Graphs**

Provenance graphs have emerged as a powerful mechanism within the domain of Host Intrusion Detection Systems (HIDS), offering a dynamic way to visualize and understand the intricate relationships and interactions within system activities. These graphs trace the lineage and interdependencies of various system events, providing a comprehensive map of digital actions that can be crucial for uncovering subtle, sophisticated threats that might otherwise evade detection by conventional methods.

Li et al. highlight in their survey [16] the use of provenance graphs for modeling threats with a high level of semantic expressiveness and historical correlation capabilities. They introduce the basic concepts and architecture necessary for integrating provenance graphs into threat detection frameworks, emphasizing their role in collecting, managing, and analyzing data to enhance security measures effectively.

Additionally, Dong et al. [17] provide insight into the industrial application and effectiveness of Provenance-Based Endpoint Detection and Response (P-EDR) systems. Despite their potential, the adoption of P-EDR systems faces challenges such as high operating costs and discrepancies between academic innovations and industry needs, reflecting a gap that future research must bridge.

Research toward provenance graphs excelled in advanced manners like with the PROGRAPHER framework [18], which demonstrates how embedding provenance graphs can enhance the detection of advanced persistent threats (APTs) by efficiently managing the complexity and volume of data in modern computing environments. On the other hand, Threatrace [19] and UNICORN [20] both utilize provenance graphs to provide a deep, contextual analysis of host-based data, tailoring their approaches to the unique dynamics of specific threats. These studies underscore the potential of provenance graphs to revolutionize the field of cybersecurity, offering new methodologies for robust and dynamic threat detection and analysis.

## **2.5 LLMs in HIDS**

In the realm of cybersecurity, the integration of Large Language Models (LLMs) has opened up new vistas for enhancing host-based intrusion detection systems (HIDS) by leveraging their advanced capabilities in natural language processing and understanding. The utilization of LLMs in cybersecurity, as discussed in a comprehensive survey [21], highlights their dual potential to both enhance security measures and introduce new vulnerabilities. The authors categorize the applications of LLMs into beneficial uses, such as improving code security and data privacy, and potential risks, such as enabling more sophisticated cyberattacks due to their advanced reasoning capabilities.

Furthermore, the concept of using LLMs for processing and analyzing structured data, such as graphs, introduces a novel approach to cybersecurity. As outlined in [22], LLMs are not just limited to text but can also be effective in scenarios where data comes with inherent structural relationships, such as social networks or biochemical structures. The review explores various methods of employing LLMs in these contexts, enhancing our understanding of how these models can be adapted to extract meaningful insights from complex data structures beyond plain text.

In the context of log anomaly detection, the authors of [23] introduce LogBERT, a model adapted from BERT (Bidirectional Encoder Representations from Transformers) specifically designed to understand and predict normal and anomalous patterns in system logs. This approach demonstrates how the transformer architecture, which underpins most LLMs, can be fine-tuned to capture the subtleties in sequential log data, providing a more nuanced detection of anomalies compared to traditional methods.

Expanding the scope of LLMs in cybersecurity, Happe et al. [24] explored the use of these models in penetration testing. They investigated how LLMs can function as 'AI sparring partners', aiding in high-level task planning and low-level vulnerability identification within cybersecurity operations. The interaction between LLMs and real-time system states via simulated environments showcases the practical applications of these models in dynamic testing scenarios.

Lastly, the integration of LLMs with ML techniques for network anomaly detection was further exemplified in [25], where HuntGPT combines the analytical power of LLMs with explainable AI (XAI) frameworks. This integration not only enhances the detection capabilities but also improves the transparency and user

understanding of the detection processes, addressing one of the major hurdles in the adoption of AI in cybersecurity.

These studies collectively underline the transformative impact of LLMs in cybersecurity, especially in the domains of log analysis, anomaly detection, and penetration testing. They also highlight the ongoing need to explore further the capabilities and implications of these technologies in safeguarding digital infrastructures.

# CHAPTER 3

## OVERVIEW OF LOG AUDITING

### 3.1 Windows Event Log

The detailed and systematic analysis of system behavior through logging mechanisms is vital for detecting anomalies and maintaining robust security protocols. This thesis leverages the rich data provided indirectly by Sysmon (System Monitor) and Windows Event Logs, both of which are essential for capturing comprehensive records of system activities and events. These tools not only enable a profound understanding of system operations but are also critical in identifying and analyzing potential security threats.

Windows Event Logs play a crucial role as an integral component of the Windows operating system. These logs are methodically structured to record a wide range of system activities, providing administrators and security professionals with insights necessary for effective system management and security monitoring:

- Application logs contain events posted by applications. These logs are invaluable for developers and IT professionals as they provide insights into application performance and highlight errors that may affect user experience or system stability.
- Security logs capture security-related events, such as account logon successes and failures, policy changes, and privilege escalations. These logs are essential for security auditing and compliance, as they help track unauthorized access attempts and other potentially malicious activities.

- System logs record events related to the core operations of the Windows system files. They are crucial for system administrators to diagnose hardware and software issues and to ensure the overall health of the system.
- Setup logs document the installation, updates, and removal of system software, which is critical for maintaining the integrity of the system and ensuring that all changes are accounted for and authorized.
- Forwarded Events represent logs collected from other machines within a network, enabling centralized monitoring and management of distributed systems and facilitating broader security oversight.

Sysmon [26], a robust tool from Microsoft's Sysinternals suite, extends the capabilities of Windows Event Logs by logging more granular information about process creations, network connections, and file modifications. Sysmon is designed to enhance the depth of security data available, providing detailed information that is crucial for comprehensive security analysis and incident response.

One of the most significant features provided by Sysmon is Process Execution Logging. This feature meticulously tracks the creation and termination of processes, offering detailed insights into the execution pathways within a system. Such logging is invaluable for identifying unauthorized or malicious activities, as it includes comprehensive information about parent processes. The lineage or traceability of a process, indicated by its parent process, is critical for understanding how potential malware or unauthorized actions propagate within a system.

Event Codes are another crucial feature within Windows Event Logs. Each event in the system is assigned a unique identifier, known as an event code, which helps in

categorizing and quickly identifying the nature of each event. For security-focused analysis, certain event codes are particularly pertinent:

- Event ID 4624: This event code signals a successful account logon, which can be crucial for tracking user activities and detecting unauthorized access.
- Event ID 4657: Indicates a registry value modification, which could signify unauthorized changes to system settings, potentially pointing to security breaches.
- Event ID 4688: Captures the creation of new processes, providing essential data that can be correlated with Sysmon logs to monitor process origins and behaviors in detail.

Event Viewer is the graphical interface used to navigate and analyze Windows Event Logs. It allows users to filter, search, and manage logs efficiently, making it an indispensable tool for security monitoring and system troubleshooting. Event Viewer's intuitive interface enables both novice and expert users to extract and analyze critical log data, facilitating rapid response to potential security incidents.

Figure 1 shows an example of a Sysmon log.

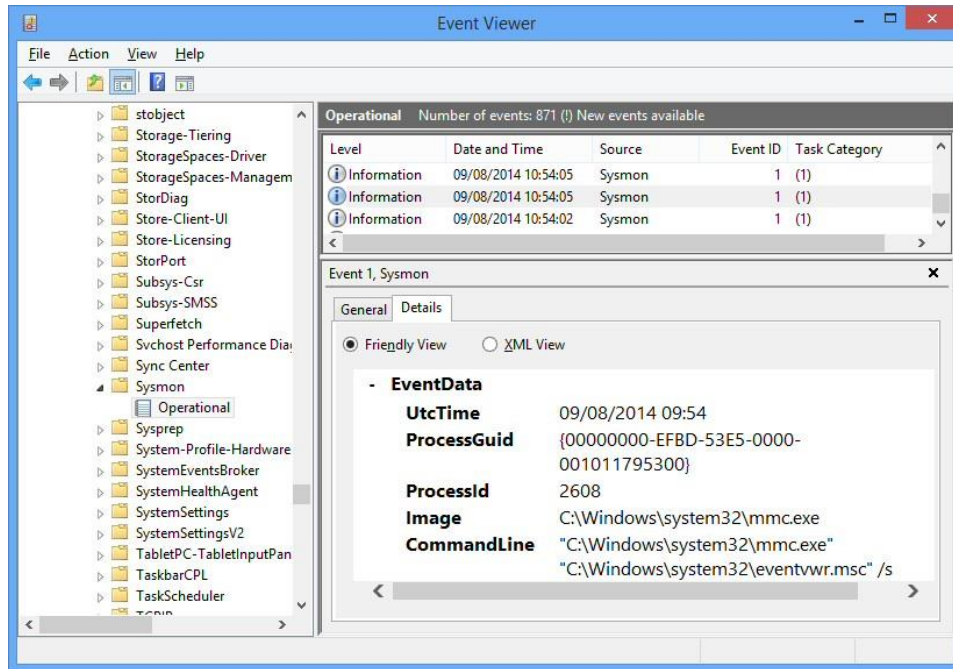


Figure 1: Sysmon Log Example

Through the integration of Sysmon’s detailed logging capabilities with the extensive data capture of Windows Event Logs, this thesis establishes a robust logging and monitoring framework. This comprehensive system significantly enhances cybersecurity defenses by providing the tools necessary for detecting, documenting, and analyzing security incidents effectively. This framework not only supports proactive security measures but also underpins reactive strategies by offering detailed forensic data that can be crucial in post-incident analysis and recovery efforts.

### 3.2 Endpoint Monitoring Architecture

The deployment of an effective Endpoint Monitoring Architecture is essential for the seamless operation and security integrity of Security Operations Centers (SOCs). This architecture not only enables the detection, analysis, and swift response to emerging threats but also ensures comprehensive surveillance over all network

endpoints, from user devices to servers and workstations. Herein, we discuss in detail the components and processes integral to the endpoint monitoring framework utilized in SOCs, underlining its crucial role in holistic cybersecurity management.

### ***3.2.1 Basics of Endpoint Monitoring***

Endpoint monitoring involves the continuous observation and analysis of activities at network endpoints—devices and systems connected to the network—to identify and mitigate potential security threats. The process is critical for the early detection of unauthorized actions, malware infections, and other security issues that could compromise organizational data integrity and confidentiality. The importance of endpoint monitoring lies in its ability to provide real-time security alerts and insights, facilitating immediate interventions to preemptively address security breaches before they escalate.

### ***3.2.2 Beats: Data Collection Mechanisms***

At the core of the endpoint monitoring architecture are Beats, a suite of lightweight, single-purpose data shippers developed by Elastic [27]. Each Beat is tailored to collect specific types of data and is deployed directly on the host it monitors. This setup minimizes resource usage while maximizing data collection efficiency. Key examples of Beats include:

- **Filebeat:** Tailored for harvesting logs from files, making it ideal for collecting data from local logs.
- **Metricbeat:** Gathers metrics and statistics, providing insights into system and service performance.

- Packetbeat: Captures network traffic data, offering visibility into the flow of information between systems.
- Winlogbeat: Specializes in shipping Windows event logs, crucial for monitoring Windows environments.

These Beats forward the collected data to Elasticsearch, a robust platform for data indexing, search, and analysis, ensuring efficient data handling and minimal impact on system performance.

### ***3.2.3 Elasticsearch and ElastAlert: Storage and Alert Management***

Elasticsearch is the backbone of the data storage and analysis system, equipped to handle vast volumes of data swiftly and in near real-time. This capability is pivotal for the SOC's ability to make data-driven security decisions quickly. Alongside, ElastAlert by Yelp enhances the architecture with powerful alerting capabilities that trigger notifications based on specific, predefined criteria, ensuring that potential security incidents are promptly addressed.

### ***3.2.4 Elastic Rules: Creation and Application Examples***

In the Elasticsearch environment, Elastic rules are crucial for defining specific alert conditions. These rules, structured in JSON format, can be intricately customized to meet diverse security needs. For instance, a rule may be configured to alert the SOC team about an unusually high volume of failed login attempts, potentially indicative of a brute-force attack. Another rule could monitor access to sensitive files, alerting on unauthorized retrieval attempts.

### ***3.2.5 TheHive: Incident Management Workflow***

TheHive [28] is a scalable, open-source incident response platform that significantly streamlines the management of security incidents. It enables SOC teams to manage the entire lifecycle of an incident efficiently—from detection through to resolution—within a collaborative environment. The SOC analyst can use TheHive platform to get an overview of the generated case (alert) in addition to all related Indicators of Compromise (IOCs).

### ***3.2.6 Investigation Process: From Alert Detection to Resolution***

The investigation process in a SOC unfolds through a structured sequence of steps, starting with the detection of an alert and culminating in the resolution of the incident. This process initiates when an alert from ElastAlert indicates a potential security issue. SOC analysts then leverage TheHive to manage and respond to the incident, assessing its severity and potential impact, gathering further information, and conducting an initial diagnosis. Depending on these initial findings, subsequent actions may involve deeper forensic analysis, implementing containment measures, or initiating recovery processes. Throughout, TheHive facilitates critical communication and collaboration, ensuring that all response team members are synchronized and that every action taken is thoroughly documented and auditable.

This sophisticated endpoint monitoring architecture represents a dynamic, robust approach to cybersecurity within a SOC. Integrating cutting-edge tools and detailed procedural workflows ensures that all network endpoints are under vigilant surveillance, with effective mechanisms in place to detect, analyze, and respond to potential threats. This architecture not only secures information assets but also supports regulatory

compliance, thereby enhancing the organization's overall security posture and resilience against cyber threats.

### ***3.2.7 Incident Example***

To illustrate the endpoint monitoring and response process in a Security Operations Center (SOC), consider the detection of a potentially malicious command executed on a network endpoint (Figure 2):

1. **Detection:** Filebeat is configured on a host to monitor Windows Event Logs for specific event IDs, such as 4688, which logs new process creations. It detects a suspicious command execution: `powershell.exe -ExecutionPolicy Bypass -NoProfile -EncodedCommand JABzAGMAcgBpAHAAAdA==`.
2. **Alert Generation:** The command triggers an alert because it matches a predefined Elastic rule in Elasticsearch. This rule is designed to identify patterns of script obfuscation and potential attacks.
3. **Alert Processing:** ElastAlert, integrated with Elasticsearch, processes this alert and automatically creates a case in TheHive for further investigation by SOC analysts.
4. **Investigation:** SOC analysts use TheHive for an overview of the reported case, further investigation includes the executing process and decoding of the PowerShell command line arguments confirming its malicious nature.
5. **Containment and Response:** Once confirmed as malicious, the SOC team initiates containment measures, such as isolating the affected endpoint and blocking the related network traffic.

6. Documentation and Review: All steps from detection to response are meticulously documented within TheHive. This documentation includes details of the investigative process and actions taken, serving as a critical audit resource and helping refine response strategies for future incidents.

This workflow demonstrates the coordinated use of various security tools within a SOC to manage real-time threats effectively, ensuring rapid detection, analysis, and response to maintain network integrity and security.

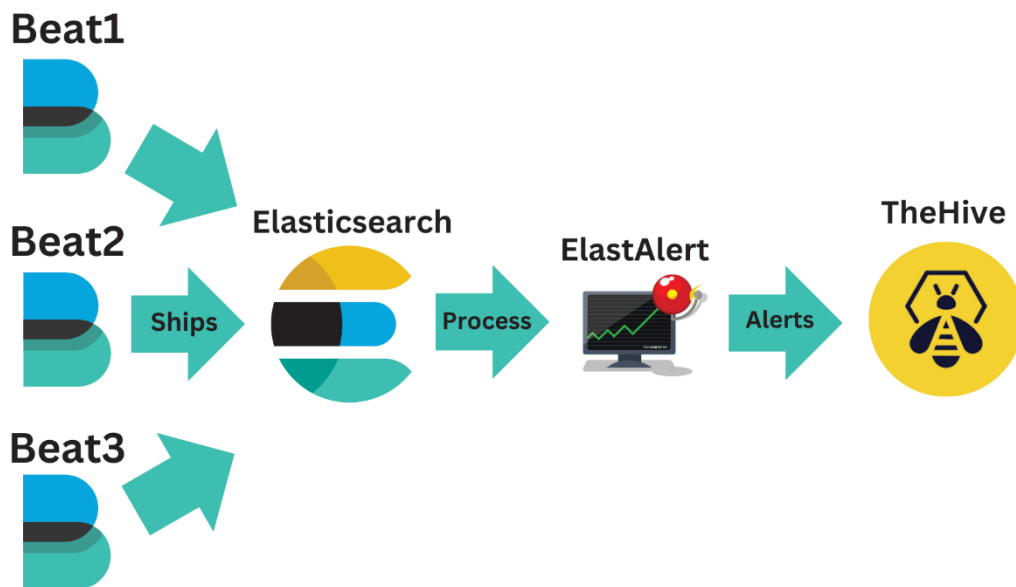


Figure 2: Overview of the Alerting Flow

### 3.3 Large Language Models (LLMs):

Large Language Models (LLMs) such as GPT-3, BERT, and others have significantly advanced the field of natural language processing. Their ability to process, generate, and understand complex language structures makes them particularly valuable in cybersecurity contexts. For example, they can analyze unstructured text data from

various sources, including audit logs and network traffic annotations, to extract meaningful patterns that are not immediately obvious to human analysts.

### ***3.3.1 Text Classification in Cybersecurity***

LLMs excel in text classification tasks, which are pivotal in cybersecurity for sorting vast amounts of data into actionable categories. For instance, these models can be trained to distinguish between benign and malicious activities within audit logs by recognizing subtle textual cues that differentiate normal system behavior from potential threats. An example could be classifying SSH login attempts as legitimate or suspicious based on the command sequences logged, aiding in the detection of brute force attacks or unauthorized access attempts.

### ***3.3.2 Anomaly Detection Techniques***

For anomaly detection, LLMs process sequences of audit data converted into tokens that represent discrete pieces of information. Each log entry, when tokenized, might include elements like operation codes, process identifiers, or user IDs, which the model learns to expect in specific sequences under normal operations. Deviations from these sequences, identified through ongoing analysis of live data, flag potential security incidents. An LLM trained on audit log data can classify these instances by analyzing the textual content of the logs and identifying unusual patterns or anomalies. For example, a typical benign command might be ``sudo apt-get update``, which a system routinely executes to update its software. In contrast, a command like `curl -s http://example.com/script.sh | sudo bash` might be flagged as suspicious. This command uses curl to download a script from a potentially untrusted URL and executes

it with root privileges, a common tactic used in payload executions to gain unauthorized access or install malware.

By tokenizing these command sequences and learning from vast datasets of similar logs, an LLM can discern between normal administrative tasks and potential security threats. It then classifies log entries accordingly, alerting cybersecurity teams to investigate further, thereby preventing possible breaches before they can cause harm. This ability to quickly and accurately parse through extensive log data and highlight anomalies plays a crucial role in modern security operations centers, enhancing their response capabilities and overall security efficacy.

### ***3.3.3 Enhancements in Security Applications***

Using LLMs to analyze and interpret audit data can dramatically enhance the efficiency of security operations centers (SOCs). These models provide deep insights into data trends and help predict future threats by learning from historical incidents. Their ability to process information at scale allows for real-time threat detection, significantly reducing the time between the onset of a breach and its detection. This rapid processing capability is crucial for implementing effective containment and mitigation strategies before significant damage occurs.

### ***3.3.4 Data and Computational Demands***

The deployment of LLMs in cybersecurity does come with its set of challenges, primarily related to data volume and computational intensity. The models require large datasets for training, which must be meticulously curated to include representative samples of both normal and anomalous behaviors. The computational power needed to

train and continuously run these models necessitates robust hardware and can lead to increased operational costs. Moreover, the handling of sensitive audit data requires strict adherence to data privacy laws and internal security policies to prevent data breaches and ensure compliance with regulatory standards.

By integrating LLMs into cybersecurity frameworks, organizations can leverage cutting-edge technology to bolster their defensive measures. These models not only speed up the detection of potential threats but also enhance the accuracy with which these are identified, thereby streamlining security operations and reinforcing the overall security posture.

## CHAPTER 4

### PROPOSED METHODOLOGY

#### **4.1 Testbed and Baseline Data**

To address the challenge of reducing false positives in a Security Operations Center (SOC) focused on Host Intrusion Detection Systems (HIDS), a comprehensive pipeline has been established, necessitating a robust testbed setup. This testbed is situated within a large enterprise environment and includes several machines, each outfitted with Winlogbeat agents. These agents are crucial for collecting and transmitting data, enhancing the detection accuracy and operational efficiency of the SOC. These agents are configured to collect and forward Windows Sysmon logs to a centralized Elasticsearch cluster.

Once these logs are ingested into Elasticsearch, they are processed and analyzed against a set of predefined SOC rules. These rules are designed to detect patterns and anomalies that may indicate malicious activity or policy violations. When logs match these detection rules, an alert is generated and escalated into a case within the SOC's management system.

At this stage, the SOC team steps in to conduct a thorough investigation of the alert. This involves analyzing the context of the alert, cross-referencing against known issues and external threat intelligence, and applying forensic analysis techniques where necessary. The outcome of this investigation leads to the case being labeled either as a true positive, confirming a legitimate security threat, or a false positive, where the alert is deemed non-malicious or erroneous.

In this project, the baseline data consists of logs formatted by Winlogbeat, which captures Windows Event Logs and structures them into a concise JSON format for Elasticsearch. Each log entry includes critical information encapsulated within key-value pairs, making them accessible for automated analysis. A simplified example of a Winlogbeat log entry might include fields like `@timestamp` for the event time, `event.id` for the unique identifier of the event, `process.name` for the name of the executing process, and `user.name` for the account under which the process is running. This format enables precise querying and analysis of specific log attributes, essential for monitoring and security assessments in a SOC environment.

## **4.2 Attack Simulation**

To address the challenge of distinguishing between true positives and false positives within a Security Operations Center (SOC) focused on Host Intrusion Detection Systems (HIDS), extensive simulation of cyber-attacks was undertaken. Given the rarity of true positive incidents—potentially due to robust security measures or successful blocking by defensive technologies—it was imperative to create a realistic testing environment to validate the detection capabilities of the SOC.

Utilizing the Atomic Red Team toolkit, an array of cyber-attack simulations was orchestrated to mimic real-world threats within the enterprise network. This initiative was critical in assessing and honing the accuracy and responsiveness of SOC systems and protocols. For instance, the "Domain Account" attack simulation (T1136.002) involved creating or manipulating domain accounts to test the SOC's ability to detect unauthorized changes that could allow attackers to establish persistence or escalate privileges within the network. Another key simulation was the "Domain Trust

Discovery" (T1482), designed to probe the system's trust relationships to uncover potential paths for lateral movement or escalation that attackers might exploit. Further simulations included "Boot or Logon Autostart Execution" (T1547), which aimed to evaluate the SOC's effectiveness in identifying and mitigating autostart configurations—a common technique used by malware to ensure persistence after system reboots. An additional scenario, "Security Software Discovery" (T1518.001), was conducted to test how well the SOC system could detect reconnaissance activities aimed at identifying and undermining security applications that protect the enterprise environment, in addition to multiple attack methods that were used to mimic real intrusion scenarios.

Each simulated attack was meticulously planned to generate logs that mimicked authentic interactions with the system's defenses. These logs were processed through the SOC's established pipeline—from initial data capture by Winlogbeat, through analysis and correlation in Elasticsearch, to eventual alert generation and case handling by SOC analysts. This comprehensive approach not only stress-tested the detection systems but also provided a practical, hands-on training platform for the SOC team. The simulations fostered a deeper understanding of attack methodologies and enhanced the team's ability to quickly and effectively respond to real threats, thereby strengthening the overall security posture and resilience of the enterprise against potential cyber-attacks.

### **4.3 Graph-Based Process Analysis**

In the Windows operating system, a process is defined as an instance of a computer program that is actively executing. It encompasses the program's code and its

current state, including the program counter, system registers, and allocated memory spaces. Traditional host-based intrusion detection systems typically look at each individual process in isolation to spot potential signs of malicious activity. This approach often overlooks the interconnected nature of processes and how they interact within the system's broader operational context. For instance, a scenario where a benign PDF reader launches a PowerShell script can be indicative of a fileless malware attack, exploiting legitimate processes for malicious activities.

The first step of the proposed methodology to address this complexity involves adopting a graph-based analytical approach. This technique is particularly advantageous as it allows for a comprehensive examination of how processes interact within the system's broader operational context. It involves mapping out the process execution timelines and the interactions among various processes. This is done by creating a visual representation, or a graph, where each node represents a process and the edges depict the interactions or communications between these processes. Such a graph not only illustrates the parent-child relationships among processes but also highlights anomalous patterns like unexpected process hierarchies or unusual process behaviors, which are often signs of sophisticated multi-stage cyber-attacks. This method enhances the detection capabilities of intrusion detection systems by providing a holistic view of the system's process activities and their interdependencies.

To map the raw baseline logs into a JSON process execution map effectively, the process involves parsing and transforming the Winlogbeat detailed log data. Each log entry, formatted in JSON, includes comprehensive details such as the event's timestamp, process identifiers, and associated user information. These elements are critical for tracing the sequence and hierarchy of process executions within the system.

The technical aspect of this mapping focuses on the extraction of key parameters such as `process_id` and `parent_process_id` from the log data. These parameters are essential for establishing the parent-child relationships between processes. The script systematically processes each log entry, using the `parent_process_id` to link child processes to their respective parents, thus constructing a hierarchical model of process interactions. This model provides a clear visualization of the process flow across the system, highlighting how different processes are spawned and how they interact. Figure 3 shows an example of a generated graph for an alert.

To enhance the usability and analytical value of the JSON process maps, additional contextual data is integrated, such as command-line arguments and network connections linked to each process. This enriched JSON structure allows for a more detailed analysis, helping to identify potential security threats such as unauthorized access or malware execution. The mapping script also incorporates error handling to ensure that anomalies or discrepancies in the data, such as orphaned processes without a clear parent, are flagged and can be investigated further.

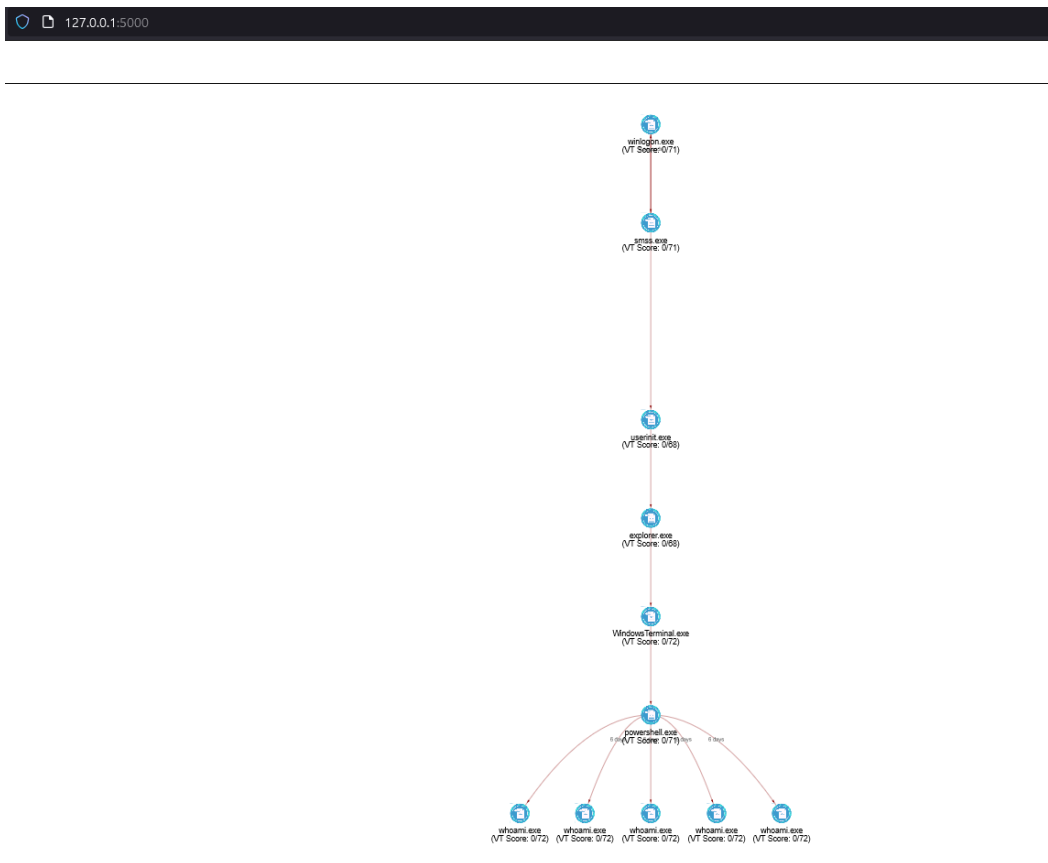


Figure 3: Generated Graph of an example alert

### 4.4 Graph Contextualizer

The "Graph Contextualizer" module introduces a transformative approach to processing security log data, designed to leverage the capabilities of Large Language Models (LLMs) in understanding and predicting security threats within an enterprise environment. This module refines raw JSON graph logs, which are typically cluttered with extensive system details and personal user data, into an anonymized, structured narrative format that LLMs can process more effectively. It is worth mentioning that this approach hasn't been used in previous research, and after leveraging it, the results improved exponentially.

One crucial functionality of the Graph Contextualizer is the anonymization of data. It strips out all personal identifiers and user-specific information from the logs. By focusing exclusively on the behavior of processes rather than on user identities, the module not only ensures compliance with privacy regulations but also eliminates the risk of bias that could skew the analysis. This step is vital for maintaining the integrity of the security analysis, ensuring that the focus remains solely on the technical and behavioral patterns that might indicate security threats.

Another key functionality is the contextualization of data. This process converts the JSON-like technical details from the logs into a form of narrative data that describes the interactions between processes in a story-like format and can be fed into LLM for inference. For instance, if a benign process like a PDF reader unexpectedly executes a PowerShell script, the module will detail this anomaly by describing the sequence of events, the relationship between these processes, and the temporal gaps between their executions.

The module uses a series of algorithms to determine the temporal relationships between events, calculating time differences using precise date-time parsing to contextualize how events unfold over time. It might describe a process spawning another as happening "seconds after" the parent process started, providing clear chronological insights into process behavior. Each log entry is transformed into a detailed description that includes essential attributes such as the process name, PID, executable path, file size, and MD5 status. The descriptions also integrate security analysis details like virus total scores (if applicable), which help in assessing the threat level of each process. Arguments passed to executable files are also logged in a sanitized form to provide context without revealing sensitive command-line

information. Additionally, the Graph Contextualizer builds narrative links between parent and child processes. It notes the creation of new processes, but also contextualizes these events within the broader activity of the system, identifying and narrating potential unauthorized escalations or lateral movements within the network. This is crucial for detecting sophisticated multi-stage attacks where the sequence and nature of process interactions are more indicative of a breach than the individual actions themselves.

## **4.5 LLM-Based Classification**

### ***4.5.1 Creating a Dataset***

The process of training our Large Language Model (LLM) on alert logs begins with the creation of a dataset structure derived from the output of the Graph Contextualizer. This dataset will hold the cases that are used to train the LLM with labels as either 'TP' or 'FP' resembling true positives or false positives. To ensure comprehensive training and validation, the dataset is divided into training, evaluation, and testing sub-datasets.

### ***4.5.2 Creating a Tokenizer***

Tokenization is a crucial preprocessing step in training LLMs especially when preparing contextualized text data for model ingestion. This process involves parsing descriptive log entries into manageable pieces, or tokens, that represent words or subwords, helping the model capture the underlying semantics of the text. In this application, a tokenizer like `BertTokenizer` or `GPT2Tokenizer` from Hugging Face's [Transformers](#) [29] library is used, as these can effectively handle the specialized

vocabulary and formats of system logs. These tokenizers are adept at breaking down both common language and domain-specific terms such as IP addresses, file paths, or command-line arguments, ensuring the model grasps the context and significance of these elements within security-related data.

#### ***4.5.3 Building the LLM with a Classification Head***

The construction of the LLM involves integrating a classification head with a pre-trained base model, such as one from the popular `Transformers` library using `AutoModelForSequenceClassification.from_pretrained`. This configuration is designed to classify input sequences into categories—TP or FP (True Positive or False Positive). The classification head is typically a neural network layer that maps the encoded text representations to a label and is fine-tuned during training to optimize classification accuracy.

#### ***4.5.4 Finetuning the Model***

Finetuning the LLM involves using a comprehensive set of training parameters that define how the model learns from the data. This includes setting the number of training epochs, batch size, learning rate, and other hyperparameters that influence training dynamics. The `Trainer` class facilitates this process by automating batch processing, gradient updates, and checkpoint saving, making it easier to maintain and iterate on the model's training regimen.

#### 4.5.5 Inference and Performance Metrics

Post-training, the model is subjected to rigorous testing to evaluate its performance on the evaluation set. This stage is critical for understanding the model's practical capabilities and readiness for deployment. Performance metrics such as accuracy, precision (the model's ability to identify true anomalies), recall (the model's ability to capture all potential anomalies), and the F1-score (a harmonic mean of precision and recall) provide insights into the model's effectiveness. These metrics help in fine-tuning the model's parameters before deployment and setting benchmarks for its operational performance in real-world SOC environments. Figure 4 illustrates the proposed pipeline. The actual experimentations and results will be shown in the next section.

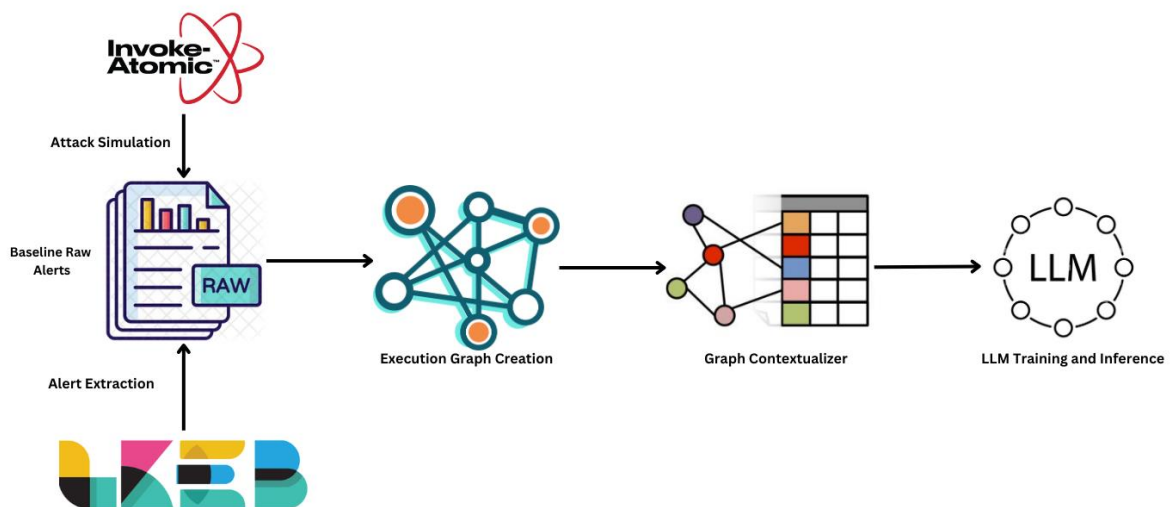


Figure 4: General Overview of the Proposed Pipeline

## CHAPTER 5

### EXPERIMENTATION AND RESULTS

#### 5.1 Experimentation Dataset

After automating the procedure of log extracts, label extraction, logs contextualization, and attack simulation, the resultant dataset is generated to be used for training and testing several LLMs and to choose the best candidate for later deployment.

The dataset used for the experiments in this study comprises a total of around 700 cases, which have been divided into training, testing, and evaluation sets (70% training, 15% evaluation, and 15% testing). This dataset includes around 550 cases labeled as false positives and 150 identified as true positives. This distribution was specifically chosen to reflect the real-world scenario where false positives are predominant, presenting a unique challenge in tuning the model to effectively discriminate between true and false alerts.

For the sake of removing ambiguity between the terms 'false positive' and 'true positive' used for label names and the true positive and false positive results in evaluation metrics:

- False positives will be referred to as "label 0" or "fake threats".
- True positives will be referred to as "label 1" or "real threats".

#### 5.2 Experimentation Metrics

It is important to reiterate that the main target is to maximize the detection of 'fake threats' to lower the workload on SOC analysts and book their time on real investigations, while at the same time never missing the real threats. It would be counterproductive for a system to identify a "fake threat" while overlooking actual "real

alerts”. A clear parameter definition needs to be created to numerically monitor if the goal is reached, as shown in the confusion matrix (Figure 5). To understand how well the model performs, it's essential to understand the entries in the confusion matrix:

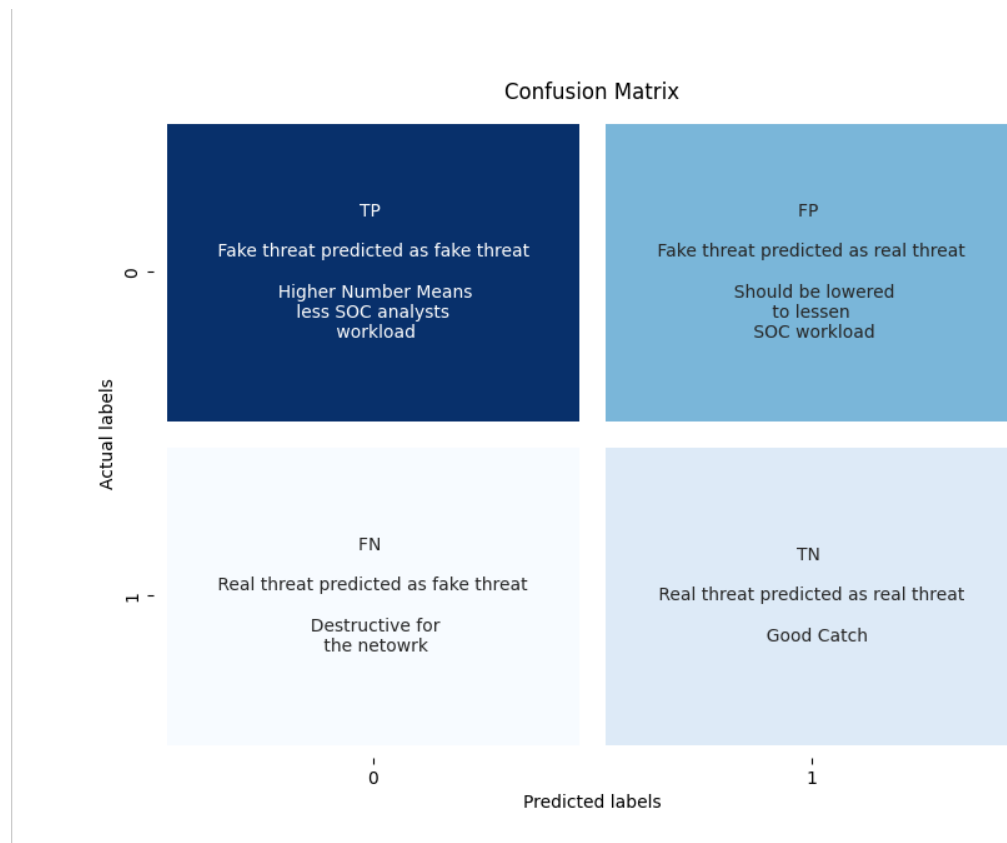


Figure 5: Experiment Confusion Matrix defining metrics of interest

Given this matrix, our aim is to maximize TN and TP, while minimizing FP and FN.

The evaluation metrics such as precision, recall, and F1-score are employed to monitor the effectiveness of the model:

- **Precision**  $\frac{TP}{TP + FP}$ : measures how many of the alerts labeled as fake threats

(label 0) are indeed fake, thus reducing false positives and lowering SOC workload.

- **Recall**  $\frac{TP}{TP + FN}$ : ensures that the system captures all real threats (label 1).
- **F1-score**  $\frac{2 * (Precision * Recall)}{Precision + Recall}$ : provides a balance between precision and recall, especially useful in this context of imbalanced datasets where the goal is to maintain high precision without sacrificing recall.
- **F2-score**  $\frac{(1+2^2) * (Precision * Recall)}{(2^2 * Precision) + Recall}$ : places greater emphasis on recall compared to precision, making it particularly useful in contexts where detecting all actual threats (minimizing false negatives) is more crucial than reducing false positives.

To achieve the target of reducing false positives while not missing real threats, we will closely monitor these metrics but especially the F2 score which gives an understanding of how the LLM is performing versus FN and FP while biased toward not allowing any real threats to pass, and reducing the false alarms.

### 5.3 Experimentation Results and Analysis

The next step was to train a group of LLMs of different sizes. We choose models smaller than 7B where research shows smaller models excel in domain-specific tasks like the one at hand [30]. The group of LLMs chosen in addition to their confusion matrices on inference is shown in the table below (Figure 6):

Model	TP	FP
	FN	TN
GPTNeo	70	13
	12	13
DistilledBert	78	5
	0	25
TinyLlama	82	1
	0	25
GPT2	82	1
	2	23
Roberta	82	1
	2	23
DeepSeek	82	1
	1	24
Phi	81	2
	24	1

Figure 6: Chosen Models in addition to their Confusion Matrixes

Based on the confusion matrices generated, we can calculate our desired metrics and compare the models accordingly:

	Precision	Recall	Accuracy	F1 Score	F2 Score
<b>GPTNeo</b>	0.677	0.682	0.769	0.679	0.68
<b>DistilledBert</b>	0.917	0.97	0.954	0.939	0.96
<b>TinyLlama</b>	0.981	0.994	0.991	0.987	0.99
<b>GPT2</b>	0.967	0.954	0.972	0.96	0.96
<b>Roberta</b>	0.967	0.954	0.972	0.96	0.96
<b>DeepSeek</b>	0.974	0.974	0.981	0.974	0.97
<b>Phi</b>	0.552	0.508	0.759	0.467	0.52

The final results demonstrate that the best-performing model is TinyLlama, achieving the highest F2 score of 99%. This performance highlights TinyLlama's ability to excel in both precision and recall, indicating its effectiveness in accurately identifying both false and real threats. The F2 score is particularly relevant here as it provides a combined metric that emphasizes recall more than precision, which is critical in security contexts. A high recall ensures that nearly all actual threats are detected, thereby minimizing the risk of malicious activity being overlooked.

It's important to note that while some other models may also exhibit strong performance, the sensitivity of the task at hand demands exceptionally high metrics. Therefore, the rigorous evaluation of the models was focused on achieving a balance where both precision and recall are maintained at optimal levels. The F2 score was chosen to underscore a combination of recall and precision metrics with a bias toward recall, but it's imperative that both precision and recall are sufficiently high to deem a model acceptable.

In practical terms, this means that TinyLlama not only reduces the workload on SOC analysts by accurately identifying false positives but also ensures that real threats are not missed. This dual capability is essential for maintaining robust security operations, as it allows the SOC to focus its resources on genuine threats while minimizing the distraction caused by false alarms. The meticulous evaluation and selection process underscores the critical need for a model that can reliably handle the high stakes of cybersecurity in an enterprise environment.

#### **5.4 Future Work**

In future work, the primary objective will be to deploy the trained model within the enterprise environment to monitor threats in real time. This deployment will operate in parallel with existing systems to test the model in a live setting. Due to the lack of publicly available benchmarking datasets for this task—primarily due to the confidential nature of audit logs—this in-the-wild evaluation is crucial. It will provide a robust test of the model's capabilities in detecting both false and real threats under real-world conditions. This approach allows for continuous assessment and validation, ensuring the model performs effectively outside of a controlled environment.

In addition to real-time deployment, an ongoing retraining strategy will be implemented. As cyber threats continually evolve and adapt, the model must be capable of learning from new data and threat patterns. This continuous retraining will ensure that the model remains current with the latest attack techniques and trends, thereby maintaining its efficacy over time. Regular updates to the model will also help in fine-tuning its precision and recall metrics, further reducing false positives and enhancing the detection of true threats.

## 5.5 Conclusion

This thesis has tackled the significant challenge of reducing false positives in Security Operations Centers (SOCs), an area critical to the efficiency and effectiveness of cybersecurity measures within large enterprises. Leveraging the capabilities of Large Language Models (LLMs), we developed a sophisticated methodology that enhances the accuracy of detecting and classifying security threats from Windows endpoint logs.

The research journey began with the assembly of a dataset consisting of both false and true positive alerts through a full-fledged pipeline, meticulously curated to reflect the real-world conditions of an enterprise environment. By converting these alerts into a structured format suitable for deep learning, specifically through the innovative use of Execution Graphs and the Graph Contextualizer, we were able to train LLMs to discern between fake and real threats with high metrics.

Our models demonstrated considerable promise in initial tests, showing a strong ability to reduce the number of false positives—thereby alleviating the workload on SOC analysts—without compromising the detection of real threats. This balance is crucial in cybersecurity, where the cost of missing a real attack can be extraordinarily high.

Looking ahead, the deployment of this model in a live setting presents an exciting next step. Real-world applications will allow for continuous model training and improvement, adapting to new threats as they evolve. Moreover, integrating this model with existing network monitoring and SIEM systems will further enhance its predictive capabilities and responsiveness.

Additionally, future work will aim to enhance the model's transparency and interpretability. It's essential for SOC analysts to understand and trust the model's decisions; thus, efforts will be directed toward developing clearer visualization tools and decision-making processes. These tools will transform complex model outputs into actionable insights that can be readily used in the fast-paced environment of a SOC.

In summary, this thesis not only advances the field of cybersecurity but also offers a practical solution that can be implemented in SOCs to improve their operational efficiency and security posture. The integration of advanced machine learning techniques with traditional cybersecurity practices represents a significant step forward in the ongoing battle against cyber threats.

## REFERENCES

- [1] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1-22, 2019.
- [2] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey," *applied sciences*, vol. 9, no. 20, p. 4396, 2019.
- [3] K. Soliman, M. A. Sobh, and A. M. Bahaa-Eldin, "Survey of machine learning HIDS techniques," in *2021 16th International Conference on Computer Engineering and Systems (ICCES)*, 2021: IEEE, pp. 1-5.
- [4] A. Chawla, B. Lee, S. Fallon, and P. Jacob, "Host based intrusion detection system with combined CNN/RNN model," in *ECML PKDD 2018 Workshops: Nemesis 2018, UrbReas 2018, SoGood 2018, IWAISe 2018, and Green Data Mining 2018, Dublin, Ireland, September 10-14, 2018, Proceedings 18*, 2019: Springer, pp. 149-158.
- [5] D. Pujol-Perich, J. Suárez-Varela, A. Cabellos-Aparicio, and P. Barlet-Ros, "Unveiling the potential of graph neural networks for robust intrusion detection," *ACM SIGMETRICS Performance Evaluation Review*, vol. 49, no. 4, pp. 111-117, 2022.
- [6] M. A. Ferrag, M. Ndhlovu, N. Tihanyi, L. C. Cordeiro, M. Debbah, and T. Lestable, "Revolutionizing cyber threat detection with large language models," *arXiv preprint arXiv:2306.14263*, 2023.
- [7] M. Landauer, S. Onder, F. Skopik, and M. Wurzenberger, "Deep learning for anomaly detection in log data: A survey," *Machine Learning with Applications*, vol. 12, p. 100470, 2023.
- [8] J. H. Ring IV, C. M. Van Oort, S. Durst, V. White, J. P. Near, and C. Skalka, "Methods for host-based intrusion detection with deep learning," *Digital Threats: Research and Practice (DTRAP)*, vol. 2, no. 4, pp. 1-29, 2021.
- [9] M. Ring, D. Schlör, S. Wunderlich, D. Landes, and A. Hotho, "Malware detection on windows audit logs using LSTMs," *Computers & Security*, vol. 109, p. 102389, 2021.

- [10] T. Bai, H. Bian, A. Abou Daya, M. A. Salahuddin, N. Limam, and R. Boutaba, "A machine learning approach for rdp-based lateral movement detection," in *2019 IEEE 44th Conference on Local Computer Networks (LCN)*, 2019: IEEE, pp. 242-245.
- [11] J. Khoury, D. Klisura, H. Zanddizari, G. Parra, P. Najafirad, and E. Bou-Harb, "Jbeil: Temporal graph-based inductive learning to infer lateral movement in evolving enterprise networks," in *2024 IEEE Symposium on Security and Privacy (SP)*, 2023: IEEE Computer Society, pp. 9-9.
- [12] A. Alsaheel *et al.*, "{ATLAS}: A sequence-based learning approach for attack investigation," in *30th USENIX security symposium (USENIX security 21)*, 2021, pp. 3005-3022.
- [13] H. Studiawan, F. Sohel, and C. Payne, "Anomaly detection in operating system logs with deep learning-based sentiment analysis," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 5, pp. 2136-2148, 2020.
- [14] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 1285-1298.
- [15] Y. Wan, Y. Liu, D. Wang, and Y. Wen, "Glad-paw: Graph-based log anomaly detection by position aware weighted graph attention network," in *Pacific-asia conference on knowledge discovery and data mining*, 2021: Springer, pp. 66-77.
- [16] Z. Li, Q. A. Chen, R. Yang, Y. Chen, and W. Ruan, "Threat detection and investigation with system-level provenance graphs: A survey," *Computers & Security*, vol. 106, p. 102282, 2021.
- [17] F. Dong *et al.*, "Are we there yet? an industrial viewpoint on provenance-based endpoint detection and response tools," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2023, pp. 2396-2410.
- [18] F. Yang, J. Xu, C. Xiong, Z. Li, and K. Zhang, "{PROGRAPHER}: An Anomaly Detection System based on Provenance Graph Embedding," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 4355-4372.

- [19] S. Wang *et al.*, "Threatrace: Detecting and tracing host-based threats in node level through provenance graph learning," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3972-3987, 2022.
- [20] X. Han, T. Pasquier, A. Bates, J. Mickens, and M. Seltzer, "Unicorn: Runtime provenance-based detector for advanced persistent threats," *arXiv preprint arXiv:2001.01525*, 2020.
- [21] Y. Yao, J. Duan, K. Xu, Y. Cai, Z. Sun, and Y. Zhang, "A survey on large language model (llm) security and privacy: The good, the bad, and the ugly," *High-Confidence Computing*, p. 100211, 2024.
- [22] B. Jin, G. Liu, C. Han, M. Jiang, H. Ji, and J. Han, "Large language models on graphs: A comprehensive survey," *arXiv preprint arXiv:2312.02783*, 2023.
- [23] H. Guo, S. Yuan, and X. Wu, "Logbert: Log anomaly detection via bert," in *2021 international joint conference on neural networks (IJCNN)*, 2021: IEEE, pp. 1-8.
- [24] A. Happe and J. Cito, "Getting pwn'd by ai: Penetration testing with large language models," in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2023, pp. 2082-2086.
- [25] T. Ali and P. Kostakos, "HuntGPT: Integrating machine learning-based anomaly detection and explainable AI with large language models (LLMs)," *arXiv preprint arXiv:2309.16021*, 2023.
- [26] M. Russinovich. "Sysmon." <https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon> (accessed July, 2022).
- [27] "Elastic." <https://www.elastic.co/> (accessed August, 2023).
- [28] "TheHive Project." <https://thehive-project.org> (accessed August, 2023).
- [29] T. Wolf *et al.*, "Huggingface's transformers: State-of-the-art natural language processing," *arXiv preprint arXiv:1910.03771*, 2019.
- [30] F. Trad and A. Chehab, "Prompt engineering or fine-tuning? a case study on phishing detection with large language models," *Machine Learning and Knowledge Extraction*, vol. 6, no. 1, pp. 367-384, 2024.