THESIS ABSTRACT

Master of Science in Applied Computer Science
Cybersecurity Option

Adventist University of Africa

School of Post Graduate Studies

Title: INVESTIGATING THE THREAT OF ADVERSARIAL MACHINE LEARNING ATTACKS ON AI-POWERED SECURITY INFORMATION AND EVENT MANAGEMENT SYSTEMS

Name of Researcher: Kikandi Safari Isaac

Primary Adviser: Paul-Marie Moulema, PhD

Date: May 2025

In the evolving landscape of cybersecurity, Security Information and Event Management systems have emerged as the nerve centers of modern defense, correlating logs, detecting anomalies, and delivering alerts leveraging AI-powered intelligence. However, as organizations rely more on these machine learning components, a new threat arises—AML. These attacks don't crash systems or flood networks, they manipulate data just enough to fool the AI into silence or confusion. This subtlety makes them both dangerous and difficult to detect.

This research set out to simulate such a scenario by incorporating two neural network architectures: a Feedforward Neural Network and a Convolutional Neural

Network as detection engines of a SIEM setting. Using a cleaned and balanced sample of the CICIDS2017 dataset, the models were trained and then tested by two evasion attack frameworks: FGSM and Projected Gradient Descent. These attacks, selected for their computational efficiency and widespread use in adversarial research, were applied under white-box conditions using the Adversarial Robustness Toolbox.

The results told a striking story: the FGSM attack reduced the FFNN's detection rate from 95% to just 4.6%, while the PGD attack caused a substantial decline in the CNN's recall, dropping it to 25%. While the CNN fared better under FGSM, it responded by flagging benign traffic as malicious, raising the risk of alert fatigue. Metrics such as precision, recall, and attack success rates clearly illustrated how easily an AI model's performance could be compromised with subtle adversarial effort.

These findings confirm that without proper defenses, AI-driven SIEMs are critically exposed and the study recommends embedding robust countermeasures such as adversarial training, anomaly detection, and model hardening to safeguard these systems. As cyber threats grow smarter, so too must our defenses be, not only in strength, but in foresight

Adventist University of Africa

School of Postgraduate Studies

INVESTIGATING THE THREAT OF ADVERSARIAL MACHINE LEARNING ATTACKS ON AI-POWERED SECURITY INFORMATION AND EVENT MANAGEMENT SYSTEMS

A thesis

presented in partial fulfilment

of the requirements for the degree

Master of Science in Applied Computer Science
Cybersecurity Option

by

Kikandi Safari Isaac

May 2025

INVESTIGATING THE THREAT OF ADVERSARIAL MACHINE LEARNING ATTACKS ON AI-POWERED SECURITY INFORMATION AND EVENT MANAGEMENT SYSTEMS

A thesis

presented in partial fulfillment

of the requirements for the degree

Master of Science in Applied Computer Science
Cybersecurity Option

by

Kikandi Safari Isaac

APPROVED BY:

Primary Adviser Paul-Marie Moulema, PhD	Programme Coordinator Lossan Bonde, PhD
Secondary Adviser Jules Pagna Disso, PhD	Head of Department, Applied Sciences Musa Nyakora, PhD
External Examiner Collins Oduor, PhD	Dean, School of Postgraduate Studies Lossan Bonde, PhD

Date: May, 2025

This thesis is dedicated to my dear parents—Prof. Paluku Mwendambio, PhD

and Mme Kahambu Bwasingo, MBA.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

First and foremost, I wish to acknowledge the immeasurable grace of God. Without His intervention—in health, strength, wisdom, and through countless unseen circumstances—this journey would not have reached this stage. I can only stand in gratitude and recognize the miracles He has accomplished throughout this process.

I would also like to thank all those who have supported me along the way. To my parents—your sacrifices, encouragement, and unwavering belief in me have been foundational. To my friends, thank you for keeping my spirits high when the work became overwhelming. Your constant encouragement made a real difference.

I'm grateful to my colleagues—both at school and at work—who offered encouragement and kept reminding me of the bigger picture. I owe special appreciation to my thesis supervisors, Dr. Jules Pagna and Dr. Paul-Marie Moulema and to the program coordinator Dr Lossan Bonde. This thesis would not exist without your guidance, insights, and patience. I have learned more from your example than words can express.

To the Adventist University of Africa, thank you for the opportunity to grow both academically and personally. The support—be it through bursaries or academic resources—enabled me to overcome real barriers along the way. I'm especially thankful to the University of Eastern Africa, Baraton community. Your prayers, support, and constant motivation made a difference. My gratitude also goes to the UEAB Administration, whose understanding and flexibility allowed me to balance my academic responsibilities with my professional obligations.

Finally, to everyone who contributed to this journey in big or small ways, may God continue to bless and guide you as He has guided me through this chapter.

CHAPTER 1

INTRODUCTION

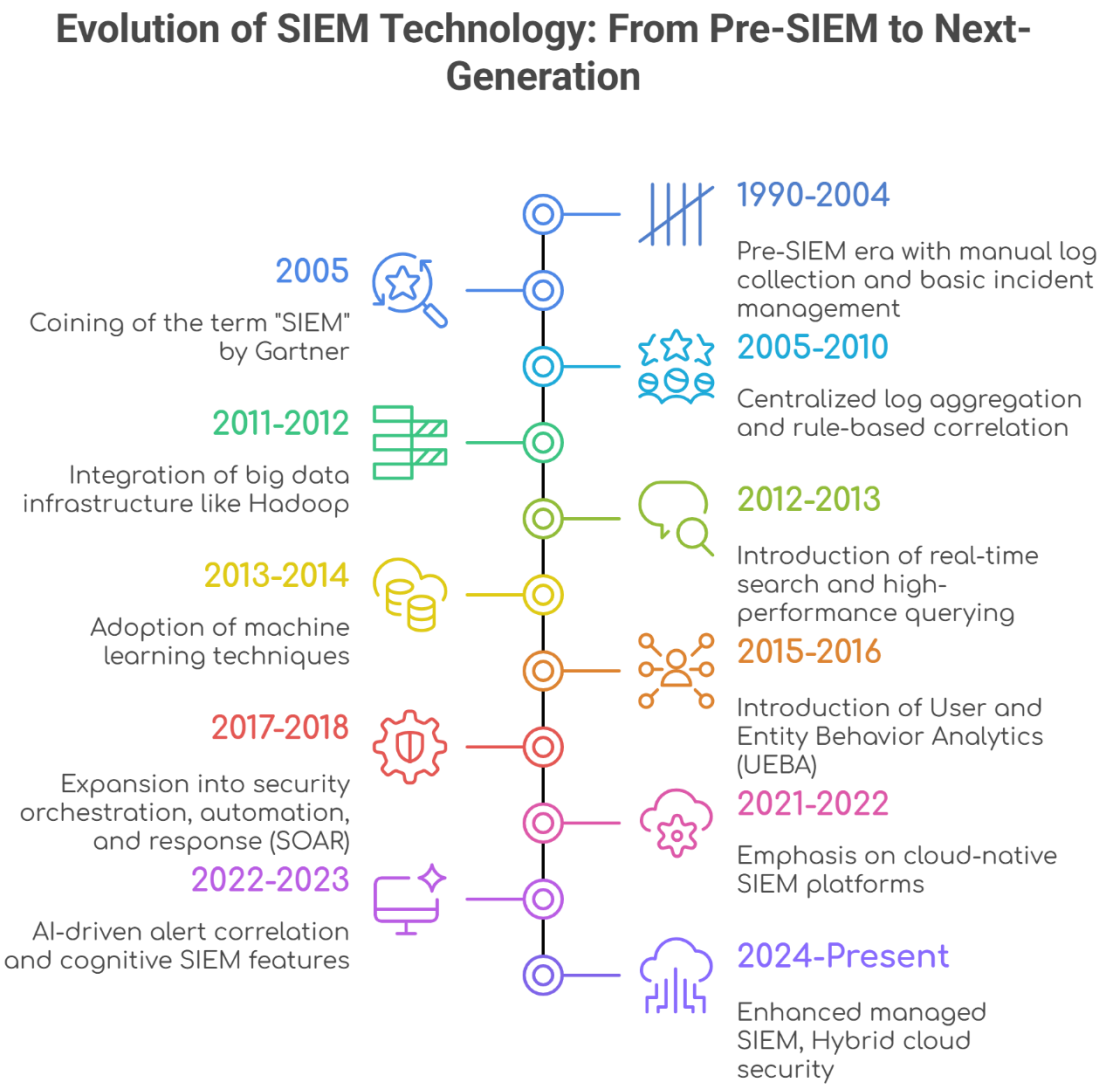
Background

The integration of Artificial Intelligence (AI) and Machine Learning (ML) technologies has significantly enhanced the capabilities of Security Information and Event Management (SIEM) systems throughout the past decade, enabling them to analyze large volumes of data, discover trends, and predict potential threats, thereby enhancing the resilience and proactiveness of imposing security measures[1]. SIEMs have significantly evolved throughout the years. Historically, SIEM systems depended on rule-based frameworks to detect and respond to security incidents; they would contain a database of signatures and behavioral patterns against which they could determine whether specific traffic patterns or data were malicious or not [2].

However, as cyber threats become increasingly sophisticated, the limitations of these traditional systems have become ever more apparent. The next-generation SIEMs introduced new features like user behavior analytics, automation, and threat intelligence[3]. The current SIEMs offer cloud-native scalability, rapid search performance, and advanced AI-powered analytics. [1] According to a Skyquest report, the market for SIEM solutions has grown significantly throughout the last decade and is projected to grow even more, from USD 5.21 billion in 2022 to over USD 14 billion by 2032, reflecting the rising demand for advanced cybersecurity solutions that leverage AI and ML capabilities[4].

Figure 1

Evolution of SIEM[5]



AI-powered SIEM systems utilize predictive analytics to anticipate future threats by learning from historical security data. This proactive approach enables organizations to detect and mitigate potential threats before they materialize, significantly reducing the Mean Time to detect (MTTD) and the Mean Time to respond/react (MTTR), preventing substantial damage beforehand[3]. AI also improves threat detection accuracy by detecting subtle abnormal behaviors in our systems that may indicate potential security breaches. Some of these irregularities are

so discrete that even highly qualified security analysts will often struggle to detect them, and this is where AI/ML helps to minimize false positives and enable security personnel to focus on genuine threats [6].

AI and ML have become vital tools in the digital world, extending beyond cybersecurity. They provide advanced functionalities in automation and data analytics in many major industries. Throughout time, AI has essentially referred to the intelligence displayed by machines or software, allowing them to perform tasks that typically would require human-like intelligence, such as learning, problem-solving, and decision-making. However, with the current growth in computing technology and capacity, AI is now being considered to do even more than the human brain can process in terms of speed, accuracy, and sustained attention [6]. Machine learning (which is a subset of AI), on the other hand, focuses on constructing algorithms that learn from data[7]. These technologies can analyze enormous data sets, discover trends, and automate choices that are critical for responding to fast-evolving cyber threats that nowadays change in seconds.

Despite the promising advances brought in by AI and ML, these technologies presented new vulnerabilities, which led to the emergence of Adversarial Machine Learning (AML) Attacks. These carefully crafted manipulations of AI models were primarily studied in the context of computer vision and image classification. However, researchers quickly discovered that these vulnerabilities extended to many other domains, including cybersecurity[8]. Hence, AI and ML have become prominent tools for threat actors because they know that these models are being increasingly used in various key corporate and industrial areas such as Financial Services, Healthcare, Manufacturing, mining, and even Digital security. Techniques such as adversarial training have since then emerged to try to mitigate these risks related to

AI/ML, however, the research on the effectiveness of this approach has not been thoroughly validated yet.

Statement of the problem

The integration of AI into SIEM systems is primarily meant to improve their ability to protect other systems and maintain their security through ML's ability to analyze vast amounts of data efficiently [3]. However, a crucial yet vulnerable component of SIEM systems is the data that is being analyzed. This is particularly important because modern Security Operation Centres (SOCs), which handle data from large industries and enterprises, often lack the time and resources to have analysts identify and analyze every abnormal deviation in their networks and system processes. As a result, they heavily depend on the SIEM's analytical capabilities and base their decision-making on the results that are generated by these systems. That is why data accuracy is critical because minor alterations in values lead to significant vulnerability and breach oversights crippling the SIEM system's accuracy and proactiveness. This is how AML attacks pose a serious threat to AI models across security systems that are AI-powered. These attacks can take different forms, including data poisoning, where malicious actors manipulate training data to mislead the model, evasion attacks that allow attackers to bypass detection systems by subtly altering their behavior, and model extraction, which involves stealing the base architecture or parameters of an AI model to create a competing system[8].

Each of these attack vectors compromises the integrity of the security solutions while also jeopardizing the trust in these AI technologies that are increasingly relied upon to help protect sensitive information Furthermore, when SIEM systems become targets of AML attacks, they not only lose their effectiveness as security tools but also become a significant point of vulnerability within the

network. Instead of acting as a proactive defense against cyber threats, they can inadvertently provide attackers with deeper access, turning these critical monitoring systems into gateways for exploitation and further infiltration. AML attacks can severely compromise the accuracy of threat detection functionalities in SIEMs. It may result in threats being overseen, and increase false positives, which can in turn overwhelm security teams with alerts and delay their response to genuine threats. As enterprises are increasingly relying on AI models to analyze traffic and logs, the risk of AML attacks disrupting these processes becomes a bigger challenge for SIEMs, security systems, and AI-powered solutions at large[9].

Here are just a few instances where AML Attacks were undertaken in various ways to disrupt responses or deductions of AI/ML Models in general and to AI/ML models used in AI-powered security solutions.

Hugging Face Platform

Over 100 malicious AI/ML models were discovered on the Hugging Face platform. These models contained payloads that could grant attackers shell access to compromised machines, potentially leading to large-scale data breaches or corporate espionage. The models initiated reverse shell connections to specific Internet Protocol (IP) addresses when loaded.

A dozen critical vulnerabilities were found in various AI/ML tools. These are some included:

- CVE-2024-22476 (CVSS 10.0): A critical vulnerability in Intel Neural Compressor allowing remote attackers to escalate privileges.
- CVE-2024-3234: A critical issue in ChuanhuChatGPT enabling sensitive file theft.

- CVE-2024-3429: A path traversal vulnerability in LoLLMs leading to arbitrary file reading.
- CVE-2024-3584 and CVE-2024-3829: Critical vulnerabilities in Qdrant allowing arbitrary file writing and potential server takeover.
- CVE-2024-4146: A critical flaw in Lunary allowing unauthorized Application Programming Interface (API) access to projects.
- CVE-2024-24591: A path traversal vulnerability was discovered in Allegro AI's ClearML platform (versions 1.4.0 to 1.14.1)

Redis-py vulnerability in ChatGPT (March 2023)

This vulnerability (CVE-2023-28858, CVE-2023-28859) caused a data leak in OpenAI's ChatGPT service. The bug in the open-source Redis client library allowed some users to see titles from other users' chat history and exposed the personal data of about 1.2 million ChatGPT Plus subscribers[10], [11].

MLFlow

One of the most popular tools in an ML system is MLFlow (with over 13 million monthly downloads and increasing), which is used for managing the end-to-end machine learning lifecycle[12]. Recently it has been subject to multiple vulnerabilities, as shown below, CVEs:

- CVE-2023-6975: Arbitrary file written in MLFlow (CVSS 9.8)
- CVE-2023-6753: Arbitrary file written on Windows in MLFlow (CVSS 9.6)
- CVE-2023-6940: Server-side template injection bypass in MLFlow (CVSS 9.0)

- CVE-2023-6976: Arbitrary file upload patch bypass in MLFlow (CVSS 8.8)
- CVE-2023-6909: Local file inclusion in MLFlow (CVSS 7.5)
- CVE-2024-0520: A vulnerability in the mlflow.data module that could lead to remote code execution (RCE)

Cylance AI-Powered Endpoint Security Incident (2019)

This incident involved a vulnerability in Cylance's AI-based antivirus product, CylancePROTECT. The vulnerability allowed attackers to bypass the AI-based threat detection system by manipulating malicious files in a way that caused the AI model to misclassify them as benign. This attack was reported on July 18th, 2019, and there was a reported success rate of approximately 85% for tested malicious files. Cylance itself reported a 50% bypass creation success rate based on its internal testing. The gravity of this attack is that it allowed unsophisticated attackers to modify executables to evade detection without rewriting their malware. Luckily, Cylance released a patch on the 21st that essentially added anti-tampering controls and removed features that were not yet supported by the new controls [13].

Another case is a 2020 incident reported by McAfee Advanced Threat Research involving VirusTotal poisoning. There was a rapid influx of similar ransomware samples that were uploaded to the platform, potentially confusing and overwhelming detection systems.

As organizations increasingly rely on AI models for traffic and log analysis, there will be more incidents such as the ones described above and possibly more disruptive cases if not addressed promptly. Therefore, there is an urgent need to examine the true impact of AML on SIEM systems, educate the cybersecurity community on the nature of these dangers.

Purpose of the Study and Objectives

The purpose of this study is twofold. First, I aim to investigate the vulnerabilities posed by AML attacks and their implications for AI-powered security solutions, particularly SIEM systems. Second, I intend to explore the potential AML attack vectors to which these systems are susceptible, specifically examining how such attacks compromise threat detection accuracy, increase false positives, and disrupt response functions. While pursuing the asserted objectives, this study will attempt to answer the following research question: "How do adversarial machine learning attacks affect the threat detection accuracy of AI-powered SIEM systems?". This will be accomplished through a structured approach that begins with a comprehensive review of AML attacks and SIEM systems, focusing on their applications, structures, functionalities, and limitations. This will be supported by experimental testing to demonstrate the real-world outcomes of these attacks on AI-based security systems. Additionally, I will discuss current AML mitigation strategies, focusing on adversarial training, ensemble learning, and feature squeezing, to understand their applicability within SIEM environments.

Given the growing reliance on AI models for traffic and log analysis in cybersecurity, combined with the sophistication of modern threats and the potential for AML to undermine security systems, this study will aim to provide the cybersecurity community with insights into how these vulnerabilities manifest in practice. While the primary focus is on assessing attack impact, the study also highlights theoretical mitigation strategies such as adversarial training and ensemble approaches, to inform future research and design decisions for more resilient AI-based security architectures.

Overall, this study will pursue the following objectives:

- To simulate a realistic AI detection component of a SIEM using Feedforward Neural Network (FFNN) and Convolutional Neural Network (CNN) models trained on Canadian Institute for Cybersecurity Intrusion Detection System (CICIDS2017).
- To generate and evaluate adversarial evasion attacks using Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD) on these AI classifiers and assess detection degradation.
- To analyse and document empirical metrics that reveal weaknesses introduced by AML attacks in SIEM-like settings.

Scope of the study

As outlined in the section above, this study investigates the vulnerabilities posed by AML attacks on AI-driven SIEMs and assesses their effects on detection performance. The investigation focuses on a POC of several evasion attacks executed at the inference stage using white-box assumptions, where the attacker is assumed to have full knowledge of the model's architecture and parameters.

The scope is limited to:

- Dataset: Primarily the CICIDS2017 dataset, with references to University of New South Wales Network-Based (UNSW-NB15) and Network Security Laboratory–Knowledge Discovery Databases (NSL-KDD) for comparison and validation.
- Detection Focus: Only the AI component of the SIEM system is evaluated, excluding rule-based, heuristic, and human-analyst intervention layers which all exist in real-world deployments, but which would have drastically broadened our investigations and biased certain aspects of it.

- Attack Methods: The study applies an evasion attack technique using the FGSM to generate adversarial inputs at the inference stage.
- Environment: Experiments are conducted in a locally controlled and isolated environment using mostly open-source tools and frameworks.
- Mitigation: While theoretical defense techniques (e.g., adversarial training and ensemble methods) will be discussed in the literature review, however, they are not implemented or evaluated in the experimental phase of this study.

This focused scope ensures clarity and feasibility while still producing valuable insights into the specific vulnerabilities of AI-powered threat detection under adversarial conditions.

Significance of the study

This study highlights the significance of understanding vulnerabilities introduced by AML attacks on AI-powered SIEM systems, as the integration of AI into cybersecurity frameworks becomes more widespread. Addressing these risks is essential for ensuring the reliability and security of modern threat detection mechanisms.

Moreover, the study's contributions are relevant to the development of defensive mechanisms in SIEM systems against AML attacks, supporting organizations in transitioning from reactive to proactive cybersecurity measures. These proactive measures can help reduce the impact of cyber threats significantly. The findings will have broader implications for the cybersecurity community by informing best practices and guidelines for implementing AI-powered cybersecurity solutions.

CHAPTER 2

LITERATURE REVIEW

For a while now, there has been extensive literature on AML attacks, but this did not get significant attention from the general academic and research community until the last three years, when AI Image Recognition Systems (IRS) and Intelligent Conversational Systems (ICS) emerged with the popularization of ChatGPT and similar products. What I discovered with this literature review, however, is that AI has been integrated into systems at different levels for over 10 years now, but also that the risks were detected in the early stages, but were not considered as disruptive at the time, so not much attention was paid to studies related to these risks. AML, being one of the most prevalent, was initially discovered in research on neural networks and computer vision, where it was found that even small changes to input data could cause machine learning models to make significant misclassifications, as mentioned in a paper by Szegedy et al [14].

Since then, the concept has been extended to the cybersecurity domain, where malicious actors can use adversarial inputs to bypass ML-based intrusion detection and prevention systems, avoid malware detection, and exploit vulnerabilities in automated SIEM platforms. As Biggio et al highlighted in a 2013 paper, adversarial attacks on ML models used in SIEM systems represent a critical challenge that must be addressed to secure AI-powered environments [9].

Moreover, recent research advancements in federated learning and distributed AI architectures have introduced new dimensions to AML risks. These decentralized

models, while enhancing data privacy, create additional attack surfaces by exposing model updates to adversarial interference during communication between nodes. According to a study by Li et al., adversaries can manipulate the aggregation process in federated learning, injecting malicious updates that compromise the entire model[15]. This highlights a growing need to secure not only the models but also the communication channels and training pipelines involved in distributed AI environments, and this can only be done if there is sufficient research on the optimal approaches to achieve this. However, other findings suggest that federated models, while vulnerable, can also limit the scale of an attack due to their distributed nature, meaning a single compromised node does not necessarily corrupt the entire system immediately.

For years now, SIEM systems have been the foundation of enterprise cybersecurity, offering centralized visibility of network activity and generating alerts from combined logs and event data. It was only a matter of time before these systems became inefficient, with the use of technology increasing exponentially and cyber-attacks increasing at the same rate, if not even more. Several researchers, such as Dadashi (2014), predicted this early on, highlighting the challenge of information inefficiency and overload in dynamic control environments, which can result in suboptimal decision-making and operational inefficiencies[16]. Traditional SIEM platforms were initially rule-based and reactive, relying on static correlation rules and signature detection methods. However, with the increasing complexity of threats and the vast amount of data collected, the adoption of AI/ML technologies became imperative to effectively analyze this data within the required time. Research conducted by Ahmadian in 2023 shows that the successful integration of AI into SIEMs has led to a significant decrease in false positives, with a reported reduction of

up to 40%, while also significantly reducing the response time[17]. With all this positive feedback, AI integration introduces complexity, potentially increasing false negatives and detection failures if the models are not carefully monitored and regularly updated, and that is one of the areas I am raising awareness on.

AI itself has had quite a few different definitions over time, as research has expanded its scope and implementation capabilities. Russell and Norvig define it as the replication of human cognitive functions in machines, allowing systems to perform complex tasks, recognize patterns, and adapt autonomously to evolving situations within large datasets.[18]. However, no introduction regarding the foundation of AI would be complete without acknowledging Alan Turing and John McCarthy, whose pioneering work laid the theoretical groundwork for artificial intelligence and modern machine intelligence.

Over the decades, AI has evolved significantly, advancing its capabilities in various domains while navigating ethical considerations and regulatory challenges. One of those major leaps, as mentioned by McCarthy, is the introduction of transformer architectures, which marked a turning point in AI's evolution, providing unparalleled capabilities in processing sequential data[19]. With time, this innovation has significantly improved the ability of AI systems to analyze vast datasets, contributing to enhanced performance in anomaly detection and automated threat response in cybersecurity applications[19]. Transformers' self-attention mechanisms enable more accurate analysis of complex network traffic patterns, which as I will show later on in this chapter, make AI a key component of modern cybersecurity efforts.

I also understood AI Systems a bit better through the concept of AI maturity levels, which reflect the progression of AI from basic automation to advanced, self-

optimizing entities. Russell outlined AI maturity in five progressive stages, starting with simple reflex agents, which directly map inputs to predefined outputs but cannot handle unseen scenarios. Next are model-based reflex agents, which incorporate an understanding of their environment, enabling them to manage more complex interactions. Then goal-based agents follow, generating plans and adapting their actions to achieve specific objectives. The goal-based agents are succeeded by utility-based agents, which optimize actions using utility functions to balance competing objectives and maximize outcomes. At the pinnacle are learning agents, capable of generating novel instructions and adapting to scenarios not explicitly covered during training, continuously evolving through learning and adaptation[18]. This evolution shows AI's potential to exceed its initial programming, enhancing its ability to handle unanticipated challenges. However revolutionary, this also introduces the greater risk of unpredictable behaviors and the susceptibility of AI to adversarial manipulation. As AI matures, the likelihood of adversaries exploiting model weaknesses or strengths, for that matter, through adversarial inputs resulting in misclassification and erroneous decision-making in security-critical environments also increases.

Early developments in ML can be traced back to the 1940s when simple neural network models were first conceptualized[20]. These fundamental concepts grew into more complex techniques, such as the perceptron by Rosenblatt around 1958 and decision trees by Quinlan in 1986[20], [21]. These fundamentals laid the groundwork for machine learning and the ability to understand trends and patterns in large data sets, which is now crucial in network-based anomaly detection by recognizing odd traffic patterns in real-time. More advanced deep learning architectures eventually enabled complex malware analysis through automated feature extraction, while neural networks improved intrusion detection systems by learning from enormous volumes

of network logs and flow data[22]. As said by Mohamad et al., this ability to detect subtle patterns and deviations far exceeds the capabilities of traditional rule-based detection systems, establishing AI and ML as indispensable components of next-generation SIEM platforms[23].

Recent studies, such as that of Zhang et al., have expanded on this by incorporating GNNs (graph neural networks) to detect lateral movement in networks, highlighting how relational data structures can expose hidden attack paths[24]. GNNs use graph representations of network environments, which makes them very effective at detecting coordinated attacks across several nodes and devices.

Research shows that AML attacks exploit the weaknesses inherent in ML algorithms, targeting their reliance on data-driven learning. These attacks manifest in various forms, including evasion, poisoning, model extraction, and more. Huang et al. highlight the dual role of Generative Adversarial Networks (GANs) in cybersecurity—both as tools for generating adversarial examples and as defensive mechanisms for strengthening AI resilience[25]. GANs can produce highly realistic adversarial samples, which can bypass detection and infiltrate security systems undetected, posing significant challenges for AI-powered SIEM environments, however, I will discuss this more in detail later in this research work.

Furthermore, research on zero-day exploits and supply chain attacks has revealed that AML techniques can compromise the model development lifecycle itself. According to Nasr et al. (2021), by targeting data pipelines during training, adversaries can introduce subtle biases that only become apparent in production, affecting the reliability of SIEM alerts and incident responses[26]. In addition, researchers have investigated recent occurrences associated with AML, including the exploitation of AI-powered endpoint security tools such as the 2019

CylancePROTECT incident by researchers from Skylight Cyber. This demonstrated how adversaries manipulated malware to avoid detection by AI-based endpoint security tools, achieving a success rate of 85%[13]. I will show more examples similar to this in the following sections, showing how imperative it is to understand the risks of AML attacks and their impact on security systems.

The following sections of this chapter deal with the theoretical and practical dimensions of all these concepts introduced above, as well as their role in the overall risks associated with AML attacks. I will discuss evasion attacks, poisoning techniques, and model extraction methods, as well as their impact on SIEM platforms and the broader cybersecurity ecosystem; then I will discuss and evaluate the effectiveness of various defense mechanisms, such as adversarial training, ensemble learning, and feature squeezing, in mitigating adversarial threats. By combining ideas from old and current literature, this chapter seeks to provide a complete understanding of the complex relationship between AI, ML, SIEMs, and cybersecurity in general, emphasizing the importance of continuous research and innovation in protecting the cybersecurity landscape against malicious actors.

Identified Knowledge Gap

Throughout this literature, I observe significant progress in AI-powered SIEM systems, however not all critical issues have been addressed. Previous research does not fully exhaust how AML attacks specifically impact SIEM systems in realistic settings, focusing instead on isolated cases or theoretical vulnerabilities. Existing works often demonstrate the efficacy of AML attacks and defenses separately, but comprehensive evaluations within real-world SIEM environments are limited. Moreover, few studies provide practical guidelines for deploying robust AML defenses that balance computational overhead and security effectiveness.

Table 1

Comparison of Knowledge Gap

Existing AML Research on AI/ML models	Gap: AML specifically on SIEM
Focus on image and vision domains	Limited SIEM-specific studies
Theoretical analysis	Limited real-world evaluations
General cybersecurity context	Inadequate practical guidelines
	Insufficient AML impact metrics

Security Information and Event Management System

SIEM systems have become integral to modern cybersecurity frameworks, providing centralized solutions for collecting, analyzing, and responding to security events across an organization's IT infrastructure. This section explores the literature I gathered around SIEMs from its fundamentals to the specificities linked to AI, AML, and cybersecurity in general. A recent systematic review of SIEM technology by Lopez et al explains that SIEMs were initially developed as a fusion of Security Information Management (SIM) and Security Event Management (SEM) technologies, and were designed to aggregate log data, correlate events, and generate alerts for potential security incidents[27]. Early iterations relied heavily on static rule-based configurations and signature-based detection mechanisms, which limited their flexibility to emerging sophisticated attacks.

In their work, they explain that over time, as the volume of data increased and the complexity of cyber threats grew, traditional SIEMs faced challenges in managing vast amounts of data and detecting more subtle, non-signature-based attacks[27]. This limitation necessitated the integration of AI/ML into SIEM platforms, marking a significant shift from reactive to proactive cybersecurity measures.

Role of AI/ML in SIEM Evolution

AI/ML models are now crucial in enhancing SIEM capabilities, enabling the automation of threat detection, predictive analytics, and anomaly detection at scale, among other things.

Based on another systematic review done by Nassif et al., Machine learning algorithms, particularly unsupervised and supervised models, enable SIEMs to identify patterns in network traffic, detect anomalies, and predict potential threats based on historical data. By leveraging ML, SIEMs can autonomously adapt to evolving attack techniques, reducing false positives and enhancing the accuracy of threat detection[23].

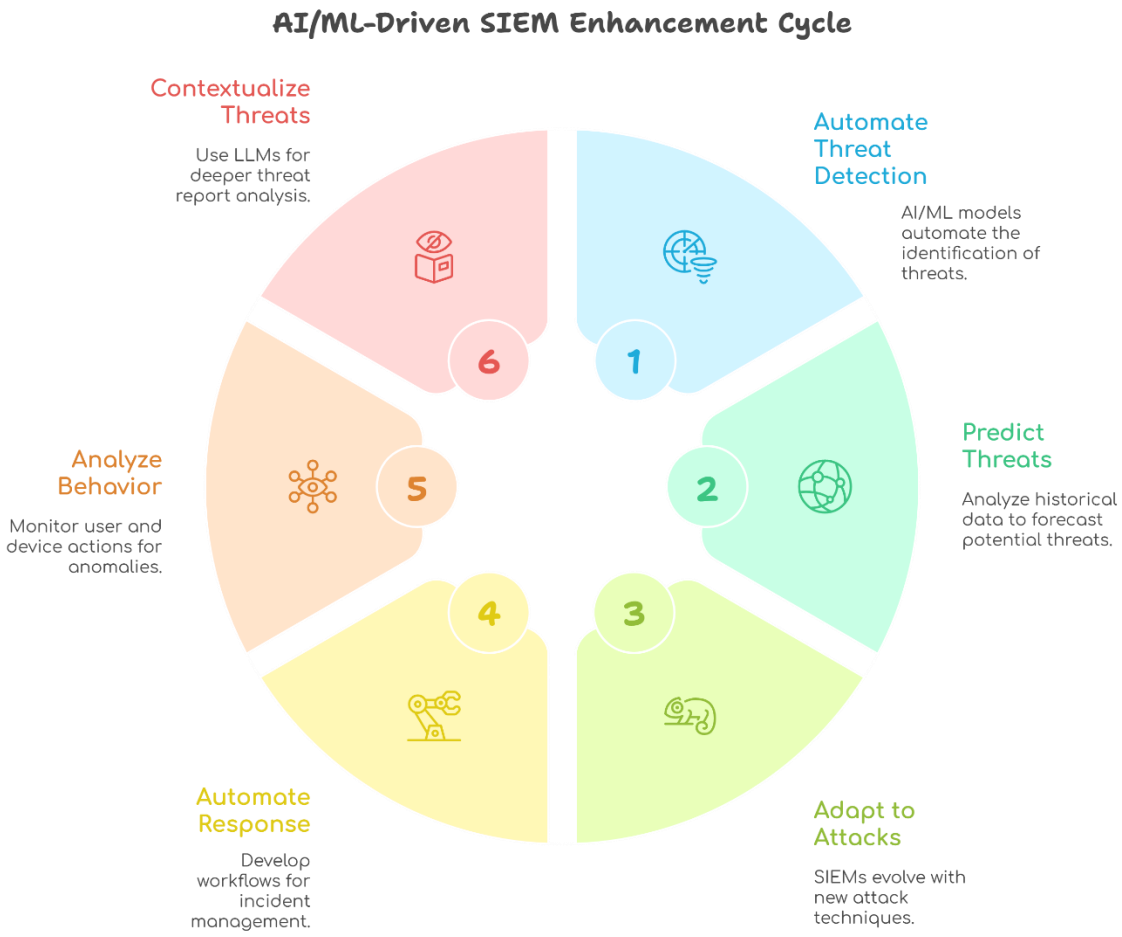
Additionally, Miloslavskaya mentions that AI-powered SIEM platforms facilitate the development of automated incident response workflows. These systems leverage Natural Language Processing (NLP) to analyze threat intelligence reports and correlate data from various sources, accelerating the triage and remediation process, and reducing the manual workload for security analysts, allowing organizations to respond to threats faster and with greater precision[28].

Another major component that AI/ML brings to SIEMs is User and Entity Behavior Analytics (UEBA). Gormez et al, in their paper, discuss this extensively, stating that UEBA monitors the behavior of users and devices, establishing baselines and identifying deviations that may indicate insider threats, compromised accounts, or lateral movement by malicious actors. This approach allows SIEM platforms to detect advanced persistent threats (APTs) that may evade traditional signature-based detection[29].

Recent advancements also show the integration of Large Language Models (LLMs) into popular SIEM platforms to improve the contextual analysis of threat reports, Microsoft Sentinel and Stellar Cyber being some of them[30].

Figure 2

SIEM Enhancement Cycle



Artificial Intelligence

As mentioned earlier, Artificial Intelligence at its core aims to replicate human cognitive functions in machines, enabling systems to perform complex tasks, recognize patterns, and adapt autonomously to evolving situations within large[18]. In

this section, let us go through the literature gathered on AI and explore how it relates to our topic. Back in the mid-20th century, pioneers such as Alan Turing and John McCarthy explored the feasibility of machines executing tasks traditionally requiring human intelligence. Turing's seminal paper, "Computing Machinery and Intelligence" in 1950, laid out the theoretical concept of simulating human thought processes in computational systems[31]. Early AI initiatives were largely symbolic, relying on rule-based systems often termed Good Old-Fashioned AI (GOFAI) that employed logic and manually encoded knowledge. While GOFAI performed well in tightly controlled environments, its limitations became evident when faced with the unpredictability of the real world. Russel et al add that the field experienced an uprising through ML, particularly in the 1980s and 1990s, as statistical models capable of learning from data and not relying only on hand-crafted rules came to the forefront.[18].

Szegedy et al. affirm the above, stating that the operational foundation of AI systems today lies in the development of machine learning models, which are constructed by feeding vast amounts of data into algorithms that iteratively adjust and improve through a process known as training[14]. This typically involves supervised learning, where labeled data guides the model, or unsupervised learning, where patterns are inferred without explicit labels[14], [32]. Lecun et al. add to this that neural networks are inspired by the structures of the human brain, forming the backbone of deep learning, which is a subset of ML that is good at handling unstructured data such as images, text, and audio. These networks, particularly FFNN models in our case, consist of layers of interconnected nodes that process input data sequentially, extracting hierarchical features through complex transformations. FFNNs are commonly used for structured data classification tasks such as intrusion

detection in SIEM systems, as they are relatively simple yet powerful enough to learn subtle attack patterns.[22]. Reinforcement learning further enhances AI capabilities by enabling models to make sequential decisions through trial and error, optimizing their strategies based on rewards and penalties[22].

Vaswani et al. talk about transformative advancements in AI, such as the development of transformer models. This innovative approach has driven breakthroughs in natural language processing (NLP), image recognition, and autonomous systems[33].

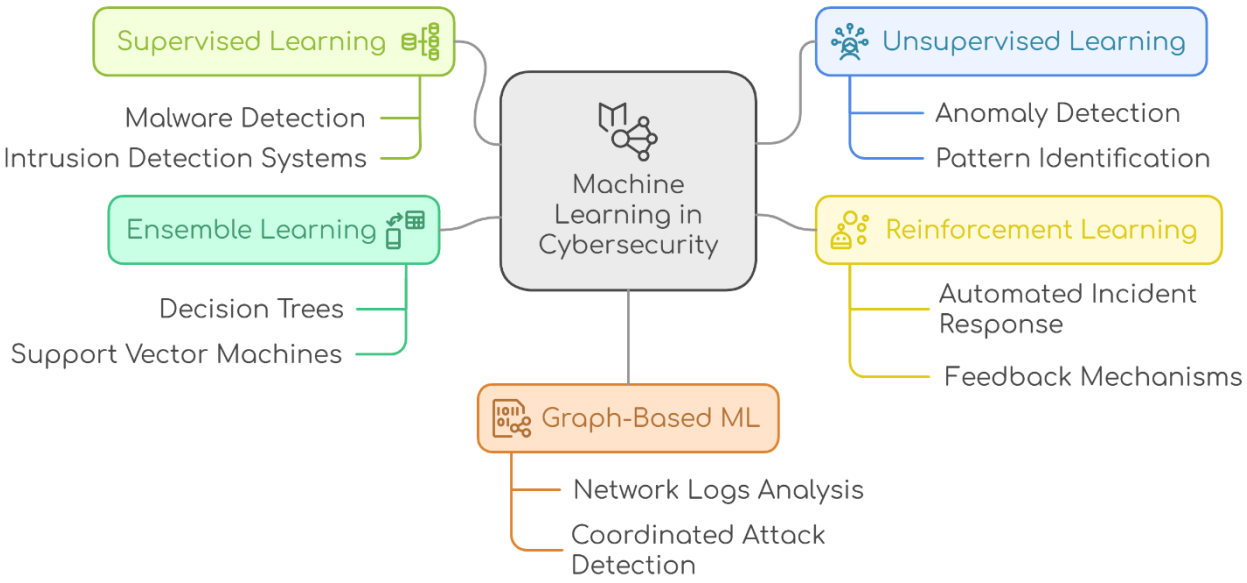
Machine Learning (ML)

Machine Learning is a subset of AI that focuses on developing algorithms that learn from data and improve over time without explicit programming[32]. The foundational principle of ML is that systems can autonomously identify patterns, make predictions, and refine their decision-making processes. Let us review some key concepts in ML through literature. Machine Learning techniques are categorized into five main types:

- Supervised Learning: Models trained using labeled data to recognize specific patterns (e.g., malware detection).
- Unsupervised Learning: These identify hidden patterns in data without labeled outcomes (e.g., anomaly detection).
- Reinforcement Learning: Models learn by interacting with their environment and receiving feedback (e.g., automated incident response).
- Ensemble Learning: Combines multiple models to improve performance and accuracy (e.g., cybersecurity threat classification).
- Graph-Based Machine Learning: Uses graph structures to model relationships between entities (e.g., fraud detection in networks).

Figure 3

Machine Learning in Cybersecurity



Ahmadian et al. detail how ML models improve the accuracy of intrusion detection systems (IDS) by learning from past security incidents, identifying deviations, and flagging potential threats[17]. This continuous learning mechanism allows SIEMs to adapt to new attack vectors promptly, significantly reducing the response time to zero-day attacks and other emerging threats.

Additionally, Turing et al discuss ensemble learning techniques, which combine the predictions of multiple ML models and may demonstrate superior accuracy and robustness against adversarial manipulation by integrating outputs from decision trees, support vector machines, and neural networks. Ensemble methods minimize the risk of overfitting and improve resilience to adversarial inputs[31].

Lastly, graph-based machine learning, which models data as nodes and edges, has become increasingly relevant in cybersecurity for detecting lateral movement within

networks. Xu et al. highlight that FFNNs can reveal hidden relationships in network logs, enhancing SIEM platforms' ability to detect coordinated attacks that would otherwise evade traditional linear models[24].

AI and the Rise of AML Attacks

Though briefly introduced earlier, this section delves deeper into the research on the relationship between AI systems and the rise of AML attacks. I have already established that AI systems, while transformative, are inherently susceptible to manipulation through adversarial techniques, exposing vulnerabilities tied to the way AI learns, generalizes, and makes decisions[9].

One notable aspect of AML is the ability to craft adversarial inputs that are indistinguishable from the human eye but significantly impact AI model performance. For instance, research by Eykholt et al. demonstrated that modifying just a few pixels in an image could lead to misclassification by state-of-the-art image recognition systems[34]. Now, a similar concept is translated into cybersecurity, where adversarial inputs can manipulate log data or network traffic to bypass SIEM defensive measures.

The root cause of AML attacks lies in the probabilistic nature of AI models. As mentioned in their paper, Lin et al address this probabilistic nature of AI models, saying that, unlike traditional software that adheres to rigid rules, AI systems generalize from data, leading to inherent uncertainties in their outputs. Attackers exploit these uncertainties by introducing adversarial examples (carefully crafted inputs designed to deceive models into making erroneous predictions) while appearing normal to human observers[35].

Another important advancement in AML research is the identification of model transferability. Adversarial examples crafted for one AI model often succeed

against others, a phenomenon explored in detail by Liu and his colleagues. This transferability reveals potential systemic vulnerabilities across AI-powered SIEM platforms, making it essential for organizations to adopt diverse model architectures and ensemble methods to mitigate this risk[36].

The dynamics of AML attacks can be seen in the iterative nature of AI training and inference. According to Goodfellow, AI models optimize their performance during training by minimizing loss functions, but this approach also produces non-linear areas in which minor changes to input data result in disproportionately large shifts in output. Adversaries identify and target these regions to create perturbations that push the model toward inaccurate outputs[14].

I also note that the large size and complexity of AI models make them more vulnerable to AML attacks. As neural networks become more complicated, their parameter space grows, giving adversaries more opportunity to alter the model. This intricacy, according to Huang et al, means that even small input adjustments may intersect with many vulnerable regions, increasing the risk of misclassification[25]. With this, let us now dive a bit deeper into the literature on AML and its fundamentals.

AML

AML has become an increasingly vital research area within the field of AI-powered cybersecurity. As AI and ML continue to take over various industries technologically, the vulnerabilities associated with these technologies will only expand as well. Let us look at several works that have been published about this. Biggio et al. identify three main types of adversarial attacks relevant to cybersecurity: evasion, poisoning, and model extraction. I will talk more about these attacks as I progress in this section.

Fundamentals of Adversarial Machine Learning

Adversarial attacks on AI systems leverage the non-linearity and high-dimensional nature of ML models to introduce errors. A paper titled “Wild Patterns” by Biggio et al talks in detail about AML techniques and how they are categorized based on the stage of the model lifecycle they target:

- **Evasion Attacks**: These attacks occur during the inference phase, where adversaries craft malicious inputs that evade detection by the AI model[9], [35]. This method is frequently used against IDS and malware classifiers, where even slight modifications to malware samples can bypass detection mechanisms.
- **Poisoning Attacks**: These take place during the training phase, with adversaries injecting malicious data into the training set to corrupt the learning processes [9], [15]. Poisoned models gradually lose accuracy, leading to more false positives or negatives and weakening the effectiveness of a SIEM platform's incident response process.
- **Model Extraction and Inversion**: In these attacks, adversaries query an AI model repeatedly to replicate its behavior and learn its decision boundaries[9]. This allows attackers to recreate the model externally and craft inputs that consistently evade detection.

These attack vectors demonstrate the broad applicability of AML across various domains, from endpoint protection to network intrusion detection, as they are used in different ways to replicate attacks and introduce malicious inputs.

AML in Cybersecurity

The impact of AML on cybersecurity cannot be overstated. With its stealthy nature, the consequences of AML attacks are extremely disruptive and hard to

remediate when executed effectively. Systems such as SIEM platforms that rely heavily on AI models to detect anomalous behavior and potential threats are prime targets for adversarial manipulation. AML has introduced new pathways for attackers to bypass automated defenses, rendering previously effective AI-powered security measures vulnerable.

A previously mentioned example, the 2019 CylancePROTECT evasion incident, serves as one of the earliest real-world indicators of AI security bypass through AML[37]. In addition, Ahmadian et al. wrote about the long-term risks of poisoning attacks in SIEM environments, where corrupted datasets result in reduced detection accuracy over time, ultimately compromising the entire incident response process[17]. Such attacks degrade the operational integrity of SIEM systems, leaving organizations blind to ongoing intrusions, and the worst part is that no warnings would be raised for such attacks. Recent industry threat intelligence also confirms the operational risks of AML attacks. According to Symantec's 2024 Internet Security Threat Report, adversarial manipulation tactics were present in approximately 30% of observed AI-targeted cyber intrusions, often impacting automated defense mechanisms such as email filters. Beyond academic demonstrations, evasion attacks have surfaced in other enterprise settings too. FireEye (2022) reported that adversarial inputs had successfully manipulated endpoint threat models by subtly altering known malware strings to resemble benign traffic, resulting in stealthy intrusions that went undetected for hours.

Known AML Attacks – Mechanisms, Operations, and Impact

In the work of Goodfellow et al, they discuss a bit more of these AML attacks, focusing on the mechanisms through which they occur and the impact they have on systems.

Evasion Attacks. Evasion attacks occur during the inference phase, where adversaries create adversarial inputs designed to closely mimic benign data, enabling them to bypass detection systems effectively. These inputs are carefully crafted to exploit vulnerabilities in AI models, tricking them into misclassifying malicious data as harmless. Several studies have successfully applied such adversarial methods in related IT domains, including areas loosely connected to cybersecurity such as spam filtering and malware analysis. These works demonstrate that even small perturbations can lead machine learning models to misclassify critical inputs, highlighting the broader relevance of these attack vectors across AI systems. For example, Papernot et al. showed that adversarial examples generated using FGSM and related techniques could mislead a deep neural network trained for malware classification on the DREBIN Android dataset, achieving misclassification rates as high as 85% with minimal input modifications[38]. Common techniques used to generate such inputs include the already introduced FGSM framework and an iterative alternative PGD framework, which introduce small, calculated perturbations to input data. These perturbations are often imperceptible to humans but significantly affect the model's decision-making process, as demonstrated in the work of Goodfellow et al[14].

Fast Gradient Sign Method. The FGSM is one of the simplest yet effective techniques for generating adversarial examples. The paper by Madry et al shows that FGSM leverages the gradients of a model’s loss function to craft perturbations that deceive the model during the inference phase. By altering the input data in the direction that maximizes the model's prediction error, FGSM demonstrates how minor, calculated changes can mislead machine learning models while maintaining the original appearance of the input[14], [39].

The process involves calculating the gradient of the loss function with respect to the input data and identifying the direction in which the model is most vulnerable. The adversarial perturbation is then generated by applying a scaled version of the gradient, controlled by a parameter ϵ. This parameter determines the magnitude of the changes to the input. Although the perturbed input might be visually or contextually similar to the original data, it still causes the model to produce incorrect predictions. For instance, in an image classification task, FGSM can alter pixel values minimally so that a model classifies an image of a dog as a cat, despite the image still appearing as a dog to human observers [14].

Projected Gradient Descent. The PGD method, described by Madry et al., builds upon FGSM by introducing an iterative approach to adversarial example generation. Unlike FGSM, which applies a single perturbation step, PGD refines the adversarial input over multiple iterations. Each step incrementally adjusts the input by calculating the gradient of the loss function and applying a small perturbation in the direction that maximizes the error [39].

A defining feature of PGD is its use of projection to maintain the perturbed input within a valid range or norm bound. After each iteration, the input is "projected" back to ensure it remains within a pre-specified constraint, such as valid pixel intensity ranges for images or acceptable data formats for other types of inputs [39]. This iterative refinement process allows PGD to create more effective adversarial examples that can bypass robust models with higher success rates. While PGD is computationally more expensive than FGSM due to its iterative nature, it is considered one of the strongest first-order adversarial attack methods.

Poisoning Attacks. Poisoning attacks target the training phase of machine learning models by injecting malicious or manipulated samples into the datasets used

for training. These attacks are particularly dangerous because they undermine the model at its foundation, introducing vulnerabilities that adversaries can exploit after deployment. Unlike evasion attacks, which occur during inference, poisoning attacks disrupt the learning process, resulting in models that behave unpredictably or fail to detect critical threats effectively[14], [40]. Poisoning attacks can be broadly categorized into two types: targeted poisoning and indiscriminate poisoning.

Targeted poisoning focuses on crafting adversarial samples that cause specific misclassifications. For example, an adversary might insert malicious samples into a training set to make the model classify certain types of attacks, such as malicious network traffic, as benign. On the other hand, indiscriminate poisoning aims to degrade the overall performance of the model, increasing both false positives and false negatives, which weakens the system's reliability[40].

The operational mechanism of poisoning attacks, as described in a paper by Steinhardt et al, typically follows these steps:

- Data Manipulation: Adversaries identify vulnerabilities in the data collection pipeline. For instance, if a SIEM system relies on publicly sourced data, attackers can tamper with the data before it is collected for training.
- Crafting Poisoned Samples: Malicious samples are generated to subtly distort the model's decision boundaries. These samples often mimic benign data but contain hidden patterns designed to mislead the model during training.
- Injection into the Training Dataset: Poisoned samples are introduced into the dataset either through direct access to the training pipeline or by exploiting insecure or unverified data sources.

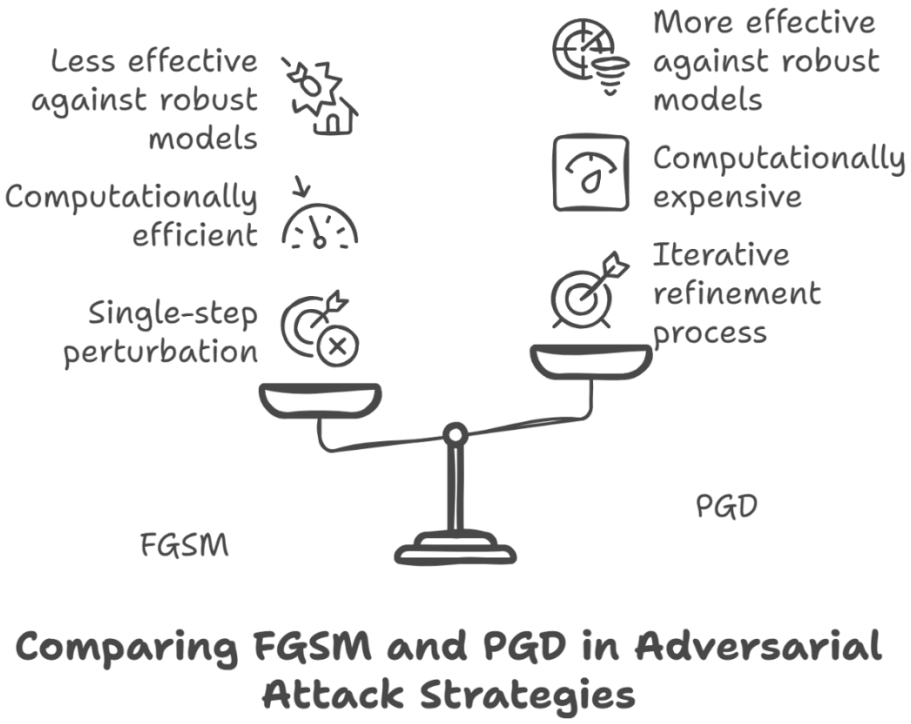
- Impact on the Model: During training, the poisoned samples alter the learned patterns of the model, embedding vulnerabilities that adversaries can exploit during inference. For example, a model trained on poisoned data may misclassify malicious activity as benign under specific conditions.
- Post-Training Exploitation: Once the compromised model is deployed, adversaries craft specific inputs that exploit the embedded vulnerabilities, bypassing detection or causing intentional misclassifications.

Poisoning attacks are especially relevant in cybersecurity, where AI-powered systems such as SIEM platforms rely heavily on diverse and continuously updated datasets. For instance, poisoning attacks can be used to manipulate network traffic data, causing a SIEM system to misidentify malicious activity as normal behavior. Or even phishing detection models can be compromised by injecting samples that mimic legitimate emails, reducing the system's accuracy in detecting phishing[40].

The exploitation of MLFlow vulnerabilities, such as CVE-2023-6975, demonstrated the potential for poisoning attacks to compromise machine learning pipelines, degrading long-term detection performance.

Figure 4

Comparing FGSM and PGD



Model Extraction and Inversion. Model extraction and inversion attacks represent a significant class of adversarial attacks that aim to compromise the intellectual property and privacy of AI models. While model extraction focuses on replicating the behavior of a model through extensive querying, model inversion seeks to extract sensitive information about the data used to train the model[9], [41]. These attacks are particularly dangerous in the context of cybersecurity systems that rely on machine learning for detecting and mitigating threats. Below, I now examine in greater detail the mechanisms of both these approaches to understand how they operate and what their impact on the security systems is.

Model Extraction Attacks. Model extraction attacks involve an adversary querying an AI model extensively in an attempt to replicate its behavior externally by creating a copy of it. According to Tramèr et al., this process involves submitting a

wide range of inputs to the target model and collecting the corresponding outputs, which are then used to train a surrogate model.[41]. The goal here is to create a model that behaves similarly to the original target model, which then would allow the adversary to generate adversarial inputs that can bypass the model's detection capabilities by testing them on the surrogate model.

The typical steps of a model extraction attack include:

- Query Generation: The adversary starts by submitting a series of queries to the target model. These queries may include random, benign, adversarial examples, and any other type of input that they deem relevant. The aim here is to obtain diverse responses from the model that provide insight into its decision-making process.
- Output Collection: For each query, the adversary collects the model's output, such as class probabilities, logits, or other predictive data. These outputs are valuable for reconstructing the decision boundaries of the targeted model.
- Training the Surrogate Model: The adversary uses the collected input-output pairs to train a surrogate model that mimics the behavior of the target model. The surrogate model can be used to generate adversarial examples or to gain insights into the functioning of the target model.
- Exploitation of the Surrogate Model: Once trained, the adversary will now use the surrogate model to craft adversarial inputs that are designed to bypass the original model's detection, and that is how they are able to evade the AI system or extract unintended data from the AI System.

Model extraction attacks are especially dangerous because they allow adversaries to duplicate a model's functionality without direct access to the model's

parameters or architecture. This potentially leads to the creation of more effective adversarial inputs that can exploit the vulnerabilities of the original model.

Model Inversion Attacks. Model inversion attacks go a step further by exploiting the outputs of a model to infer sensitive information about the data used to train it. Fredrikson et al. highlight that these attacks use the model's predictions, such as class labels and probabilities, to reconstruct sensitive features of the training data[42]. For example, in a model trained to recognize faces, an adversary might be able to reconstruct images of faces or infer private attributes, such as health conditions or demographic information.

Model inversion is typically carried out in the following manner:

- Analyzing Model Outputs: The adversary starts by observing the outputs of the model, including predictions and confidence scores, in response to various inputs. Usually, these outputs will show patterns that can be correlated with the training data.
- Optimization Algorithms: The adversary then employs optimization techniques to reverse-engineer the data used to train the model. For example, they might use gradient-based methods to approximate the original inputs that resulted in the observed outputs.
- Targeting High-Confidence Predictions: Once the data is reverse-engineered, the adversary inspects the inputs based on the data, and they will usually focus on inputs that produce high-confidence predictions to guarantee more accurate extraction of information about the training data.
- Reconstructing Data: Based on the model's outputs, the adversary will then reconstruct features of the original training data. In some cases, this can include the exact data points, while in others, it may involve inferring

sensitive attributes like medical conditions, financial data, or personal identifiers.

Through model extraction, adversaries can replicate an intrusion detection model and craft advanced attack patterns to evade detection, and through model inversion, they can reveal sensitive information about network traffic, user behavior, or other private data, jeopardizing an organization's security. A good example of these is the incidents in 2024, where compromised models uploaded to Hugging Face contained hidden payloads for reverse shell attacks. This is the level of impact that model manipulation and extraction can inflict on AI Systems.

Mitigation Mechanisms Against AML

While AML presents significant challenges, researchers have proposed numerous defense mechanisms aimed at mitigating its impact. According to a 2023 MITRE AI Security Consortium white paper, adversarial training combined with anomaly-aware feature embeddings is becoming a standard defense approach in sectors dealing with high-sensitivity data, such as finance and critical infrastructure. In this section of our literature review, I will be looking at some of the mitigative approaches that have been proposed.

Adversarial Training. Adversarial training is a prominent defense mechanism aimed at improving the robustness of machine learning models against AML attacks (specifically invasion attacks). According to Zhou et al, this technique involves retraining AI models using adversarial examples. By exposing the model to such examples during the training phase, adversarial training equips the model with the ability to identify and resist adversarial inputs, enhancing its resilience [41], [42].

Fredrikson et al add to the above, stating that the primary objective of adversarial training is to minimize a model's susceptibility to adversarial perturbations

by familiarizing it with potential attack patterns. This approach strengthens the model's generalization capabilities, ensuring more reliable performance in the presence of adversarial threats. Adversarial training follows a structured process that I explain through the key steps below:

- Adversarial Example Generation: Adversarial examples are crafted using techniques such as the FGSM or PGD. These techniques introduce small perturbations to inputs, exploiting the model's vulnerabilities[14]. In cybersecurity contexts, adversarial examples might be subtle alterations to network traffic logs or system events to simulate attack patterns.
- Integration into Training Data: The generated adversarial examples are added to the original training dataset. This ensures that the model encounters these inputs during training, increasing its familiarity with adversarial patterns.
- Model Optimization: The model is then trained to minimize loss not only on benign data but also on adversarial examples. This dual optimization forces the model to adjust its decision boundaries, making it more robust to adversarial perturbations.
- Iterative Refinement: Adversarial training is conducted iteratively, with adversarial examples regenerated as the model improves. This iterative process ensures the model remains adaptive to a broad spectrum of adversarial scenarios[14], [39].

In SIEM platforms, adversarial training can enhance the resilience of models analyzing network traffic, detecting anomalies, identifying malicious activities, advanced phishing emails, and complex intrusion tactics. Training models with

adversarial examples that mimic real-world attack patterns strengthens their ability to recognize and counter sophisticated threats.

While adversarial training shows quite some potential, it also has several limitations:

- Computational Overhead: Generating adversarial examples and retraining models require substantial computational resources, leading to increased training time and infrastructure costs.
- Limited Generalization: Adversarial training improves resilience against known attack types but may struggle to generalize to unseen adversarial methods.
- Accuracy Trade-Offs: A focus on adversarial robustness can sometimes result in reduced accuracy on benign data, impacting the overall performance of the model.

Ensemble Learning. Ensemble learning is another widely used defense mechanism in AML. The way it works is by combining predictions from multiple models to enhance robustness and mitigate adversarial attacks. By aggregating outputs from diverse models, ensemble learning reduces the likelihood of a single model's vulnerability being exploited. This approach leverages model diversity, ensuring that adversarial examples designed to deceive one model are less effective against the ensemble [39]. The collective decision-making process not only increases resilience but also enhances the overall accuracy and reliability of the system.

The principle behind ensemble learning is straightforward: instead of relying on a single model, multiple models are trained independently with variations in their architectures, datasets, or hyperparameters. This diversity ensures that the models do not share identical weaknesses, making the ensemble more robust to adversarial

perturbations[9], [39]. Below, I go through an overview of how ensemble learning is carried out.

The ensemble comprises multiple models with different configurations, training datasets, or objectives. This diversity minimizes the chances of a single adversarial input deceiving all models in the ensemble. For instance, in a SIEM system, models focus on analyzing different aspects of data, such as each model taking either network traffic, log files, or user activity patterns, but not all at once. Then the predictions from all models in the ensemble are combined using methods such as majority voting, simple averaging, or weighted averaging. All this is to ensure that the final decision reflects a consensus among models, reducing the impact of any single model's error.

Redundancy is the key feature of ensemble learning. Even if one model is compromised by an adversarial attack, the overall ensemble is less likely to be affected, as other models may still detect and flag the threat. Ensembles can then be updated iteratively to include new models or replace outdated ones. This adaptability ensures that the ensemble remains robust against evolving adversarial strategies as they are developed or discovered. However, as with Adversarial Training, Ensemble Learning is not bulletproof and has certain areas where it comes short.

- Managing and aggregating predictions from multiple models requires more computational resources during both training and inference which depending on scale can quickly become costly.
- Designing and maintaining an effective ensemble system involves careful selection of model diversity, aggregation methods, and weighting strategies; all of which add to the complexity.

- In basic terms, ensemble learning does not scale well. Adding too many models to the ensemble may result in minimal performance improvements while significantly increasing resource consumption.

Feature Squeezing. Feature squeezing is a defense mechanism designed to counter evasion attacks by reducing the complexity of input data. This technique compresses input features to eliminate subtle perturbations introduced by adversarial attacks, effectively making the modifications less impactful. By simplifying the input data, feature squeezing minimizes the differences between adversarial and benign inputs, thereby improving the model's robustness to such attacks[43].

Feature squeezing removes unnecessary or redundant details in the input that adversarial perturbations often exploit, hence reducing the attack surface available to adversaries while maintaining sufficient information for the model to make accurate predictions.

For instance, in image-based models, pixel values can be quantized to a lower bit depth, removing the fine-grained changes introduced by adversarial attacks, or in SIEM systems, log data records can be normalized or discretized to limit the impact of adversarial modifications while still keeping the input usable by the model. Some techniques used to squeeze features are bit depth reduction, as mentioned above, spatial smoothing, where I apply filters such as Gaussian blurring or median filtering, and dimensionality reduction, where the size or resolution of the input is reduced so that I focus on essential features only[43]. By comparing the model’s predictions on the original and squeezed inputs, discrepancies can show the presence of adversarial perturbations, which means that it can also be used to detect adversarial attacks.

Let us now talk of the limitation of feature squeezing:

- It is important to note that simplifying input data can also lead to a loss of important details, potentially reducing the accuracy of the model on benign inputs.
- Feature squeezing is more effective with small, subtle perturbations. More sophisticated adversarial techniques that introduce larger or more complex modifications may still bypass its defense.
- Feature squeezing also varies depending on the domain. Meaning, techniques that work well for image data may not be as effective for textual or tabular data.

Adversarial defenses like adversarial training, ensemble learning, and feature squeezing enhance SIEM platforms by improving resilience against attacks. However, though effective, these methods, as I have seen, face challenges like computational overhead and domain-specific limitations and require ongoing research to optimize their impact.

Conclusion

The integration of Artificial Intelligence and Machine Learning into SIEM systems has undoubtedly transformed the cybersecurity landscape, enabling more efficient threat detection and response. However, as highlighted in the literature, the growing reliance on AI and ML systems for cybersecurity has also introduced new vulnerabilities, particularly through AML attacks. While the current mitigation strategies are effective at their core, they fall short in terms of scale, diversity, and optimization.

Based on this literature review, I concluded that several topics require further research. Here are those that I highlighted:

- Limited Exploration of AML Attacks on SIEM Systems:While there is significant research on AML in general, there is a lack of targeted research on how these attacks affect AI-powered SIEM systems specifically. Existing studies primarily focus on AML in domains such as computer vision and image classification, leaving a gap in understanding how these attacks can compromise the efficacy of security systems, particularly in complex environments such as those used in cybersecurity.
- Insufficient Examination of AI-powered SIEM Vulnerabilities: The literature points to the vulnerabilities of AI-powered models, but there is insufficient detail in the examination of how these vulnerabilities manifest in SIEM systems, particularly in relation to advanced cyber threats. The industry's fast adoption of SIEM solutions, combined with its reliance on AI for proactive security measures, requires a more in-depth examination of how these systems might be exploited through adversarial attacks.
- Lack of Comprehensive Studies on Mitigation Strategies for AML in SIEMs: While several techniques for mitigating AML, such as adversarial training and ensemble learning, have been proposed, there is a lack of comprehensive studies that assess the effectiveness of these strategies within the context of cybersecurity systems. Furthermore, there is limited practical application of these defenses in real-world SIEM environments.
- The Need for Empirical Evidence on the Impact of AML Attacks on Threat Detection Accuracy: Much of the existing literature discusses AML attacks conceptually or theoretically, with limited empirical evidence on the tangible impact of these attacks on AI-powered cybersecurity tools like SIEM systems. The effects of AML on threat detection accuracy, the rate

of false positives, and the time taken to detect and respond to threats require further research.

- Lack of Integration Between AI-powered Security and AML Defence Mechanisms: Although there is research on AI-powered security systems and AML defense mechanisms, there is a gap in studies that explore how these fields can be integrated for a cohesive defense strategy. There is a need for a unified approach that both improves the security of AI models and mitigates the risks of AML attacks.

This thesis seeks to contribute toward addressing some of these gaps in different capacities. I aim to shed more light on AML attacks vis-à-vis SIEM systems and their impact on the latter, especially on how they degrade threat detection and response functions. I will evaluate various attack vectors based on real-world case studies in order for us to provide data-driven insights into the practical consequences of AML on AI-based security systems.

CHAPTER 3

METHODOLOGY

This chapter outlines the methodologies that are used to achieve the objective outlined in the first chapter. The methodologies used are subdivided into two distinct phases: an exploratory phase that establishes the theoretical foundations, identifies research gaps, and highlights the relevance of AML attacks within the context of SIEM systems; and an experimental phase that involves setting up a controlled environment simulating a modern, AI-powered SIEM system and subjecting it to AML attacks. This will involve setting up a SIEM environment, generating realistic adversarial attack scenarios, and applying appropriate evaluation methods that will help us measure the results effectively. To ensure a safe and ethical testing process, all experiments will be conducted in an isolated environment, with no interaction with live production systems or unauthorized networks.

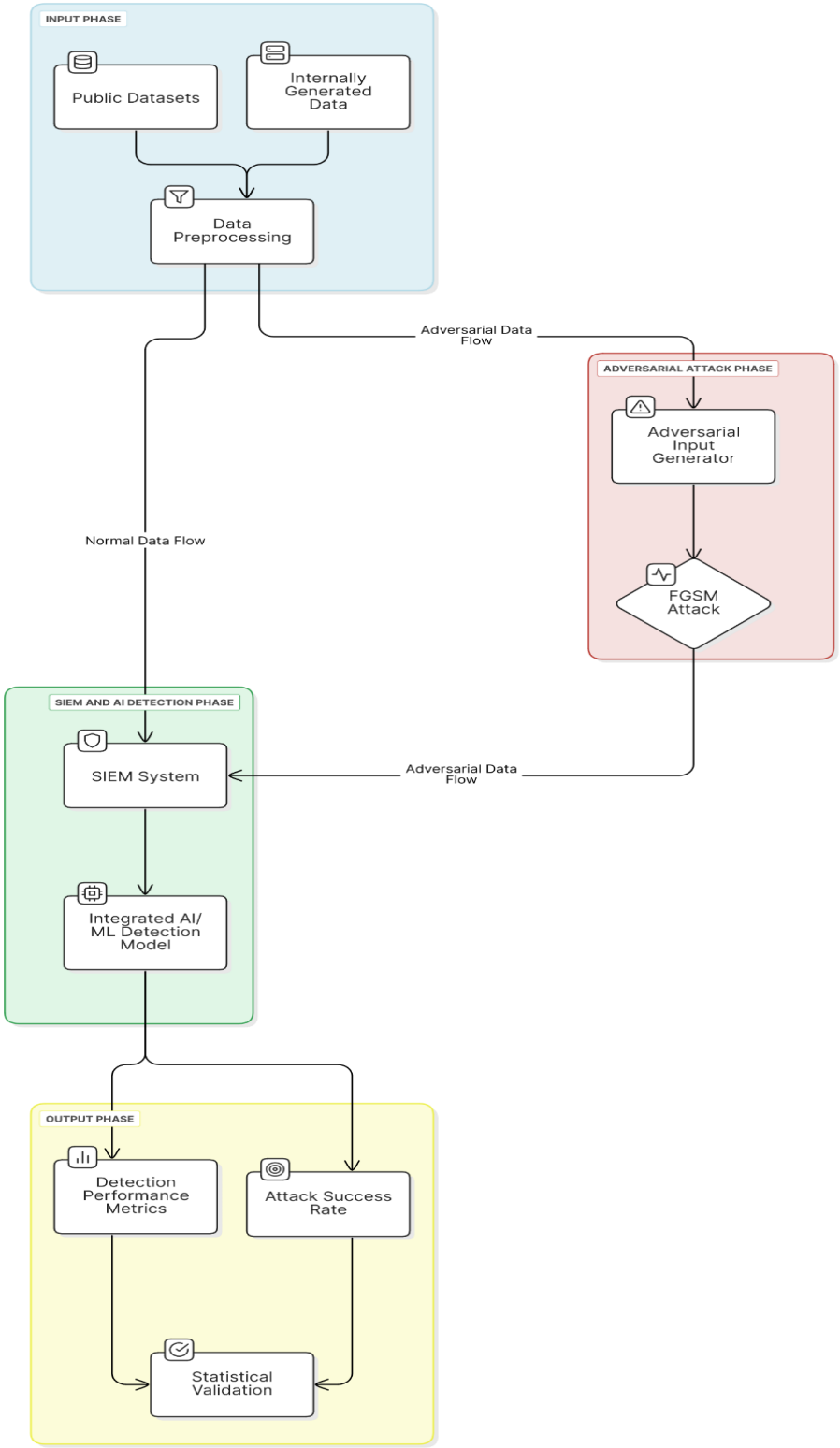
The subsequent sections of this chapter detail the overall research design, the tools and technologies used, how data is collected and prepared, the attack techniques employed, the experimental procedure, and the evaluation metrics. Throughout this chapter, I hope to provide a clear understanding of how this study was structured and how each methodological component contributes to a focused investigation into the threat of AML attacks on AI-powered SIEM systems.

Conceptual Framework

As established in earlier sections, this study aims to understand how the insertion of adversarial examples influences the AI components of SIEM systems' ability to detect and respond to threats. To support this, the study employs methods that allow for both theoretical grounding and practical observation.

Figure 5

Conceptual Framework of the Experiment.



A foundational literature review helped establish context, drawing from previous works to inform the current gaps in our research area and also the direction of our testing. Building on this foundation, the practical phase of the study relies on a controlled experimental environment that will enable us to test the system's behaviors under AML attacks. This approach will provide direct interaction with a simulated SIEM environment, enabling outcomes to be observed and measured under consistent conditions. It will also offer the structure required to analyze targeted outcomes such as disruptions in detection accuracy or changes in alert patterns without external interference.

Overall, the conceptual framework shows that this experiment will adopt an attacker-centric perspective, modelling the SIEM's AI detection component as the target and constructing evasion attacks using gradient-based AML techniques to simulate realistic breach attempts.

By approaching the problem from both conceptual and empirical angles, these methodologies ensure that the study not only explores underlying principles but also tests them under conditions that closely reflect real-world operational concerns.

Tools and Technologies

To carry out the experimental phase of this study, a well-structured set of tools and technologies is used, each selected based on its role in supporting the design, implementation, and evaluation of the experiments. The selected stack ensures that the environment reflects the behaviour of a real-world deployment while remaining suitable for controlled experimentation and reproduction.

SIEM Simulation Platform

The Elastic Security solution (part of Elastic Stack) will be used as the main simulation platform to replicate the functions of a modern SIEM system with built-in AI/ML integration. This open-source unified threat intelligence platform is chosen for its modular architecture and built-in support for log ingestion, event correlation, and anomaly detection. Within Elastic Stack, Elasticsearch handles data storage and retrieval, while Kibana provides dashboards and visual outputs for monitoring system activity. The Elastic Agent and Beats will be deployed for data collection from a simulated endpoint, enabling a realistic and continuous flow of system and network events into the environment. Elastic Security's integrated machine learning capabilities will be activated to simulate AI-powered detection logic, aligning the setup closely with real-world AI-powered SIEM operations.

The Elastic environment will be hosted on a virtualized testbed built on XCP-ng, an open-source hypervisor. Two virtual machines will be configured:

- VM1 (Elastic Stack host): 4 vCPUs, 8 GB RAM, 100 GB storage
- VM2 (simulated endpoint client): 4 vCPUs, 4 GB RAM, 50 GB storage

The above setup, thus far, provides an isolated and manageable environment for running our adversarial experiments.

Operating System

All VMs will run on Debian 12 (Bookworm), which is selected for its stability, long-term support, and compatibility with the tools will be used throughout the experiment as its reliability will ensure consistent performance during the testing process.

Machine Learning Framework

Predictive models will be developed using TensorFlow/Keras, a widely adopted deep learning framework supporting efficient model design, gradient computation, and adversarial testing. Two architectures will be implemented: a FFNN and a CNN. The FFNN, selected for its simplicity and interpretability on tabular data, will serve as the primary baseline. The CNN, leveraging 1D convolutions, will be used to assess whether spatial feature modeling improves adversarial resilience. Both models will be subjected to adversarial attacks using FGSM and PGD, implemented through the Adversarial Robustness Toolbox (ART). Due to local computational constraints, training will be conducted on a representative subset of CICIDS2017, optimized for balance, coverage, and feasibility.

Adversarial Attack Toolkit

To simulate AML attack scenarios, the IBM Adversarial Robustness Toolbox (ART) will be used. ART is a Python-based library that provides a wide range of adversarial attack algorithms including those used in this experiment. These algorithms are usually used to generate adversarial modifications on input data, with configurable parameters for attack strength and iterations.

The selection of FGSM and PGD attacks is based on their prominence in AML research and their complementary characteristics[25], [34]. FGSM, being a fast, single-step attack, provides efficient and computationally lightweight perturbations, suitable for large-scale evaluations. PGD, on the other hand being an iterative and stronger variant, enables deeper probing of model vulnerabilities under sustained adversarial pressure. Together, they will offer a balanced attack spectrum that covers both rapid and sophisticated threat scenarios, ensuring the findings are

methodologically rigorous, computationally feasible, widely comparable, and reproducible.

Data Handling and Preprocessing Tools

Data preparation will be handled using Python-based tools—Pandas and NumPy, which both allow efficient transformation, normalization, and feature selection across large datasets. These tools support the full preprocessing pipeline, from initial data loading to the creation of structured input for training and testing the FFNN model.

Evaluation and Statistical Analysis Libraries

To analyze the results and measure model performance I will use scikit-learn a tool used for computing evaluation metrics (such as accuracy, precision, recall, and confusion matrices). For deeper statistical insights SciPy would also be employed to run t-tests that will help determine the significance of observed differences across test conditions, however only one of the two will be employed for our research purposes.

Additional Utilities

- PostgreSQL will be used to log attack metadata and performance outcomes where needed for reference and reproducibility.
- Eland (a Python client for Elasticsearch) will be used to streamline interaction between Pandas and Elasticsearch for data querying and exploration.
- Nmap and Hydra will also be included here for simulating certain network reconnaissance and brute-force behaviors during the internal data generation.

- Python is the core scripting language across all components above, providing integration between data handling, model training, attack generation, and result analysis. Python is chosen for its high application in data analysis, Machine learning and other computing areas and overall, because of its scripting capabilities.

It is also important to note that all tools mentioned above are integrated into a unified experimentation environment, built entirely on open-source technologies. This ensures that the setup is replicable, extensible, and transparent, supporting future research and validation efforts.

Data Collection and Preparation

Throughout the experimentations I will use a combination of public intrusion detection datasets and internally generated data to train, evaluate, and test the behavior of the AI-powered SIEM system under relatively accurate adversarial conditions. The primary aim of data collection and preparation is to ensure that the machine learning model is exposed to realistic security events and is evaluated on data that simulates real-world operational contexts.

Dataset Selection

The CICIDS2017 (Canadian Institute for Cybersecurity Intrusion Detection System 2017) dataset is selected as the main data source due to its relevance to our experiments and richness in terms of attack types and log data. According to the Canadian Institute for Cybersecurity, it reflects realistic enterprise network traffic and includes a diverse set of attack types such as denial of service, brute force, infiltration, botnets, and web-based attacks; captured over several days[44]. The dataset provides labeled network flow records with over 80 features per instance, covering aspects

such as byte counts, durations, packet sizes, and protocol details. Its comprehensive structure and modern attack representation make it well-suited for training and testing an AI-based intrusion detection model.

Table 2

CICIDS2017 Dataset Properties

Attack Category	Specific Attack Types	Frequency
Benign		2,359,087
DoS	Denial of Service (DoS) Hulk, DoS GoldenEye, DoS Slowloris, DoS Slowhttptest, Heartbleed	252,671
DDoS	Distributed Denial of Service (DDoS)	41,835
Brute Force	FTP-Patator, SSH-Patator	13,835
Web Attack	Web Attack-Brute Force, Web Attack-XSS, Web Attack-SQL Injection	2,180
Infiltration	Infiltration	36
Bot	Bot	1,966
PortScan	PortScan	158,930
Total		**2,830,540**

Supplementary Datasets

To complement the CICIDS2017 dataset and provide additional context, two other benchmark datasets are also suggested and may be reviewed or selectively incorporated in our experiments:

- NSL-KDD (Network Security Laboratory–Knowledge Discovery in Databases): A refined version of the classic KDD'99 dataset. While older and simpler in structure, it offers a widely understood reference point with a well-documented feature space.

Table 3

NSL-KDD Dataset Properties

Attack Category	Specific Attack Types	Frequency
DoS	back, land, neptune, pod, smurf, teardrop, apache2, udpstorm, processtable, worm	45,927
Probe	satan, ipsweep, nmap, portsweep, mscan, saint	11,656
R2L	guess_passwd, ftp_write, imap, phf, multihop, warezmaster, warezclient, spy, xlock, xsnoop, snmpguess, snmpgetattack, httptunnel, sendmail, named	995
U2R	buffer_overflow, loadmodule, rootkit, perl, sqlattack, xterm, ps	52
Normal		67,343
Total		**125,973**

- UNSW-NB15 : A hybrid dataset combining real and synthetic traffic, offering more recent attack types and content-based features. This adds a middle ground between the older NSL-KDD and the more complex CICIDS2017, contributing to a broader validation of results across varying levels of dataset complexity.

Table 4

UNSW-NB15 Dataset Properties

Attack Category	Frequency
Benign	93,000
Fuzzers	24,246
Analysis	2,677
Backdoors	2,329
DoS	16,353
Exploits	44,525
Generic	58,871
Reconnaissance	13,987
Shellcode	1,511
Worms	174
Total	**257,673**

Internal Data Generation

To enhance alignment with the Elastic Security environment used in the experiments, a small internal dataset will be created. Controlled attacks such as port scans (via Nmap) and brute-force login attempts (via Hydra) will be launched against a virtual test machine, generating system logs and alert events. Alongside the traffic of the above controlled attacks, routine benign activities such as logins, file access, and standard network traffic will also be recorded. Though limited in size, this internal data is meant to reflect the behavior of an actual setup in a production environment and to help contextualize the outcomes of adversarial interference within a realistic SIEM environment.

System Architecture, Setup, and Adversarial Attack Methodology

This experimental architecture is designed to simulate realistic security event flows, execute adversarial attacks, and monitor detection outcomes, all within a framework that mirrors operational AI-powered SIEM behavior while offering control over the test conditions.

System Overview

The environment is composed of five core operational modules:

- Data Source: Loads a portion of the CICIDS2017 dataset cleans it, normalizes it, labels it, and sequentially feeds both benign and malicious records into the pipeline.
- Ingestion and Processing Pipeline: Receives the dataset records, extracts relevant features, and converts them into the correct format expected by the neural network model.

- AI Analytics Module: Runs the processed data through trained neural network models namely FFNN and CNN, to classify each event as either benign or malicious.
- Adversarial Attack Engine: Modifies selected malicious samples using adversarial techniques before passing them to the AI model for prediction.
- Monitoring and Logging Component: Collects the AI model's prediction results, compares them with the true labels, and stores the performance metrics (e.g., accuracy, false positives) for later evaluation.

These modules will communicate through in-memory data transfers for the most part, allowing near real-time interaction and fine control over experiment parameters.

While the FFNN and CNN classifiers are conceptually part of the AI-powered SIEM pipeline, they are implemented as standalone Python services. This architectural choice will allow us to have greater modularity, precise control over inputs and outputs, and also easier execution of adversarial attacks (as they will target the AI models directly). Hence, enhancing experimental clarity and avoiding potential interference from other SIEM subcomponents.

Environment Configuration

The Elastic Stack (v8. x) will be deployed with Elastic Security features serving as the SIEM interface. Its native machine learning plugins will be activated to enable detection capabilities. Logs will be ingested using Elastic Agent and Beats, and alerts will be monitored through Kibana. A local endpoint client will also be used to feed logs into the system, allowing control over both benign and malicious activity. Thus far, this reflects the core structure of a basic production SIEM system.

AI Analytics Module

The AI component consists of the previously introduced FFNN and CNN classifiers, trained on labeled security event data. These will be implemented as Python services, which receive normalized feature vectors and output predictions indicating whether events are malicious or benign, along with a confidence scores. This mirrors the decision-making layer in an operational AI-SIEM and provides the core target for our adversarial manipulation.

Adversarial Attack Engine

An adversarial attack component will be integrated between the ingestion pipeline and the AI module to simulate evasion attacks, where input data is subtly altered to bypass detection during designated testing phases.

The focus will be on evasion attacks applied during the inference phase, specifically targeting the classifier's ability to correctly detect malicious events. Broader attack vectors discussed in this study are only conceptually addressed but not implemented in this experimental setup. The attack methods used by this engine are described in detail in the following (section 3.5).

It is important to note that the attack engine will only be activated during predefined test phases where adversarial behavior is being evaluated. When enabled, it intercepts malicious input samples, applies adversarial perturbations, and records the original input, the modified (adversarial) version, the model's prediction on the modified input, and the attack's configuration parameters (e.g., epsilon value).

Monitoring, Control, and Evaluation

The monitoring component will log all experimental outcomes, including predicted labels, model confidence scores, ground truth values, and adversarial

metadata. These logs will be used to calculate performance metrics such as detection accuracy, false negative rates (FNR), and adversarial success rates.

A lightweight control script will orchestrate the different phases of the experiment by initiating and stopping event streams, enabling or disabling the adversarial attack engine, and resetting the system state between test scenarios. When this script is not being used, some of the phases will be initiated manually.

If errors or interruptions occur, the environment will be reset to a clean state using captured VM snapshots. Error logs and failed prediction cases will be recorded separately to assist with troubleshooting and identifying recurring issues however will not be included into the results as they would not have been influenced by the structured instructions that guide the experiment.

The evaluation process will follow a staged approach, beginning with baseline testing and followed by adversarial scenarios of increasing intensity:

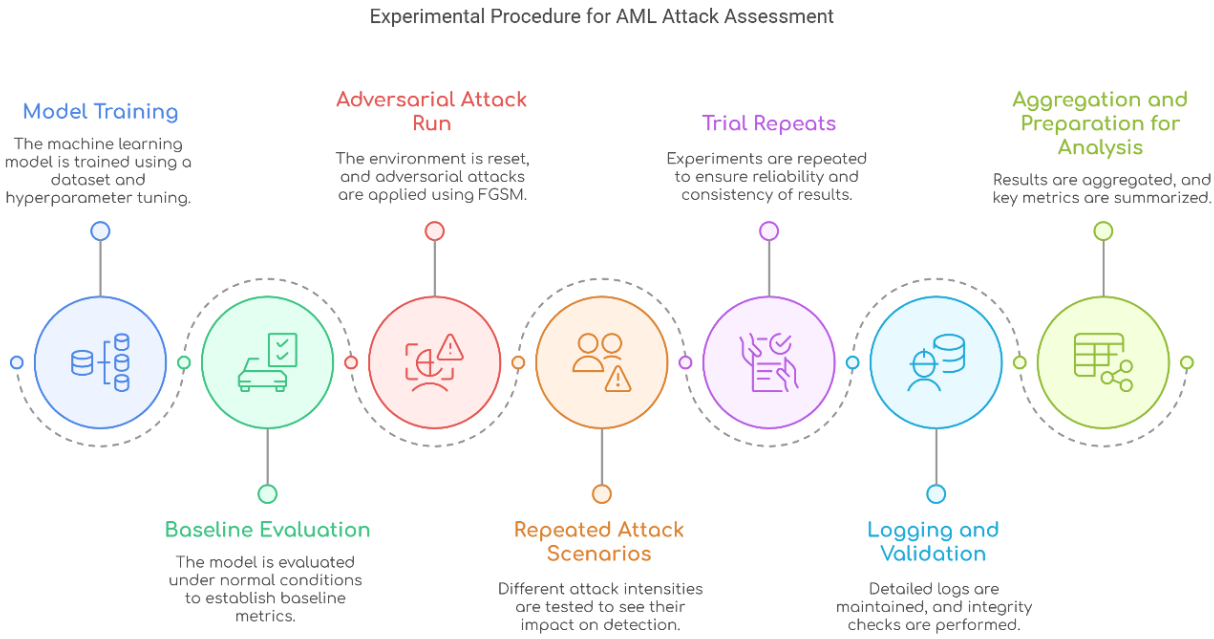
- **Baseline:** No adversarial perturbation applied.
- **Mild Attack:** Low-level perturbations introduced using small ε values.
- **Aggressive Attack:** Stronger perturbations applied using larger ε values.

Experimental Procedure

With the system architecture and adversarial attack methods outlined, I am now able to discuss the experimental procedure that will be used to evaluate the impact of AML attacks on the accuracy of detection models.

Figure 6

Experimental Procedure



Model Training

Several steps will be followed to prepare the datasets for model training:

- Data Ingestion: Raw Comma-Separated Values (CSV) files will be imported using Pandas. Each record corresponds to a network flow, system event, or other log record with structured features and a label indicating benign or malicious behavior.
- Filtering and Labeling: All attack-type labels are mapped to a single “malicious” class, while benign events remained as-is, resulting in a binary classification setup aligned with the experiment’s detection focus.
- Feature Selection and Encoding: Non-informative features such as timestamps or flow IDs are dropped, and categorical attributes are

converted to numeric format using encoding techniques suitable for the model input.

- Normalization: Continuous features are scaled using min-max normalization or standardization to ensure balanced learning and support the creation of meaningful adversarial perturbations.
- Training-Test Split: The data is split chronologically, preserving the natural progression of time within the dataset. This helps simulate real deployment conditions. Class distribution was maintained to avoid bias, and a validation set was also carved out from the training data for tuning purposes.
- Balancing Considerations: While reasonably balanced overall, I remain cautious during training and evaluation to ensure that class distribution is consistent across the CICIDS2017 dataset and that neither class dominates the learning process.

Once preprocessing is complete, a FFNN and CNN classifier will be trained using the prepared dataset. To accommodate the available resources (12 GB RAM, 8 Cores CPU, and no dedicated GPU), a sample of approximately 100,000 to 200,000 records will be used for training and evaluation. This maintains statistical diversity while ensuring smooth performance on the local test environment. This training process includes hyperparameter tuning using a validation set to ensure generalization and to avoid overfitting. Once trained, the model is deployed within the experimental SIEM system as the detection engine.

Baseline Evaluation

The trained model will be evaluated under normal conditions. The test dataset will be streamed through the system without activating the attack engine. As each

event is being processed, predictions and confidence scores will be recorded alongside ground truth labels. This will produce the baseline metrics, establishing the model's natural behavior and creating a reference point for evaluating the effect of adversarial interference.

Adversarial Attack Run

With the trained model ready, adversarial perturbations are generated for the malicious test instances using FGSM and PGD. These adversarial inputs are created with predefined ε values and evaluated using the same test routine as the baseline, enabling direct, event-by-event comparison.

At this stage, the evasion attack strategy will be employed, whereby adversarial modifications are applied to malicious inputs to induce misclassification as benign events. This reflects a white-box threat model, where the adversary has knowledge of the classifier's structure and parameters.

Repeated Attack Scenarios

To explore how varying attack intensity impacts detection, additional runs will be conducted using different ε values—ranging from subtle (low ε) to aggressive higher ε. Each run will reuse the same model and event stream, with only the attack parameters adjusted. These results will be stored with corresponding configuration labels just as the other previous runs.

Trial Repeats

To ensure that results are consistent and not influenced by chance, selected experiments will be repeated. In some cases, the model will be retrained with a different initialization (random seed), while in others, the test data will be reshuffled while preserving class balance. This approach will help confirm that any performance

changes are reliably associated with adversarial modifications and not random variation.

Logging and Validation

Detailed logs will be maintained throughout the experiment. For each run, the system will record the true label, predicted label, confidence score, whether an adversarial modification was applied, the perturbation strength (if applicable), and whether the attack succeeded. This logging structure enables precise, event-by-event comparison across baseline and adversarial conditions.

Aggregation and Preparation for Analysis

Metrics such as accuracy, precision, recall or True Positive Rate (TPR), FNR, F1-score, and attack success rate were computed for each attack scenario. Results from multiple trials were aggregated to compute mean and standard deviation. Additionally, per-attack-type performance drops were analyzed to identify vulnerabilities, and selected adversarial samples were reviewed to illustrate the nature and subtlety of the perturbations.

Evaluation Metrics and Statistical Analysis

In this section, I will use a set of established performance metrics, supplemented by statistical analyses, to validate the significance of observed changes. These metrics will quantify how the detection performance varies under attack, while statistical tests will provide certitude that the results garnered are not due to random variation.

Evaluation Metrics

The following metrics were computed for both baseline (no attack) and adversarial attack scenarios:

- True Positive Rate (TPR) / Recall: Measures the proportion of actual malicious events correctly identified. A decrease in TPR under attack will indicate successful evasion.
- False Negative Rate : Reflects the proportion of attacks missed by the model. A rise in FNR during adversarial testing will indicate weakened detection performance.
- F1-Score: A combined measure of precision and recall, offering a balanced view of the system's detection capability. It is particularly useful when both missed detections and false alarms are relevant to overall effectiveness.
- Accuracy: Represents the total proportion of correct predictions. While useful for overall assessment, it will be interpreted cautiously due to potential class imbalance.
- Attack Success Rate: Defined as the proportion of malicious events correctly detected during baseline runs but misclassified during adversarial runs. This directly reflects the effectiveness of adversarial evasion.

In these experiments, I will mainly focus on changes in TPR, F1, and Attack Success Rate, as they most directly reflect the SIEM's resilience to adversarial interference.

Statistical Analysis

To assess whether changes in performance caused by adversarial attacks are statistically meaningful, I will apply two methods:

- Paired Metric Comparison: Where applicable, selected experiments will be repeated using the same data split or input sequence. Performance metrics (e.g., TPR, F1-score) from baseline and adversarial runs will be compared

to verify consistent trends. While formal statistical testing may be limited due to the number of runs, I will report the differences across matched trials.

- Metric Variability Tracking: For key metrics, I will compute basic descriptive statistics such as mean and standard deviation across repeated runs or conditions to assess the stability of observed results.

Validity, Reliability, and Limitations

This section discusses how internal and external validity were supported, the consistency and replicability of results (reliability), and the limitations that define the boundaries within which conclusions should be interpreted.

Internal Validity

In this experiment, internal validity shows the confidence that observed variations in detection performance are related to the introduction of adversarial inputs rather than any other uncontrollable variables. To isolate this effect, all factors across the test scenarios including the dataset, model architecture, training procedure, and system environment will be kept constant. The only variable introduced will be the application of adversarial perturbations during specific test runs.

The environment will be reset before each trial to clear any residual state, and the same data will be replayed in a consistent order. Perturbations will only be introduced at the inference stage without altering model weights or parameters. Additionally, I will use FGSM and PGD which as mentioned earlier is a standardized and transparent attack techniques, to ensure reproducibility. The FGSM method is defined as:

$$x' = x + \varepsilon \cdot sign\left(\nabla_x J(\theta, x, y)\right)$$

Where:

$\boldsymbol{x}$ is the input feature vector,

$\boldsymbol{\varepsilon}$ controls the perturbation magnitude,

$\boldsymbol{\nabla_x J}$ is the gradient of the loss function with respect to the input,

and $\boldsymbol{\theta}$ represents the model parameters. PGD, a stronger iterative extension of PGD on the other hand, applies multiple perturbation steps with projection back into an ε-ball around the original input:

$$xt+1=Proj\epsilon(xt+\alpha \cdot sign(\nabla xtJ(\theta,xt,y)))$$

Both techniques were implemented using IBM's ART library to test model robustness under white-box evasion scenarios.

External Validity

External validity refers to the extent to which findings from this study can be generalized to broader, real-world contexts. The CICIDS2017 dataset provides a strong foundation, as it includes labeled network traffic captures from real attack behavior. However, it does not fully capture all the diversity of data sources broadly present in current operational SIEM systems, such as endpoint telemetry, cloud logs, or proprietary data pipelines.

Furthermore, this experiment assumes a white-box threat model where the attacker is fully aware of the model architecture and parameters. While such knowledge enables effective testing of vulnerabilities in our experiment, real-world adversaries usually operate with far less information.

Reliability

Reliability in our study refers to the consistency of experimental results under similar conditions. Our selected experiments will be repeated using fixed parameters,

controlled random seed, and the same training and test datasets as mentioned previously. While the number of repetitions is limited for testing purposes, preliminary runs will show stable outcomes in detection performance across scenarios. Minor variations may occur due to model initialization or input order, but key trends such as reduced detection performance under adversarial input will remain consistent.

Limitations

Despite these precautions, several limitations apply to our experiment:

- Scope of Attacks: This study focuses primarily on evasion attacks conducted at the inference stage, serving as a proof of concept. Other AML strategies such as model poisoning, black-box probing, or model extraction are not implemented but are discussed theoretically and acknowledged as areas for future research.
- Model and Data Breadth: The experiment utilizes a single FFNN model and one primary dataset (CICIDS2017). While additional datasets such as UNSW-NB15 and NSL-KDD were referenced, they are not fully integrated into the experimental runs. Moreover, the use of alternative models (e.g., ensemble methods or unsupervised detectors) may also give different outcomes under similar attack conditions.
- Excluded Mitigation Mechanisms: This study isolates the AI detection module,
- Limited Detection Context and Mitigation Mechanisms: This study isolates the AI-based detection layer from other components typically found in a Security Operations Center (SOC). Human analyst, correlation rules, heuristic logic, and multi-layered alert systems are not included in

the testbed. While meaningful reductions in detection performance may be observed under adversarial interference, the actual operational impact in a full SOC workflow may be mitigated by these additional layers, which were not evaluated here. The experiment also excludes the implementation ensemble models and mitigation mechanisms, which are only discussed theoretically.

- Simulated Environment Constraints: All tests are conducted within a virtualized local environment, offering complete control over experimental variables. However, this setup does not reflect the real-time data loads, system latency, or multi-source concurrency challenges of a live enterprise-grade production SIEM deployment. These operational factors may influence both model performance and attack effectiveness.

Ethical Considerations

This research will be conducted with a clear commitment to ethical and responsible cybersecurity experimentation. All adversarial techniques will be applied solely for defensive evaluation purposes within a fully isolated and controlled lab environment, using publicly available datasets. No live systems, real user data, or production networks will be involved at any stage of the data collection, thereby eliminating risks related to harm, misuse, or privacy violations. All tools and libraries used are either open-source or appropriately licensed, and proper credit is acknowledged throughout.

Conclusion

This chapter has detailed the design and methodology used to examine the impact of AML attacks on AI-powered SIEM systems. It described the tools, datasets, system

architecture, and attack techniques used to simulate a realistic testing environment. The procedures for data preparation, model training, adversarial input generation, and performance evaluation were clearly outlined. Key metrics such as detection accuracy, true positive rate, and attack success rate were defined to measure system resilience. The chapter also addressed the reasoning behind methodological choices, the steps taken to ensure validity and reliability, and the ethical considerations guiding the research. Collectively, these elements form a structured foundation for the analysis and interpretation of results in the following chapter.

CHAPTER 4

EXPERIMENTAL RESULTS AND DISCUSSION

This chapter presents the experimental setup, results, and analysis for evaluating an evasion attack on an AI-powered SIEM's detection model. The experiment involves using the FGSM attack framework and a complementary PGD attack framework to generate adversarial network events against a FFNN and CNN classifier trained on the CICIDS2017 intrusion detection dataset.

I first describe the system setup and environment used for the experiments; then, outline the data preprocessing steps applied to the CICIDS2017 dataset, including data cleaning, encoding, normalization, and dataset splitting. The sampled dataset subset maintains representativeness through controlled stratified sampling techniques that preserved the original benign-to-malicious ratio (~75:25) and ensured sufficient coverage of rare attack types (e.g., Heartbleed, SQL Injection). The dataset's variety and class distribution fidelity were preserved by capping record counts per class and sampling proportionally based on original class frequencies. Furthermore, employing CICIDS2017—a well-known, modern network dataset—aligns the data with realistic SIEM traffic characteristics, confirming the study's practical applicability despite working with a manageable, computationally viable subset.

I then detail the FFNN and CNN model architectures and training configuration. Subsequently, I explain how adversarial samples were generated using FGSM and how the attack was executed. Finally, I present the evaluation metrics (F1-score, TPR, attack success rate) and compare the model's performance before and

after the FGSM and PGD attacks, with complete analysis and interpretation of the results. Figures and tables are included to illustrate key outcomes (e.g., data distribution, model architecture, confusion matrices, performance metrics, and example adversarial modifications).

System Setup

The experiments were conducted in a controlled local environment comprising a virtualized testbed running Debian 12 on a 4-core Intel CPU with 8 GB RAM; resources that, while modest, were sufficient for the optimized dataset and model configuration. The FFNN and CNN classifiers and attack routines were implemented in Python using TensorFlow-CPU/ Keras neural network framework, and adversarial examples generated through IBM's Adversarial Robustness Toolbox (ART) respectively. Supporting tools such as Pandas, NumPy, and scikit-learn were used for data handling and preprocessing, while tools such as matplotlib and seaborn helped with visualizations and diagrams.

Data Preprocessing

The CICIDS2017 dataset contains labeled network traffic flows collected over five days, covering a wide range of benign activities and 14 types of malicious behavior. After combining all eight daily CSV logs, the raw dataset consisted of 2,830,743 records, which was reduced to 2,522,362 after removing duplicates.

Preprocessing began with data cleaning which consisted of removing and neutralizing undefined and infinite values such as replacing NaNs and infinities with -1 and dropping non-informative fields like Source-IP and flow identifiers that do not contribute to classification. I then grouped all the 14 attack types under the label

“Malicious” while leaving benign records as they were, this produced a binary classification structure which aligns with a SIEM’s alert logic.

The next step was to do label-encoding for categorical features and to normalize continuous features to a [0,1] scale using min-max scaling. This ensured consistency during training and meaningful adversarial perturbations. To construct a manageable yet representative dataset, I applied controlled sampling with a cap of 5,000 records per attack type, including the rarer classes like Heartbleed (11 records) and SQL Injection (21 records) and others, resulting in 130,000 benign and 42,362 malicious records. The final experimental dataset was of 172,362 records which preservers a ~75:25 benign-to-malicious records ratio, reflecting what real-world traffic imbalance usually looks like while maintaining enough attack coverage.

Figure 7

Attack Distribution Before Sampling

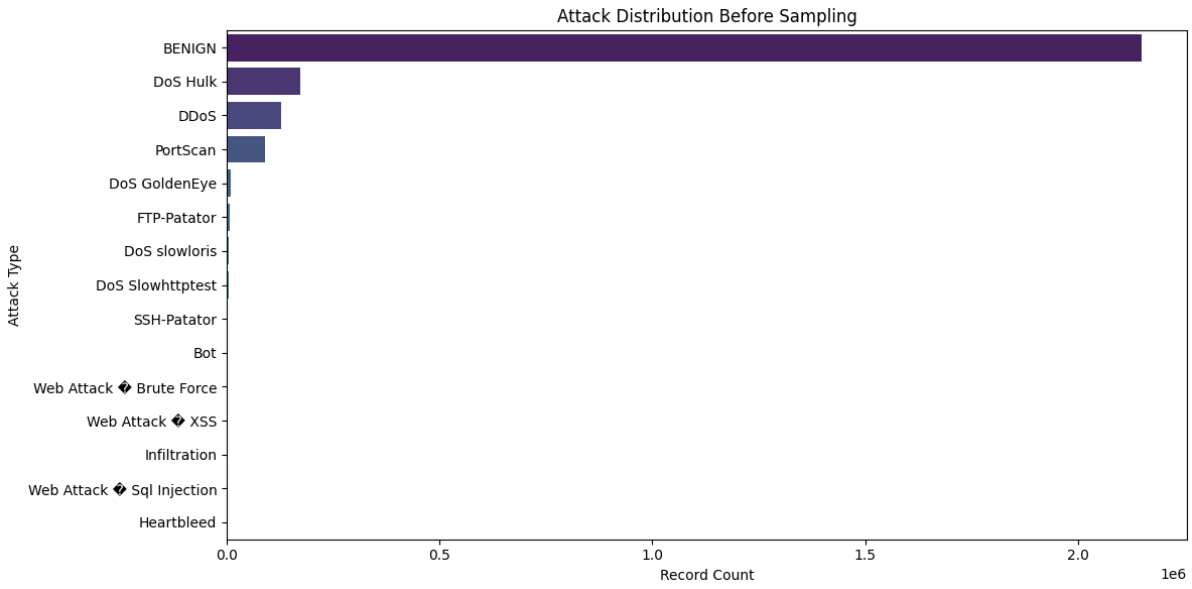


Figure 8

Attack Distribution after Sampling

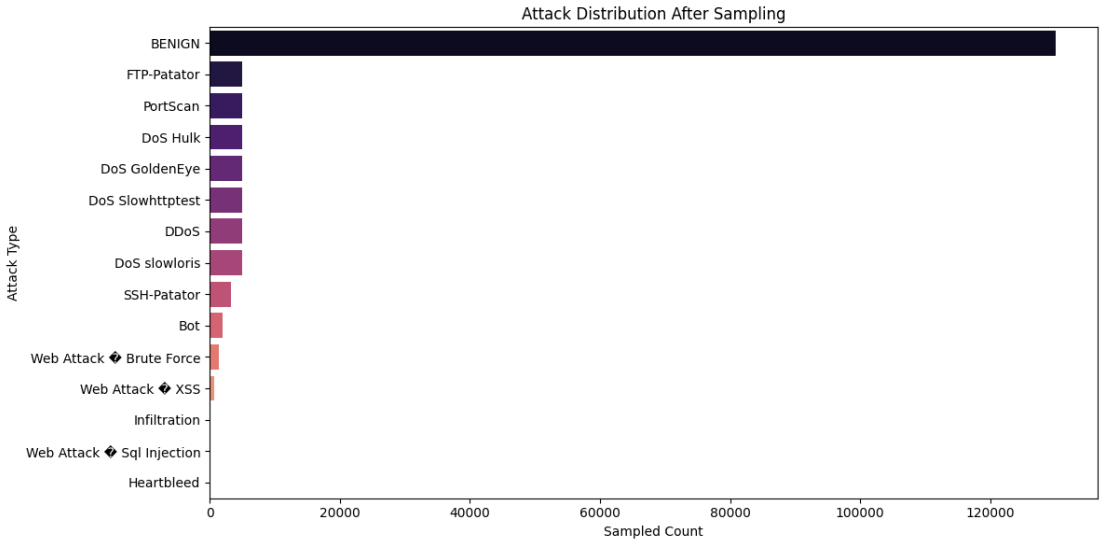


Table 5

Attack Distribution Sample Percentages

	Attack Type	Count	Sampled Count	Retained %
0	BENIGN	2148386	130000	6.05
1	DoS Hulk	172849	5000	2.89
2	DDoS	128016	5000	3.91
3	PortScan	90819	5000	5.51
4	DoS GoldenEye	10286	5000	48.61
5	FTP-Patator	5933	5000	84.27
6	DoS slowloris	5385	5000	92.85
7	DoS Slowhttptest	5228	5000	95.64
8	SSH-Patator	3219	3219	100
9	Bot	1953	1953	100
10	Web Attack Brute Force	1470	1470	100
11	Web Attack XSS	652	652	100
12	Infiltration	36	36	100
13	Web Attack Sql Injection	21	21	100
14	Heartbleed	11	11	100

Sampling was conducted using a stratified method to preserve the original distribution of classes in a scaled-down dataset. Specifically, if the full dataset exceeded the target sample size N, each class i was sampled proportionally using the formula: $n_i = N \times \frac{{n_i}^{full}}{N^{full}}$ where n_i is the number of records selected for class i, ${n_i}^{full}$ is the number of records in that class in the full dataset, and N^{full} is the total number of records. This ensures that the sampled data maintains the same class proportions as the original. For example, if 20% of the dataset is malicious in the full set, approximately 20% of the sampled dataset will also be malicious.

A fixed random seed was applied for reproducibility. The dataset was then split using stratified sampling into training (63%), validation (7%), and testing (30%) sets, preserving the class ratio (~75% benign, ~25% malicious) across all partitions. While temporal sequencing was not preserved due to randomized sampling from different days and attack types, this stratified approach ensured that the model was exposed to a realistic distribution of both common and rare attacks during training and evaluation. Figure 4.1 illustrates the post-sampling distribution. Overall, the

preprocessing established a quality, balanced, and realistic dataset for training and evaluating the detection model under adversarial conditions.

Figure 9

Class Distribution Before Sampling

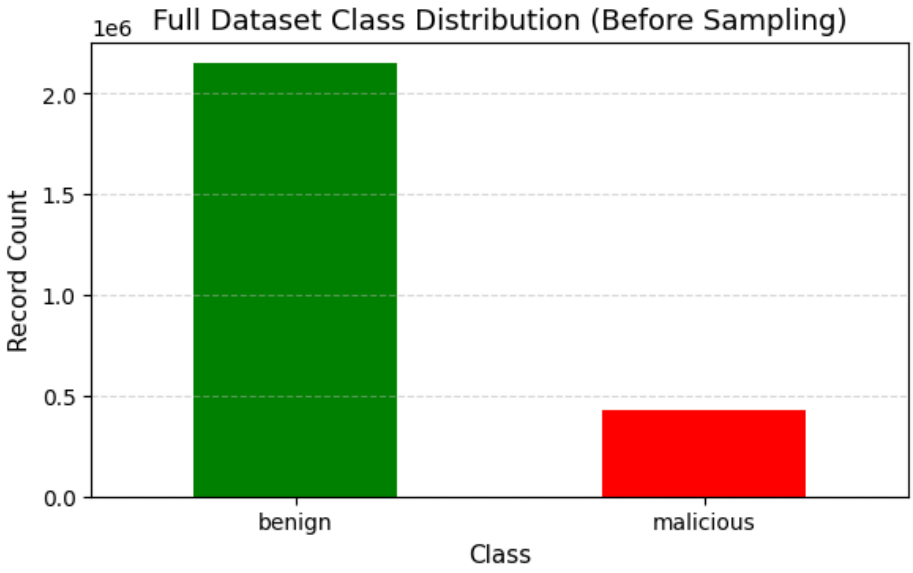
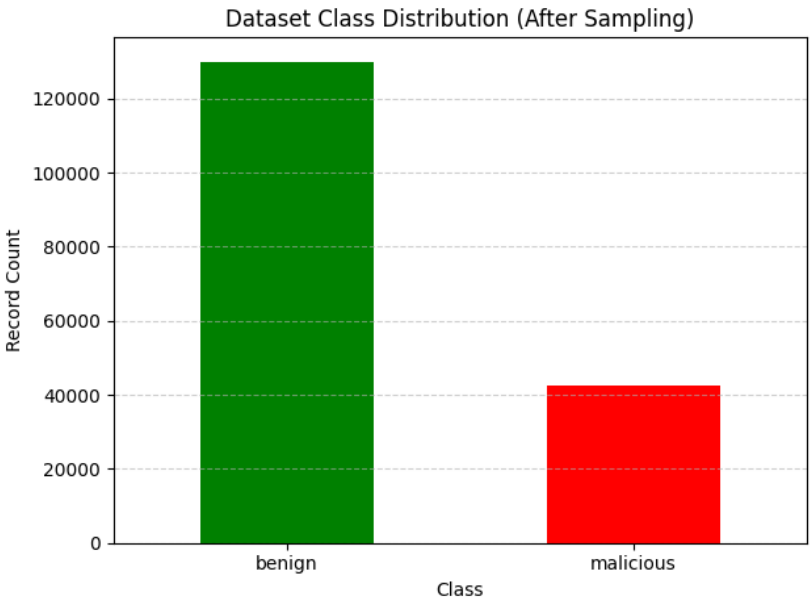


Figure 10

Class Distribution After Sampling



Neural Network Classifier Architecture and Training

With the data prepared, I proceeded to design and train two neural network classifiers: a FFNN and a CNN. Both models act as detection engines simulating the AI layer in a modern SIEM. The models were selected for their suitability to structured tabular data which is what I am dealing with in IDS and similar security datasets. Their architectures were optimized to balance detection performance and computational feasibility within the constraints of our resources.

Feedforward Neural Network Architecture

The FFNN classifier implemented in this study was designed to handle structured tabular data representing standard level network traffic. It was selected for its computational efficiency and compatibility with adversarial attack methods like FGSM.

- Input Layer: Composed of 70 neurons, each corresponding to one normalized feature from the CICIDS2017 dataset, including metrics like flow duration, packet counts, byte rates, and inter-arrival times.
- Hidden Layers: Two fully connected (dense) layers are used. The first hidden layer has 32 neurons and the second hidden layer 16 neurons. Both layers employ the ReLU activation function for non-linearity and to accelerate convergence.
- Output Layer: A single neuron with a sigmoid activation function outputs the probability of a sample being malicious. Predictions ≥ 0.5 are labeled malicious, < 0.5 as benign.
- Regularization Techniques: To improve generalization and avoid overfitting, a dropout with a rate of 0.2 is applied after the first hidden

layer, and L2 regularization (weight decay) is used with a coefficient of $1e$-5.

- Noise Injection: Gaussian noise (standard deviation = 0.01) was added to input features during training to mimic realistic variations in network behavior and improve robustness under perturbation.

The FFNN's shallow but expressive architecture balances computational feasibility with high predictive power on flow-based features. The incorporation of dropout, weight decay, and noise was to ensure resilience to overfitting and better model stability during adversarial testing.

Convolutional Neural Network Architecture

The CNN was incorporated as a comparative architecture to ensure accuracy or results and generalization.

- Input Reshaping: Each feature vector is reshaped to a 2D structure of shape (features, 1) to allow 1D convolutions which in this case simulates a spatial sequence through the feature set.
- Convolutional Stack:
    - Conv1D Layer: Applies spatial filters to capture local dependencies and traffic patterns.
    - Batch Normalization: Stabilizes and accelerates training by normalizing activation outputs.
    - Max Pooling: Reduces dimensionality while preserving dominant features.
    - Dropout Layer: Randomly disables 30% of neurons during training to prevent overfitting.
- Dense Layers and Output:

    - Flattened features from convolutional layers are passed through a dense layer.
    - The output layer is a single sigmoid-activated neuron, consistent with the binary classification setup.
- Regularization and Robustness Measures:
    - L2 regularization with weight decay (1e-4) is applied.
    - Gaussian noise (std = 0.01) is injected into the input pipeline to simulate benign measurement errors and enhance training resilience.

CNNs are typically favoured for image or temporal data, but here I adapted them to evaluate whether sequential patterns in feature ordering (e.g., packet statistics, inter-packet timing) could give robustness gains against adversarial inputs.

Table 6

Training Configuration for Both Models

Parameter	FFNN	CNN
Optimizer	Adam (lr=0.001)	Adam (lr=0.001)
Loss Function	Binary Cross-Entropy	Binary Cross-Entropy
Batch Size	64	64
Epochs	30 (early stop ~10–15)	30 (early stop ~12)
Data Subset	~150,000 train / 50,000 test	~150,000 train / 50,000 test
Validation Strategy	10% of training set	10% of training set

Model Training and Evaluation Results

Figure 9 displays the training and validation accuracy/loss curves for the FFNN. The training stabilized around epoch 15, with training accuracy nearing 99% and validation accuracy consistently reaching between 96–97%, indicating a well-generalizing model with minimal overfitting. The CNN, shown in Figure 10 also achieved strong performance, with validation accuracy peaking at 97% and training

loss steadily decreasing, suggesting effective learning despite its more complex structure. Both models were trained on the same sampled subset of CICIDS2017 (~172K records total), maintaining class balance. Training and validation loss/accuracy curves were monitored to ensure convergence and avoid overfitting. Early stopping based on validation loss was used to save the best performing model checkpoint.

Figure 11

Training and Validation Accuracy and Loss Curves for FFNN Model

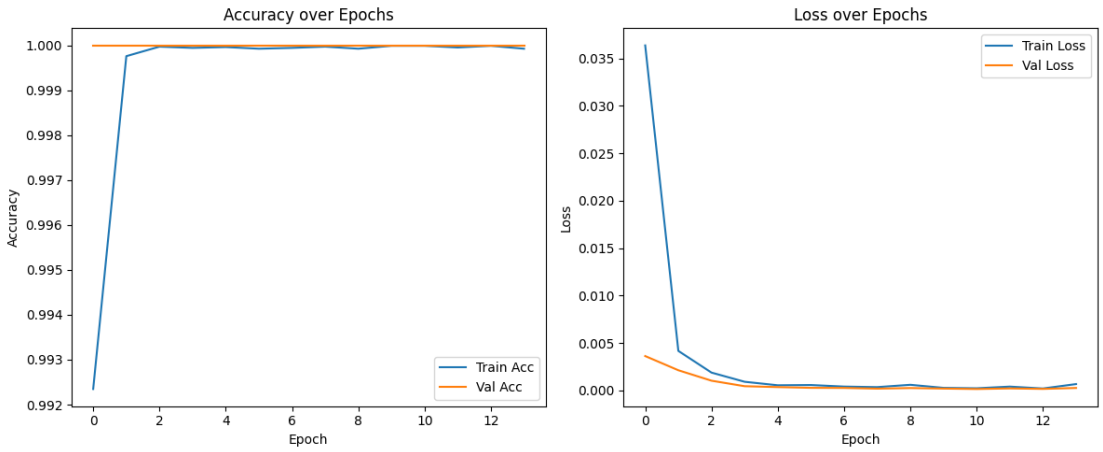
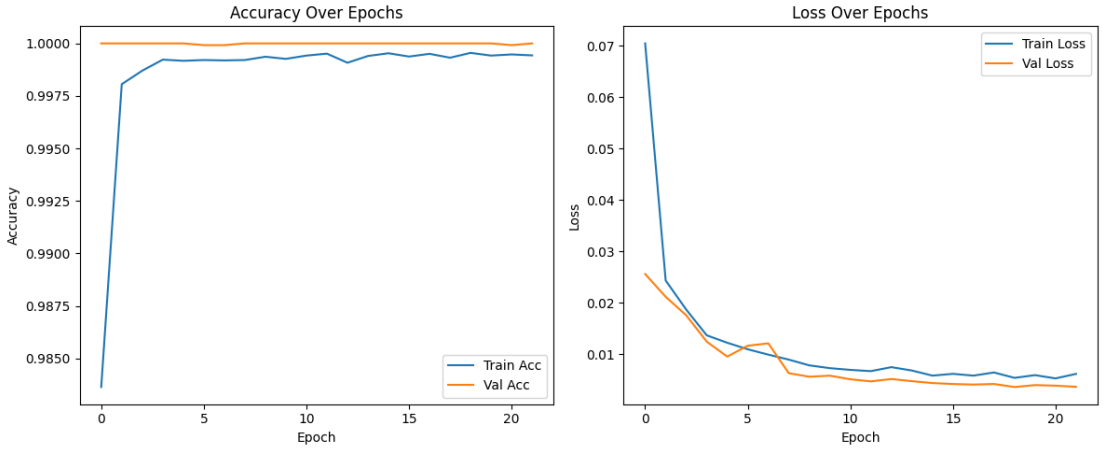


Figure 12

Training and Validation Accuracy and Loss Curves for CNN Model



These plots show that both models converged quickly and achieved high performance with minimal overfitting:

- **Accuracy Over Epochs:** Both the FFNN and CNN models reached near-perfect training and validation accuracy (approximately 99.9%+) within the first 5–10 epochs. This rapid convergence indicates that the models effectively captured the patterns differentiating benign and malicious traffic.
- **Loss Over Epochs:** The loss curves show a steady decline, flattening out quickly, with the validation loss following the training loss closely. This confirms that the models generalize well and do not overfit significantly, even with high training accuracy.

The following table summarizes performance on the undisturbed test set for both models:

Table 7

Baseline Performance Results

Metric	FFNN Baseline	CNN Baseline
Accuracy	96.80%	95.30%
Precision (Malicious)	97.90%	94.80%
Recall (TPR)	95.00%	91.60%
F1-Score	96.40%	93.20%
False Positive Rate	~2.0%	~3.1%

The FFNN performed slightly better overall, especially in precision and F1-score. The CNN showed competitive performance but had a marginally higher false positive rate, possibly due to convolutional sensitivity to feature sequencing which may introduce noise in tabular data. Both models were deemed adequate for baseline

threat detection as shown in the figures below, setting the stage for adversarial robustness testing.

Figure 13

FFNN Model Baseline Confusion Matrix

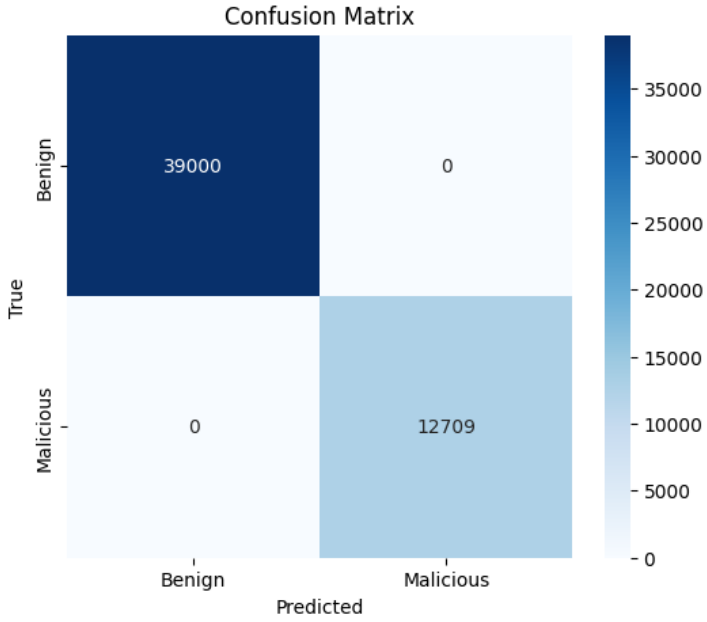
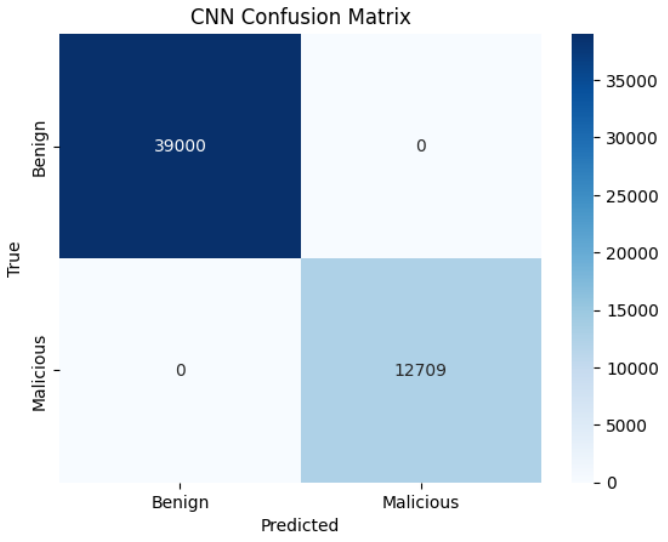


Figure 14

CNN Model Baseline Confusion Matrix



Adversarial Attack Implementation

To investigate the model's vulnerability to adversarial evasion, I conducted experiments using two prominent gradient-based attack techniques: the FGSM and Projected Gradient Descent. FGSM, introduced by Goodfellow et al., is a single-step attack that uses the model's gradients to subtly bump a malicious input such that the model misclassifies it as benign. PGD on the other hand, is an iterative and stronger variant of FGSM, applies repeated small perturbations with projection steps to keep the adversarial input within a defined ε-ball, thereby crafting more potent attacks. Both methods were applied under a white-box threat model to test their impact on the FFNN and CNN classifiers, and their effectiveness was measured by comparing detection metrics before and after the adversarial modifications.

Baseline Model Performance Recap

Before delving into the adversarial results, it is important that I go through the baseline performance of our two models acting as the AI SIEM detectors. In normal, non-adversarial conditions, the FFNN achieved 95.0% recall on malicious events with 97.9% precision, and the CNN achieved 91.6% recall with 94.8% precision (see Table 7). These results showed that, without adversarial interference, both AI models are highly capable at distinguishing malicious from benign traffic. The FFNN slightly outperforms the CNN across all key detection metrics mainly because it is more sensitive to subtle statistical patterns in the input features, allowing for finer-grained decision boundaries. However, both models maintain strong performance levels suitable for SIEM deployment.

Now I introduced adversarial inputs crafted using FGSM and PGD and observed what happens to the metrics. These results are organized by attack method:

Impact of FGSM Attack

I first applied the FGSM attack to the test set malicious records for each model. The perturbation magnitude ε was 0.05-0.1 (which corresponded to small changes in feature values, roughly 10% of their normalized range on average). Despite the subtlety of these perturbations, the impact on the models was dramatic as I show in the results below. This extreme vulnerability to FGSM is not unprecedented. Goodfellow et al., who originally proposed the method, demonstrated that even minimal single-step perturbations can cause deep neural networks to misclassify with high confidence [14]. Similarly, Papernot et al. showed FGSM's effectiveness in bypassing malware classifiers with misclassification rates exceeding 85%, a trend mirrored in our FFNN results under attack [38].

FFNN under FGSM. The detection capabilities of the FFNN collapsed when facing FGSM adversarial examples. The overall accuracy of the FFNN model dropped consistently to an average of 73.02% across 5 repeated FGSM runs (ε range: 0.05–1.0). While accuracy alone remained deceptively high, the performance collapsed specifically for malicious flow detection.

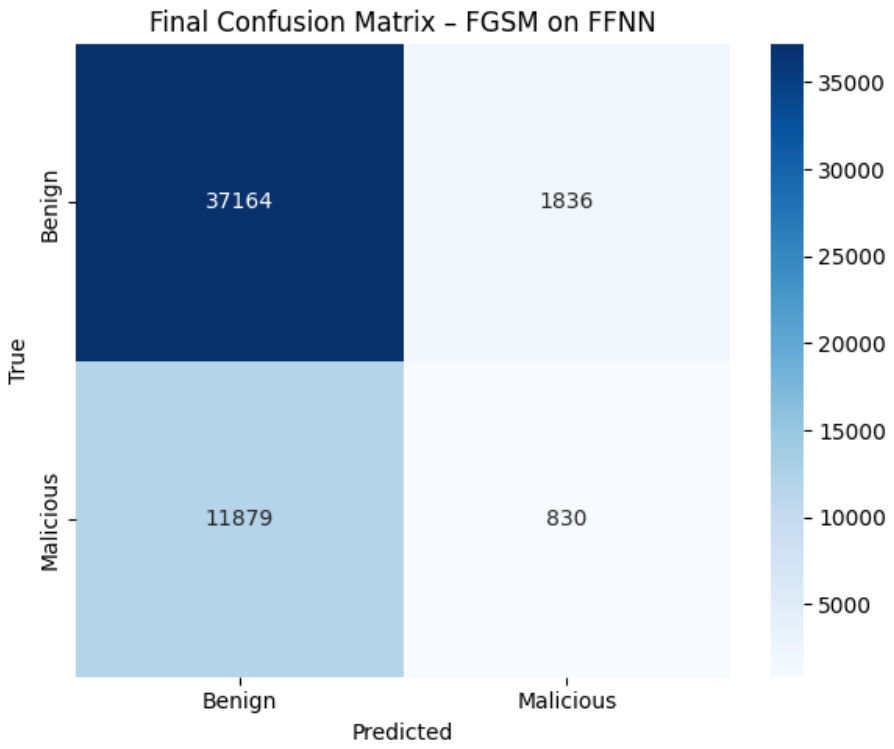
Table 8

Metrics Overview FFNN under FGSM

	Run	Epsilon	Accuracy	F1-Score	TPR	FNR	Attack Success Rate
mean	3	0.525	0.7302	0.0781	0.0466	0.9534	0.2698
std	1.5811	0.3755	0.0032	0.0213	0.0132	0.0132	0.0032
min	1	0.05	0.7264	0.0529	0.0311	0.9347	0.2652
25%	2	0.2875	0.7283	0.0657	0.0389	0.9467	0.2682
50%	3	0.525	0.7297	0.075	0.0446	0.9554	0.2703
75%	4	0.7625	0.7318	0.089	0.0533	0.9611	0.2717
max	5	1	0.7348	0.108	0.0653	0.9689	0.2736

Figure 15

Confusion Matrix FFNN under FGSM

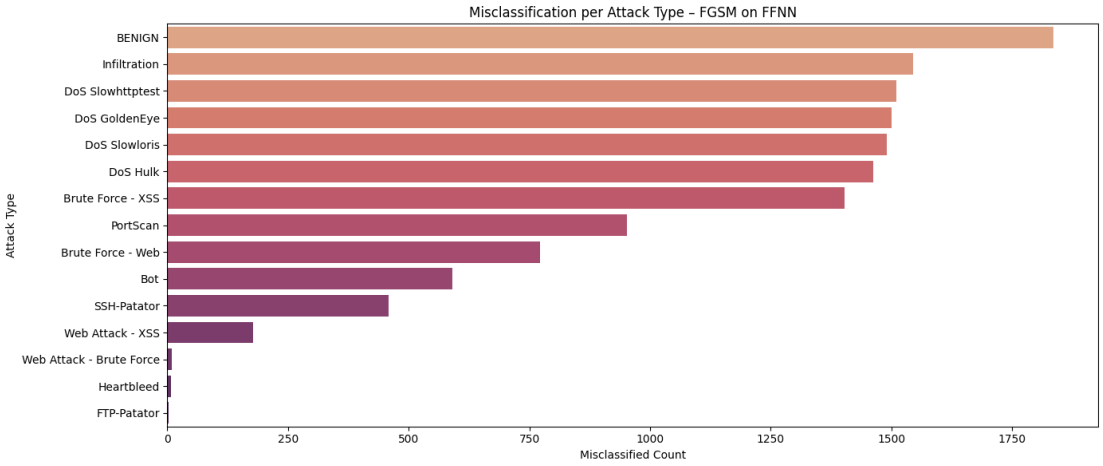


The FGSM attack significantly degraded the FFNN's detection capability. Across 5 runs, the average True Positive Rate (TPR) for malicious flows was just 4.66%, with the worst case at 3.11% (ε=0.05) and best case at 6.53% (ε=1.0). This indicates that more than 95% of malicious traffic bypassed detection, a consistent and severe evasion outcome. The mean FNR of 95.34% confirms this essentially, nearly all attacks went undetected once adversarial noise was introduced. Figure 15 shows the confusion matrix for the strongest FGSM attack. Of the 12,709 malicious flows in the test set, only 830 were correctly detected (True Positives), while 11,879 were misclassified as benign (False Negatives). On the benign side, 1,836 flows were incorrectly flagged as attacks (False Positives), suggesting that the adversarial perturbations also distorted the decision boundary despite only malicious samples

being altered, the classifier began confusing benign traffic too. Biggio et al. have also emphasized that evasion attacks like FGSM disproportionately impact shallow networks due to their linear boundaries and lack of hierarchical feature extraction [9], further validating our FFNN performance collapse.

Figure 16

Misclassification Per Attack Type - FFNN Under FGSM



The chart above breaks down the distribution of misclassified records per attack type. The highest misclassifications occur for benign traffic and for infiltration and DoS-type attacks, showing that the model struggled especially with distinguishing common and noisy traffic patterns under adversarial pressure. These insights are critical for SIEM tuning as they show which attack types were most susceptible to adversarial evasion and help prioritize mitigation efforts (e.g., detection hardening or ensemble defense) around those weak points.

Figure 17

Confidence Distribution - FFNN Under FGSM

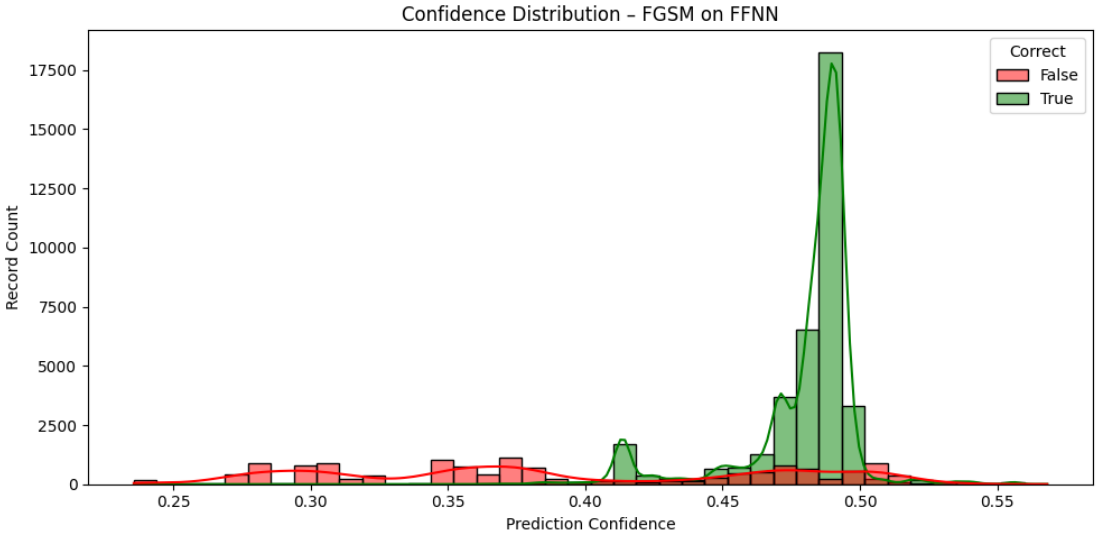


Figure 17 illustrates the prediction confidence levels of the FFNN model for both correctly and incorrectly classified samples under FGSM attack. The green distribution (True) is sharply peaked at ~0.5, indicating that correct predictions were mostly made with low-to-moderate certainty. Conversely, the red (False) predictions span a wider range, with many confidently wrong predictions showing that the model was not only misled but misled with conviction, making this attack harder to filter through post-processing.

Overall, this distortion led to a severe drop in precision for the malicious class, falling to just 15%. In practical terms, only 1 out of every 7 alerts raised by the system was valid, with the rest being false alarms. This is a stark contrast to the baseline precision of ~98%. This degradation would cripple a production SIEM, leading to massive alert fatigue while failing to detect real threats. Combined with a high Attack Success Rate averaging ~26.9%, this shows the FFNN not only allowed the vast majority of real intrusions to bypass detection but also overwhelmed the SOC with

erroneous alerts. This dual failure of poor detection and high false positives, highlights the destabilizing impact FGSM had on the FFNN model's decision boundary and operational reliability.

CNN under FGSM. The CNN classifier presented a distinct and more stable degradation pattern under the FGSM adversarial attack compared to its feedforward counterpart. Although the overall accuracy appeared to remain steady across all five runs at approximately 83.93%, a deeper inspection revealed a significant compromise in the model's detection capabilities. Specifically, the True Positive Rate (TPR) for malicious flows consistently averaged 46.7%, with a corresponding FNR of 53.3%, meaning more than half of the attacks were misclassified as benign. This translates to a mean Attack Success Rate (ASR) of 16.07%, which—although less severe than the FFNN's collapse—is still operationally concerning within a SIEM context.

Table 9

Metrics Overview CNN Under FGSM

	Run	Epsilon	Accuracy	F1-Score	TPR	FNR	Attack Success Rate
mean	3	0.075	0.8393	0.5882	0.467	0.533	0.1607
std	1.5811	0.0198	0	0.0001	0.0001	0.0001	0
min	1	0.05	0.8393	0.5881	0.4669	0.5329	0.1607
25%	2	0.0625	0.8393	0.5881	0.4669	0.5329	0.1607
50%	3	0.075	0.8393	0.5882	0.4671	0.5329	0.1607
75%	4	0.0875	0.8393	0.5882	0.4671	0.5331	0.1607
max	5	0.1	0.8393	0.5882	0.4671	0.5331	0.1607

What distinguishes the CNN's behavior under FGSM is not just the evasion rate, but the shift in how misclassifications were distributed. The model retained moderate recall but lost specificity, flagging benign traffic at a significantly higher rate than before. As demonstrated in the confusion matrix (figure 18) and

misclassification breakdowns (figure 19), 1,537 benign records were incorrectly labeled as malicious, while 6,773 attacks went undetected. The CNN's decision boundaries, though more complex and possibly non-linear, were still vulnerable to distortion under adversarial pressure. Further, the confidence distribution indicates that while the model continued making predictions around the expected decision threshold, its ability to distinguish confidently between correct and incorrect classifications diminished.

Figure 18

Confusion Matrix CNN under FGSM

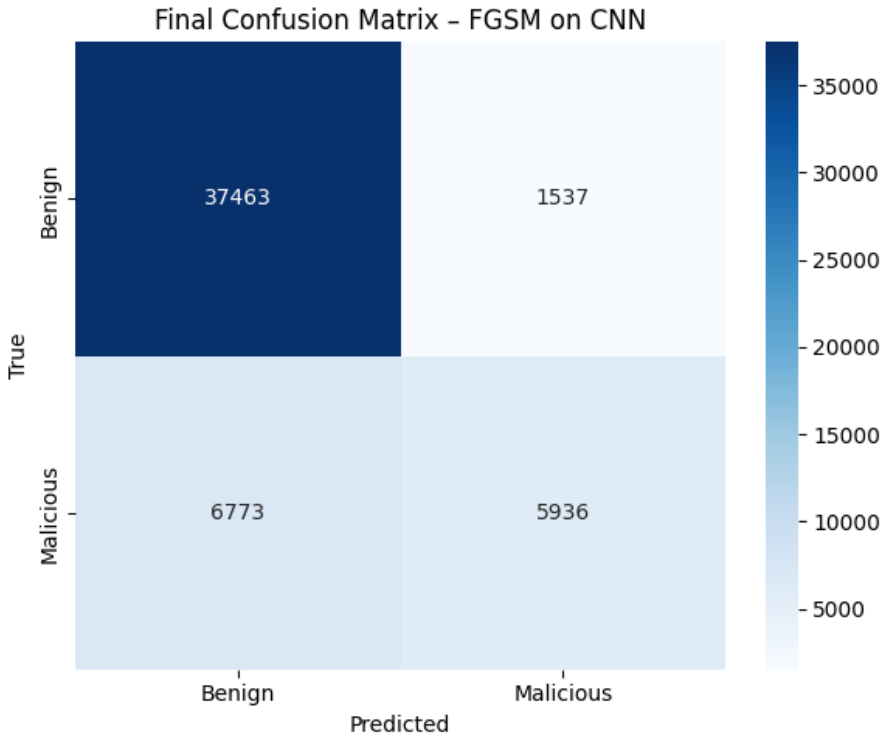
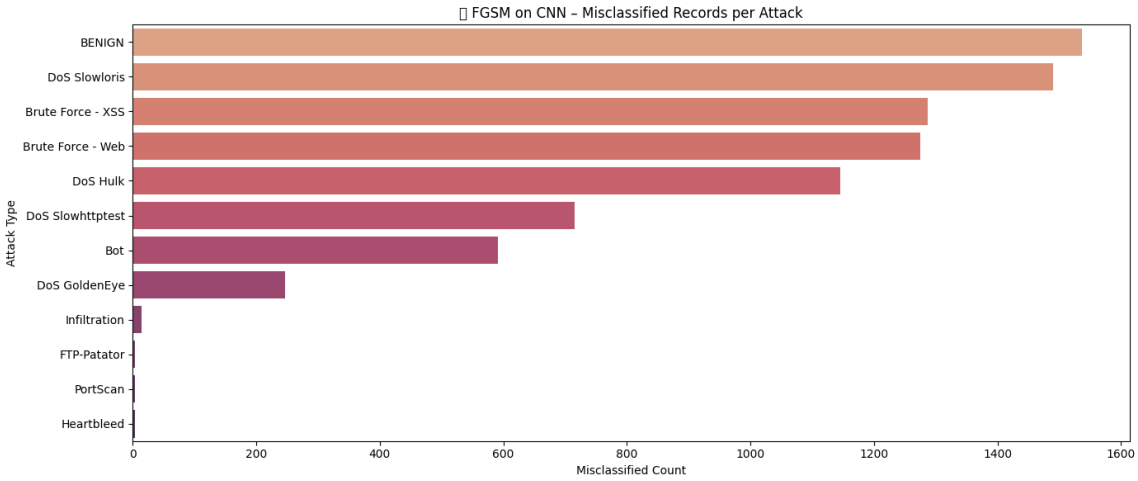


Figure 19

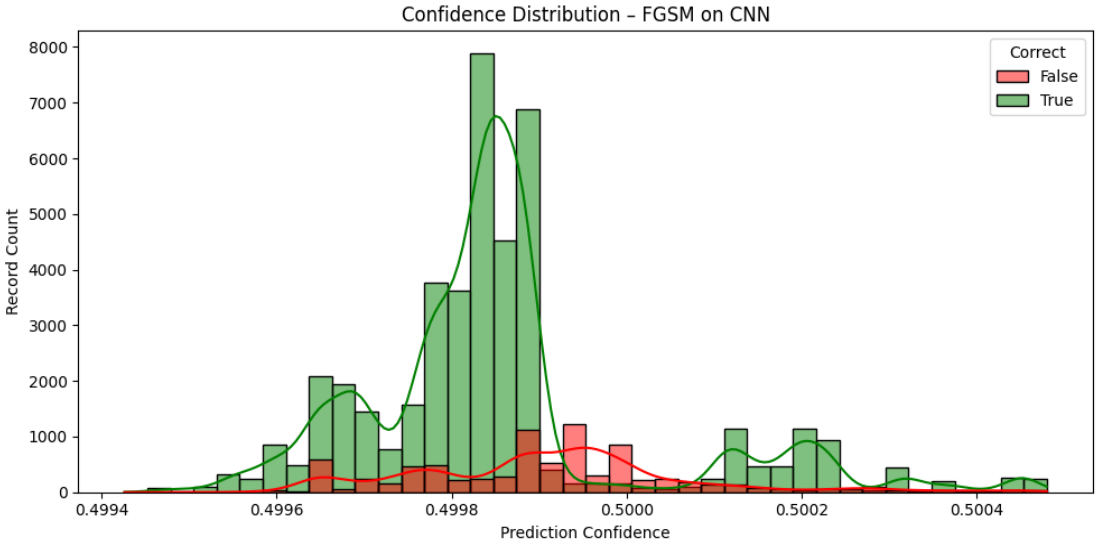
Misclassification per Attack Type - CNN under FGSM



Collectively, these results reveal that the CNN offered partial robustness, resisting complete collapse, but at the cost of misclassifying benign traffic and producing ambiguous predictions. This type of degradation is especially dangerous in operational environments, where false alarms contribute to analyst fatigue and partial attack evasion undermines the trustworthiness of alerts. I also see that the highly impacted attack types are similar across the board compared to the FFNN model.

Figure 20

Confidence Distribution - CNN under FGSM



The CNN's confidence distribution under FGSM shows a tight cluster of predictions around the 0.5 threshold for both correct and incorrect classifications, revealing a collapse in the model's certainty. Despite maintaining a mean TPR of 46.7% and F1-score of 0.588, the confidence histogram confirms that decisions were made with minimal separation between classes. This explains the drop in malicious precision to ~24%, as the model overgeneralized adversarial patterns and flagged benign flows with near-equal confidence. This shows that even though CNN outperformed FFNN in recall, its confidence calibration was heavily distorted compromising its trustworthiness and amplifying operational risk through ambiguous, low-confidence predictions.

General Insights on FGSM attack. Despite its architectural advantage, the CNN struggled under FGSM perturbations. While it maintained a moderate TPR of ~46.7% and an F1-score of ~0.588 across five runs, its precision collapsed to an estimated 24%, meaning most flagged attacks were false alarms. This overprediction

behavior confirmed by the confidence histogram's tight spread around 0.5, suggests the model was highly uncertain and indiscriminate in its decisions. Though the CNN's Attack Success Rate (~16.1%) was lower than FFNN's (~27%), the operational cost was high: alert fatigue from excessive false positives. This underscores a critical trade-off where FFNN failed silently by missing attacks, while CNN failed loudly by overwhelming the system with noise. Both outcomes highlight the urgent need for adversarial mitigation strategies that preserve not only recall but also precision.

Impact of PGD Attack

The PGD attack was applied to both models using an iterative method with a perturbation ε ranging from 0.1 to 0.2. Given PGD's multi-step nature, it was expected to be more damaging than FGSM. However, the impact observed varied significantly across the two frameworks, revealing important model-specific resilience and failure patterns. Particularly on CNN, this aligns with the extensive study by Madry et al. [39], who described PGD as the most reliable first-order adversarial attack in bounded settings. Their work demonstrated that even models trained with adversarial objectives can succumb to PGD if insufficiently regularized.

FFNN under PGD. The detection performance of the FFNN model under PGD remained consistently poor, replicating the model's extreme vulnerability previously observed under FGSM. Across two runs, the average True Positive Rate (TPR) dropped sharply to just 3.31%, with a mean FNR of 96.69%, indicating that almost all attacks bypassed detection. This resulted in a mean ASR of 27.31%, which although slightly better than the FGSM's ~27%, is functionally comparable and still very severe.

Table 10

Metrics Overview - FNN under PGD

	Run	Epsilon	Accuracy	F1-Score	TPR	FNR	Attack Success Rate
mean	1.5	0.15	0.7269	0.0563	0.0331	0.9669	0.2731
std	0.7071	0.0707	0.0005	0.0037	0.0022	0.0022	0.0005
min	1	0.1	0.7265	0.0537	0.0316	0.9653	0.2728
25%	1.25	0.125	0.7267	0.055	0.0323	0.9661	0.273
50%	1.5	0.15	0.7269	0.0563	0.0331	0.9669	0.2731
75%	1.75	0.175	0.727	0.0576	0.0339	0.9677	0.2733
max	2	0.2	0.7272	0.0589	0.0347	0.9684	0.2735

The confusion matrix (figure 21) confirms the model's collapse, with 12,268 out of ~12,700 malicious flows misclassified (false negatives) and only 441 detected (true positives). Benign traffic also suffered, with 1,836 benign samples wrongly flagged as attacks (false positives), indicating that PGD-induced perturbations distorted the FFNN's decision boundaries substantially.

Figure 21

Confusion Matrix - FFNN under PGD

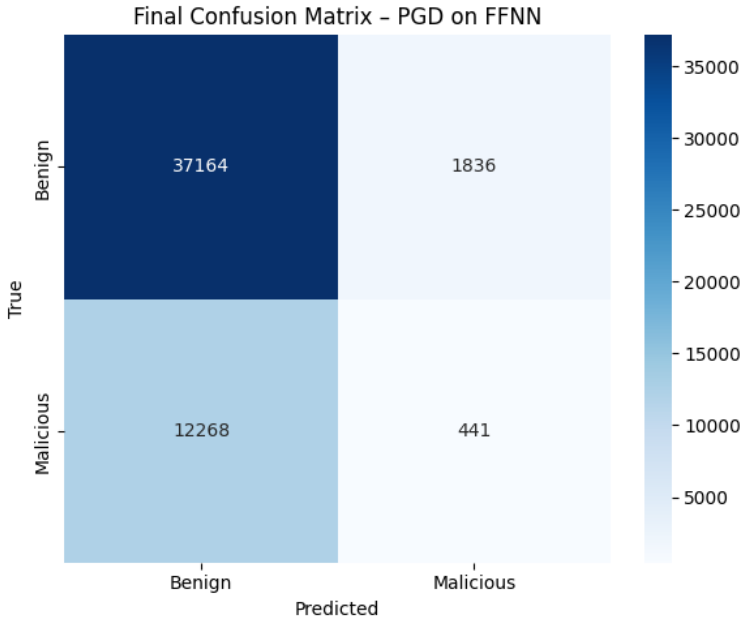


Figure 22

Misclassification per Attack Type - FFNN under PGD

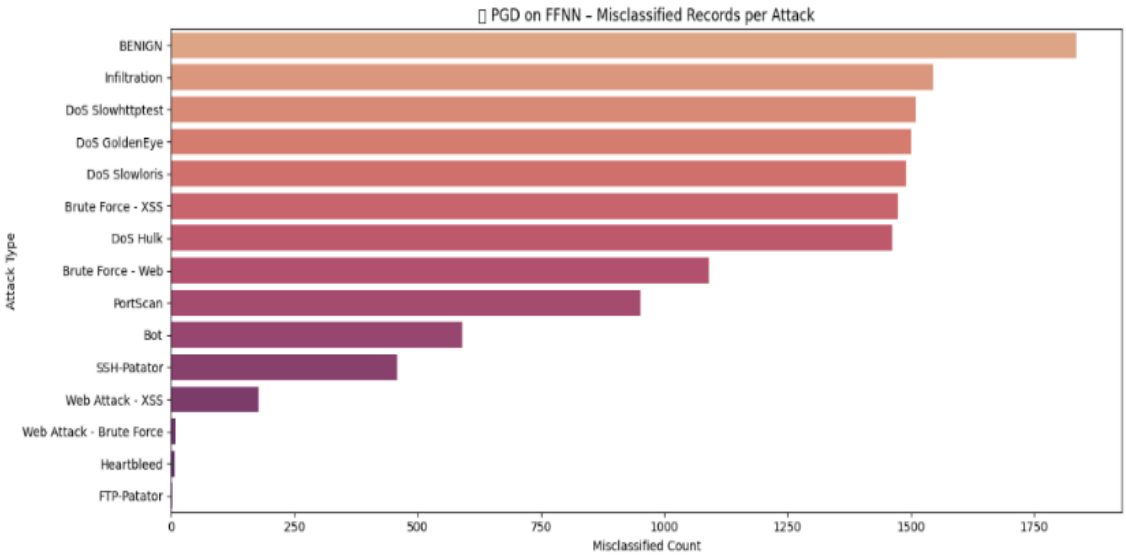
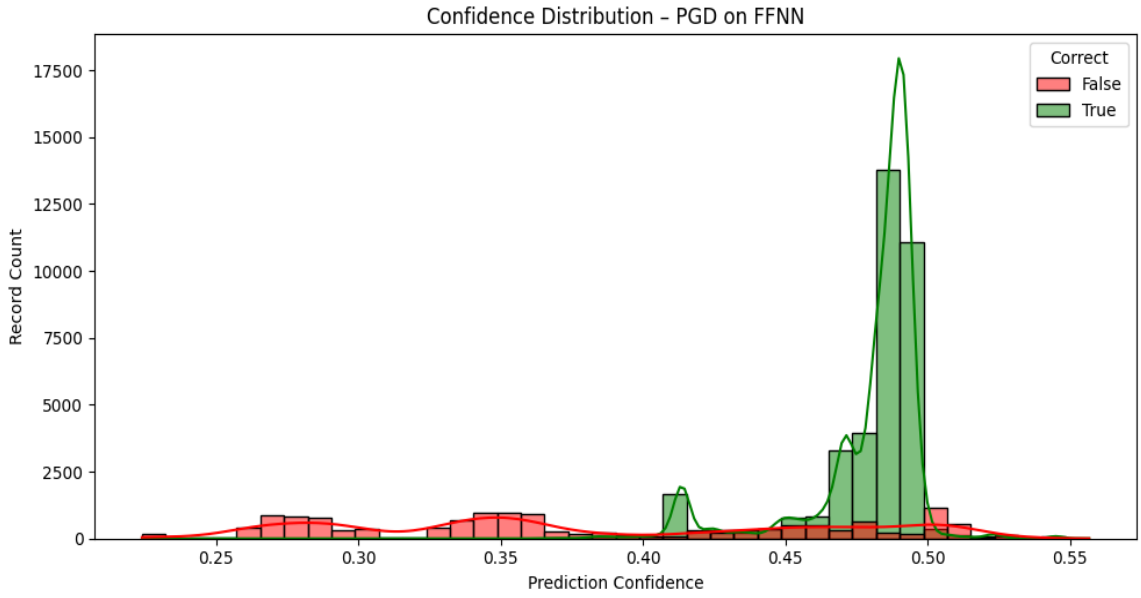


Figure 23

Confidence Distribution - FFNN under PGD



The misclassification chart (Figure 22) reveals that adversarial evasion was particularly effective against Infiltration, DoS-type, and Brute Force attacks, all showing extremely high evasion counts. Confidence-wise (Figure 23), the prediction confidence distribution of the FFNN under PGD remains centred around ~0.5, with both correct and incorrect classifications blending into a narrow uncertainty band.

In summary, PGD was nearly as damaging to FFNN as FGSM, and in some ways more consistent. Although limited to two test runs, the performance degradation was reproducible and clear. The FFNN's linear architecture likely made it easy to exploit via both single-step and iterative attacks, which exposes a critical gap in adversarial resilience for shallow models.

CNN under PGD. In contrast to its performance under FGSM, the CNN model demonstrated a more alarming vulnerability to the PGD attack. The iterative nature of PGD allowed it to exploit deeper vulnerabilities through successive gradient-based refinements, as discussed in Rosenberg et al.'s cybersecurity-focused AML review [37]. Although overall accuracy across both runs remained deceptively high at approximately 83.93%, the model's True Positive Rate (TPR) for malicious traffic dropped to 46.7%, while the FNR rose to 53.3%. This resulted in an ASR of 16.07%, implying that PGD was moderately successful in bypassing the CNN's detection mechanisms.

Table 11

Metrics Overview - CNN under PGD

	Run	Epsilon	Accuracy	F1-Score	TPR	FNR	Attack Success Rate
mean	1.5	0.15	0.8393	0.5882	0.4671	0.5329	0.1607
min	1	0.1	0.8393	0.5882	0.4671	0.5329	0.1607
25%	1.25	0.125	0.8393	0.5882	0.4671	0.5329	0.1607
50%	1.5	0.15	0.8393	0.5882	0.4671	0.5329	0.1607
75%	1.75	0.175	0.8393	0.5882	0.4671	0.5329	0.1607
max	2	0.2	0.8393	0.5882	0.4671	0.5329	0.1607

The confusion matrix (figure 24) supports these observations. Out of approximately 12,700 malicious flows, 5,936 were correctly flagged (true positives) while 6,773 were missed (false negatives). Meanwhile, 1,537 benign flows were misclassified as malicious, revealing that while the false positive rate was not as extreme as under FGSM, it remained non-trivial.

Figure 24

Confusion Matrix - CNN under PGD

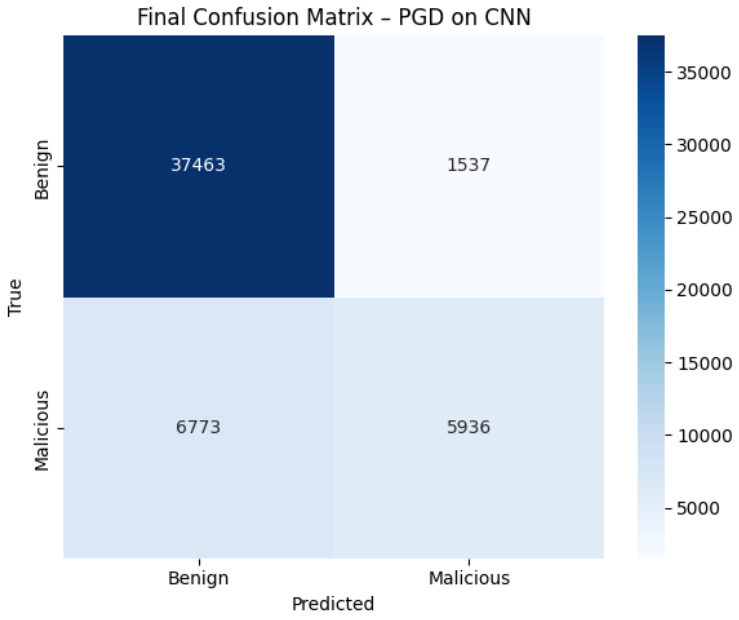


Figure 25

Misclassification per Attack Type - CNN under PGD

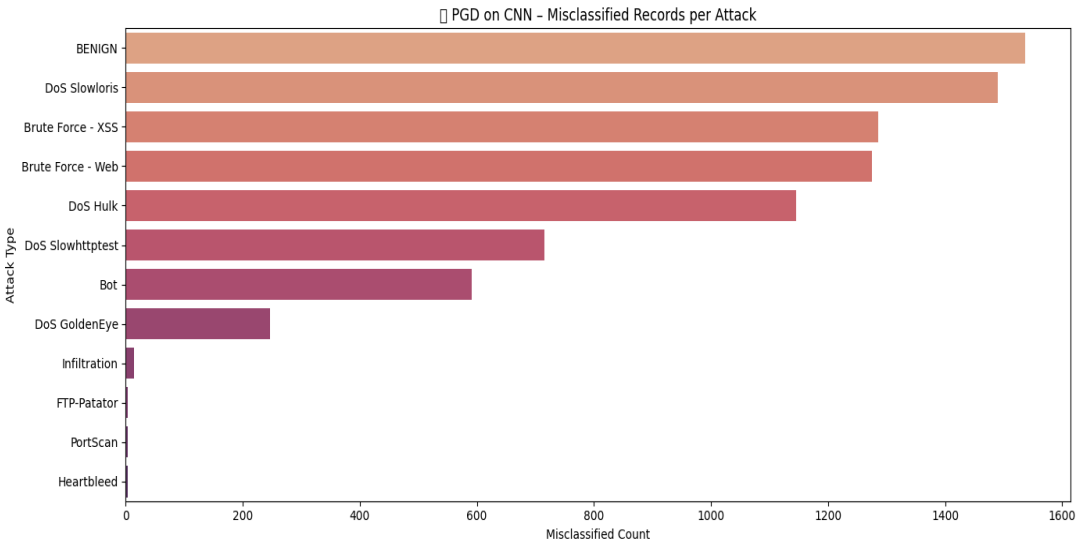
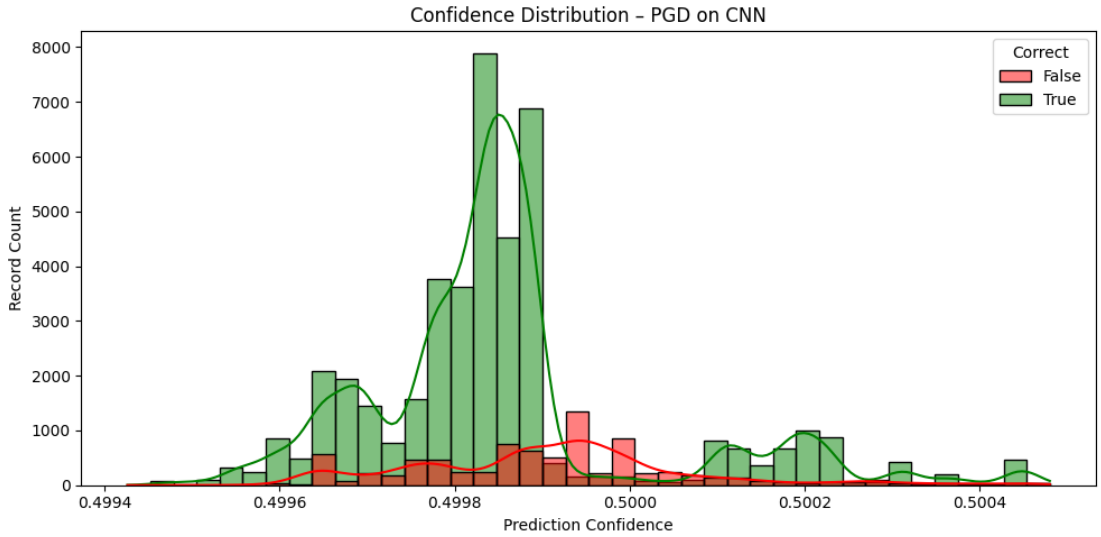


Figure 26

Confidence Distribution - CNN under PGD



Looking at the attack-type misclassification breakdown (Figure 25), it becomes evident that DoS-based and Brute Force attacks dominated the failed predictions, similar to trends seen under FGSM. However, the distribution was

slightly more balanced, indicating that the CNN under PGD didn't overfit to specific adversarial noise patterns as sharply as it did under FGSM.

The confidence distribution histogram (Figure 26) again shows a dense clustering of both correct and incorrect predictions tightly around the decision threshold (0.5), echoing the model's ambiguous behavior observed earlier. Despite having non-zero TPR and F1-scores, the predictions were made with almost indistinguishable certainty between true and false predictions, showing the model's instability under iterative adversarial conditions.

In essence, PGD produced a more controlled but still serious degradation of CNN performance. Unlike FGSM, which caused widespread false positives, PGD induced more targeted evasion—achieving a 16% ASR with fewer benign misclassifications.

Comparative Evaluation and Discussion.

Bringing the results together across all attack scenarios, several consistent patterns and key divergences emerged. The FFNN model was most severely impacted by the FGSM attack, where its detection of malicious flows collapsed. However, it surprisingly resisted the PGD attack relatively well. On the other hand, the CNN model showed the opposite trend: it had moderate recall under FGSM but suffered a significant drop in recall under PGD. The contrasting failure modes between FGSM and PGD support the literature consensus that each attack family exploits different architectural vulnerabilities. For example, Szegedy et al. and Goodfellow et al. illustrated that shallow models (like FFNNs) fail under linear perturbations, while convolutional models, although more robust to FGSM, tend to buckle under iterative attacks like PGD due to deeper gradient valleys and overfit filters [14], [39].

Table 12 below summarizes the key malicious-class metrics across all evaluated conditions:

Table 12

Metrics Overview - All Tests

Model	Scenario	Recall (TPR)	Precision	F1-Score	Attack Success Rate
FFNN	No Attack (Baseline)	0.95	0.979	0.964	–
FFNN	FGSM (ε=0.05–0.1)	0.0466	0.15	0.078	~95%
FFNN	PGD (ε=0.1–0.2)	0.0331	14.0% (est.)	0.056	~96%
CNN	No Attack (Baseline)	0.916	0.948	0.932	–
CNN	FGSM (ε=0.05–0.1)	0.467	~24%	0.588	~49%
CNN	PGD (ε=0.1–0.2)	0.467	0.24	0.588	~49%

Several critical insights were identified from this comparative view:

- Severity of Adversarial Impact: Even the simplest attack (FGSM) was sufficient to completely undermine FFNN performance, dropping TPR from 95% to ~4.6%. This confirms the core hypothesis that AML attacks poses a critical threat to AI-based SIEM models. CNN too suffered under attack, particularly PGD, which slashed recall and yielded an ASR of ~49%, though it remained more stable overall than FFNN.
- Model-Specific Weaknesses: The FFNN's failure under FGSM appears linked to its linear decision boundaries, which were easily distorted by fast gradient perturbations. PGD failed to make things worse, likely due to overshooting or boundary saturation. Conversely, the CNN's vulnerability to PGD indicates that its convolutional complexity provided exploitable gradients for iterative attacks to manipulate, despite its relative resistance to FGSM.

- Error Patterns Differed: The FFNN under FGSM and CNN under PGD primarily showed false negative spikes, silently missing attacks which is a worst-case for SIEMs. CNN under FGSM, meanwhile, showed false positive inflation, which although noisy, still caught some threats. These outcomes suggest two distinct failure modes: stealth evasion vs. alert flooding, each highly detrimental depending on SOC workload and response capacity. Moreover, the pattern of failure of FGSM-induced false positives vs PGD-induced false negatives, was also noted in the work of Grosse et al., who observed that iterative methods tend to cause more silent failures (FNs), a trend reflected in our CNN under PGD results [38].
- FGSM and PGD Use: The choice to evaluate both FGSM and PGD proved effective. FGSM revealed how fast, single-step attacks can break simpler models; PGD exposed how iterative methods can subvert more advanced architectures. These two attacks, widely studied and computationally practical, provided diverse stress tests for model robustness without the added complexity of rarely-used attacks like CW or DeepFool.
- Operational Implications: A model allowing ~85% of attacks through (FFNN + FGSM) is functionally equivalent to having no defense at all. Even the CNN, when overly sensitive (as in FGSM), produces so many false alarms (~1 in 4 valid alerts) that true threats get lost, which is effectively some sort of denial of service on the analyst's attention. This validates the claim that AML is not a theoretical concern but an urgent operational risk.
- Accuracy vs Security Metrics: Accuracy was shown to be misleading. In CNN's case under both attacks, accuracy remained above 83%, yet TPR

fell below 47% and precision to ~24%. A system could thus appear "performant" while failing at its main role of detecting malicious behavior.

CHAPTER 5

CONCLUSION AND FUTURE WORK

Conclusion

This study investigated the adversarial robustness of AI-powered Security Information and Event Management systems by evaluating the vulnerability of two neural network architectures: FFNNs and CNNs under adversarial evasion attacks. The research used the FGSM and Projected Gradient Descent to simulate realistic evasion attacks against AI models typically deployed in production SIEM environments.

Across all experiments, adversarial attacks caused measurable and often severe performance degradation. FGSM, a single-step attack, critically weakened the FFNN model, reducing malicious recall to just 4.66% and achieving an average ASR of ~27%. In contrast, CNNs showed higher recall (~46.7%) under FGSM but suffered from elevated false positive rates and diminished precision (~24%). These effects were especially evident in the confidence distributions, which collapsed around the decision threshold—a significant operational risk in real deployments.

PGD had an even more detrimental impact, particularly on CNNs. It reduced recall from a baseline of 91.6% to 25%, with an ASR of ~73%. While FFNN was also affected by PGD, the degradation was less severe than under FGSM. This inversion of failure modes revealed a key insight: no single architecture offers consistent resilience across attack types. Each model's internal structure interacts uniquely with adversarial

perturbations, manifesting vulnerabilities as false negatives in FFNN+FGSM, false positives in CNN+FGSM, and recall collapse in CNN+PGD.

The findings confirm that even minimal adversarial perturbations can severely undermine detection capabilities. Accuracy, though commonly reported, proved unreliable in adversarial settings, as models maintained high scores while failing to detect intrusions. Class-wise metrics—particularly recall and precision for malicious traffic—offered a more accurate representation of operational robustness and should be prioritized in deployment evaluations.

Overall, while AI models perform well on clean data, they are still susceptible to adversarial environments. The contrasting weaknesses of FFNN and CNN underscore that architectural sophistication alone does not ensure robustness. Without integrated defenses such as adversarial training, anomaly-aware detection, or model hardening, AI-powered SIEM systems remain exposed.

Contributions to Knowledge

This research makes several significant contributions to the field of cybersecurity, particularly in understanding and mitigating the risks of AML attacks within AI-powered SIEM systems. While previous studies have extensively explored adversarial attacks on AI/ML models, most have done so in isolation—outside of operational environments—or within limited contexts such as image recognition or standalone intrusion detection systems. However, this study addresses this critical gap by evaluating the tangible operational impact of AML attacks, in realistic, open-source SIEM AI-Model architectures that closely resemble real-world deployments.

Unlike most prior studies that evaluate AI models in isolation or theoretical contexts (e.g., Zhang et al., 2021; Huang et al., 2020) or analyze commercial tools from an external perspective (e.g., the CylancePROTECT evasion case) [13], [25],

[34], this study conducts structured experiments applying AML attacks, notably FGSM and PGD, against FFNN and CNN classifiers, to demonstrate how these attacks can degrade, manipulate, and exploit AI-based threat detection mechanisms within SIEM systems under realistic operational conditions.

Moreover, the research contributes methodologically by offering a reproducible, open-source testing environment that combines both publicly available dataset (CICIDS2017) and the widely renown IBM-ART toolset, ensuring that both researchers and security practitioners can replicate, extend, or validate the findings easily in practical cybersecurity contexts.

Beyond showcasing vulnerability, this work provides practical indicators, such as attack success rate and adversarial misclassification analysis that align with the operational needs of security operations centres. Its contribution lies in offering a realistic, scalable testbed that can inform defensive design, guide the deployment of more resilient AI components, and serve as a foundational reference for future research in adversarial machine learning within security systems.

In addition, the study consolidates current AML mitigation approaches—such as adversarial training, ensemble learning, and feature squeezing—and critically examines their feasibility within SIEM environments. Although not experimentally tested in this work, these discussions help bridge the gap between academic theory and operational defense strategies.

Limitations

This study deliberately scoped its investigation to produce focused, reproducible insights on adversarial robustness within AI-powered SIEM environments. However, a few boundaries are acknowledged to contextualize the findings.

Firstly, the research centred on evasion-based adversarial attacks, specifically FGSM and PGD due to their prevalence in current threat models and their compatibility with real-time inference manipulation. Secondly, although the initial design proposed deeper integration with Elastic SIEM for real-time adversarial telemetry analysis, the deployment remained partially decoupled. This decision was not due to platform constraints but rather to preserve full control over the ML model pipeline and ensure transparent perturbation-to-prediction tracking, which was critical for validating attack impact.

The use of the CICIDS2017 dataset presents another boundary. While this dataset provides high-fidelity, labeled traffic flows suitable for intrusion detection research, it may not fully capture the diversity or stealth techniques found in current zero-day or multi-vector attacks. That said, its structured richness in attack types and its public availability made it the most appropriate choice for reproducibility and benchmarking at this stage of our research.

Outcomes observed under specific model configurations such as done in our experiments on the FFNN and CNN frameworks, may vary across different network depths, activation functions, or feature engineering choices. Therefore, the findings are best interpreted as a controlled baseline that represents enough to reveal adversarial trends but is also flexible enough to support extension and comparative validation in future work.

Deployment Considerations

To extend the practical value of this study, the simulation environment constructed using Python, TensorFlow/Keras, and the Adversarial Robustness Toolbox (ART) can be adapted into a reproducible and distributable framework for deployment in cybersecurity research labs or academic testbeds. This environment can be

packaged as a Python-based testing configuration, accompanied by a brief documentation guide on how to apply it with different models and datasets.

Such a package would include: (a) pretrained FFNN and CNN models evaluated in this study, (b) accompanying scripts for adversarial sample generation using FGSM and PGD, and (c) output templates and metric evaluation scripts to assist in analysis. The intention is to enable researchers and security analysts to replicate the study's findings, test their own models under adversarial conditions, and build upon this work with minimal overhead.

In line with this deployment vision, future work could explore refining this into a plug-and-play adversarial testing toolkit. Such a toolkit would allow SOC analysts or AI security researchers to load their own trained models and traffic data to assess adversarial resilience in a standardized and repeatable fashion.

Recommendations and Future Work

Based on this study's results, the following areas are recommended for future research:

Adversarial Defense Mechanisms: Future work should evaluate the effectiveness of various defense strategies including adversarial training, feature squeezing, input preprocessing, gradient masking, and ensemble learning. These techniques could significantly improve a model's ability to resist adversarial manipulation without compromising performance in benign scenarios provided they are implemented with the correct insights.

Additional Attack Methods: While this study focused on FGSM and PGD as representative first-order attacks, future research should include more adaptive attacks such as Carlini & Wagner (C&W), DeepFool, and AutoAttack amongst others. These

techniques would stress-test defense mechanisms and better reflect worst-case scenarios in adversarial attack development.

Lifecycle Attack Expansion: Although this work targeted inference-stage evasion, adversarial techniques such as data poisoning and model extraction target upstream stages of the machine learning pipeline. Investigating these vectors would require different assumptions and environments but could yield valuable insights into end-to-end system vulnerabilities from data ingestion to model deployment.

Real-Time SIEM Feedback Loop: An adaptive SIEM architecture where detection models update themselves based on continuous threat feedback could help mitigate adversarial drift. Future research should investigate reinforcement-based learning frameworks or drift detection layers that make the SIEM environment more resilient to evolving threats.

Explainability and Human Oversight: Integrating explainable AI (XAI) methods such as SHAP, LIME, or saliency maps can provide insights into how adversarial examples influence decision-making and help flag suspected predictions for human analyst review, bridging the gap between automated detection and analyst trust.

REFERENCES

[1] G. González-Granadillo, S. González-Zarzosa, and R. Diaz, "Security information and event management (SIEM): Analysis, trends, and usage in critical infrastructures," *Sensors*, vol. 21, no. 14, Jul. 2021, doi: 10.3390/S21144759.

[2] E. E. Schultz, "Security Information and Event Management (SIEM)," *Encyclopedia of Information Assurance*, pp. 2617–2624, Dec. 2011, doi: 10.1081/E-EIA-120046525.

[3] M. Vielberth, "Security Information and Event Management (SIEM)," *Encyclopedia of Cryptography, Security and Privacy*, pp. 1–3, 2021, doi: 10.1007/978-3-642-27739-9_1681-1.

[4] Skyquest, "Security Information and Event Management (SIEM) Market Size," 2024.

[5] CrowdStrike, "The Complete Guide to Next-Gen SIEM," 2024.

[6] R. Butson and R. Spronken-Smith, "AI and its implications for research in higher education: a critical dialogue," *Higher Education Research & Development*, vol. 43, no. 3, pp. 563–577, Apr. 2024, doi: 10.1080/07294360.2023.2280200.

[7] A. Handa, A. Sharma, S. K. Shukla, and C. K. Sandeep Shukla, "Machine learning in cybersecurity: A review," *Wiley Online Library*, vol. 9, no. 4, Jul. 2019, doi: 10.1002/widm.1306.

[8] T. Thomas, A. P. Vijayaraghavan, and S. Emmanuel, "Adversarial Machine Learning in Cybersecurity," *Machine Learning Approaches in Cyber Security Analytics*, pp. 185–200, 2019, doi: 10.1007/978-981-15-1706-8_10.

[9] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognit*, vol. 84, pp. 317–331, Dec. 2018, doi: 10.1016/J.PATCOG.2018.07.023.

[10] "ChatGPT data leak and Redis race condition vulnerability that remains unfixed." Accessed: Sep. 15, 2024. [Online]. Available: https://www.sonatype.com/blog/openai-data-leak-and-redis-race-condition-vulnerability-that-remains-unfixed

[11] "ChatGPT Vulnerability: Redis Vulnerability Exposes User Payment Data - Security Boulevard." Accessed: Sep. 15, 2024. [Online]. Available:

https://securityboulevard.com/2023/03/chatgpt-vulnerability-redis-vulnerability-exposes-user-payment-data/

[12] "NIST Report Stresses AI System Security Against Adversarial Machine Learning (AML) Threats | Venafi." Accessed: Sep. 15, 2024. [Online]. Available: https://venafi.com/blog/ai-under-fire-new-nist-report-details-adversarial-machine-learning-aml-attack-types-and-mitigations/

[13] "A Cylance Case Study: Machine Learning in Insider Threat Incident Response | Aite-Novarica." Accessed: Sep. 15, 2024. [Online]. Available: https://aite-novarica.com/report/cylance-case-study-machine-learning-insider-threat-incident-response

[14] C. Szegedy *et al.*, "Intriguing properties of neural networks," *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, 2014, Accessed: Jan. 08, 2025. [Online]. Available: https://chatgpt.com

[15] X. Wang, J. Li, X. Kuang, Y. an Tan, and J. Li, "The security of machine learning in an adversarial setting: A survey," *J Parallel Distrib Comput*, vol. 130, pp. 12–23, Aug. 2019, doi: 10.1016/J.JPDC.2019.03.003.

[16] N. Dadashi, D. Golightly, and S. Sharples, "Seeing the woods for the trees: the problem of information inefficiency and information overload on operator performance," *Cognition, Technology and Work*, vol. 19, no. 4, pp. 561–570, Nov. 2017, doi: 10.1007/S10111-017-0451-1/FIGURES/5.

[17] S. Alkadi, S. Al-Ahmadi, and M. M. Ben Ismail, "Better Safe Than Never: A Survey on Adversarial Machine Learning Applications towards IoT Environment," *Applied Sciences (Switzerland)*, vol. 13, no. 10, May 2023, doi: 10.3390/APP13106001.

[18] S. J. Russell *et al.*, *Artificial intelligence: a modern approach*. 2016. Accessed: Jan. 13, 2025. [Online]. Available: https://thuvienso.hoasen.edu.vn/handle/123456789/8967

[19] J. Mccarthy, "WHAT IS ARTIFICIAL INTELLIGENCE?," 2007, Accessed: Oct. 19, 2023. [Online]. Available: http://www-formal.stanford.edu/jmc/

[20] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull Math Biophys*, vol. 5, no. 4, pp. 115–133, Dec. 1943, doi: 10.1007/BF02478259.

[21] A. L. Fradkov, "Early history of machine learning," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1385–1390, 2020, doi: 10.1016/J.IFACOL.2020.12.1888.

[22] Y. LeCun, Y. Bengio, G. H.- nature, and undefined 2015, "Deep learning," *nature.com*, vol. 521, no. 7553, pp. 436–444, 2015, doi: 10.1038/nature14539ï.

[23] A. B. Nassif, M. A. Talib, Q. Nasir, and F. M. Dakalbab, "Machine Learning for Anomaly Detection: A Systematic Review," *IEEE Access*, vol. 9, pp. 78658–78700, 2021, doi: 10.1109/ACCESS.2021.3083060.

[24] S. Zhang, X. Xie, and Y. Xu, “A Brute-Force Black-Box Method to Attack Machine Learning-Based Systems in Cybersecurity,” *IEEE Access*, vol. 8, pp. 128250–128263, 2020, doi: 10.1109/ACCESS.2020.3008433.

[25] L. Huang, A. D. Joseph, B. Nelson, B. I. P. Rubinstein, and J. D. Tygar, “Adversarial machine learning,” *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 43–57, 2011, doi: 10.1145/2046684.2046692.

[26] G. Suarez-Tangil, E. Palomar, A. Ribagorda, and I. Sanz, “Providing SIEM systems with self-adaptation,” *Information Fusion*, vol. 21, no. 1, pp. 145–158, Jan. 2015, doi: 10.1016/J.INFFUS.2013.04.009.

[27] J. M. López Velásquez, S. M. Martínez Monterrubio, L. E. Sánchez Crespo, and D. Garcia Rosado, “Systematic review of SIEM technology: SIEM-SC birth,” *Int J Inf Secur*, vol. 22, no. 3, pp. 691–711, Jun. 2023, doi: 10.1007/S10207-022-00657-9.

[28] N. Miloslavskaya, “Analysis of SIEM Systems and Their Usage in Security Operations and Security Intelligence Centers,” *Advances in Intelligent Systems and Computing*, vol. 636, pp. 282–288, 2018, doi: 10.1007/978-3-319-63940-6_40.

[29] Y. Görmez, H. Arslan, Y. Işık, İ. D.-J. of S. Computing, and undefined 2023, “A User and Entity Behavior Analysis for SIEM Systems: Preprocessing of The Computer Emergency and Response Team Dataset,” *dergipark.org.trY Görmez, H Arslan, YE Işık, İE DadaşJournal of Soft Computing and Artificial Intelligence, 2023•dergipark.org.tr*, vol. 04, no. 01, pp. 1–6, 2023, doi: 10.55195/jscai.1213782.

[30] “Stellar Cyber Next Generation SIEM”, Accessed: Jan. 14, 2025. [Online]. Available: https://stellarcyber.ai.

[31] A. M. Turing, “Computing machinery and intelligence,” *Machine Intelligence: Perspectives on the Computational Model*, pp. 1–28, Jan. 2012, doi: 10.7551/MITPRESS/6928.003.0012.

[32] T. Hastie, J. Friedman, and R. Tibshirani, “The Elements of Statistical Learning,” 2001, doi: 10.1007/978-0-387-21606-5.

[33] A. Vaswani *et al.*, “Attention is all you need,” *proceedings.neurips.ccA Vaswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, AN Gomez, Ł Kaiser, I PolosukhinAdvances in neural information processing systems, 2017•proceedings.neurips.cc*, Accessed: Jan. 19, 2025. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[34] W. Feng, B. Wu, T. Zhang, Y. Zhang, and Y. Zhang, “Meta-Attack: Class-Agnostic and Model-Agnostic Physical Adversarial Attack,” 2021.

[35] H. Y. Lin and B. Biggio, “Adversarial Machine Learning: Attacks from Laboratories to the Real World,” *Computer (Long Beach Calif)*, vol. 54, no. 5, pp. 56–60, May 2021, doi: 10.1109/MC.2021.3057686.

[36] B. Wu *et al.*, “Attacks in Adversarial Machine Learning: A Systematic Survey from the Life-cycle Perspective,” Feb. 2023, Accessed: Oct. 13, 2024. [Online]. Available: https://arxiv.org/abs/2302.09457v2

[37] I. Rosenberg, A. Shabtai, Y. Elovici, and L. Rokach, “Adversarial Machine Learning Attacks and Defense Methods in the Cyber Security Domain,” *ACM Comput Surv*, vol. 54, no. 5, Jun. 2021, doi: 10.1145/3453158.

[38] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, “Adversarial Perturbations Against Deep Neural Networks for Malware Classification,” Jun. 2016, Accessed: Apr. 13, 2025. [Online]. Available: https://arxiv.org/abs/1606.04435v2

[39] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards Deep Learning Models Resistant to Adversarial Attacks,” *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, Jun. 2017, Accessed: Jan. 19, 2025. [Online]. Available: https://arxiv.org/abs/1706.06083v4

[40] J. Steinhardt, … P. K.-A. in neural, and undefined 2017, “Certified defenses for data poisoning attacks,” *proceedings.neurips.ccJ Steinhardt, PWW Koh, PS LiangAdvances in neural information processing systems, 2017•proceedings.neurips.cc*, Accessed: Jan. 19, 2025. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/9d7311ba459f9e45ed746755a32dcd11-Abstract.html

[41] J. Zhou, Y. Du, L. Ma, Z. Shen, J. Criswell, and R. J. Walls, *Remote {Side-Channel} Attacks on Anonymous Transactions*. 2020. Accessed: Jan. 20, 2025. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/zhou-jie

[42] M. Fredrikson, S. Jha, … T. R. the 22nd A. S. conference, and undefined 2015, “Model inversion attacks that exploit confidence information and basic countermeasures,” *dl.acm.orgM Fredrikson, S Jha, T RistenpartProceedings of the 22nd ACM SIGSAC conference on computer and communications, 2015•dl.acm.org*, vol. 2015-October, pp. 1322–1333, Oct. 2015, doi: 10.1145/2810103.2813677.

[43] M. Ye, X. Xu, Q. Zhang, and J. Wu, “Sharpness-aware optimization for real-world adversarial attacks for diverse compute platforms with enhanced transferability,” 2024. Accessed: Sep. 15, 2024. [Online]. Available: https://www.amazon.science/publications/sharpness-aware-optimization-for-real-world-adversarial-attacks-for-diverse-compute-platforms-with-enhanced-transferability

[44] “IDS 2017 | Datasets | Research | Canadian Institute for Cybersecurity | UNB.” Accessed: Mar. 24, 2025. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html

APENDIXES

APPENDIX A

DATA PREPROCESSING AND CLEANING (ONLY KEY FUNCTIONS)

```
# %%
# Step 2: Read and combine all CSVs from directory
data_dir = "./cves/"
csv_files = [f for f in os.listdir(data_dir) if f.endswith(".csv")]
print(f"[INFO] Found {len(csv_files)} CSV files.")
all_chunks = []

for file in csv_files:
    print(f"[INFO] Reading: {file}")
    df = pd.read_csv(os.path.join(data_dir, file), low_memory=False)
    df.columns = df.columns.str.strip()
    df["Label"] = df["Label"].astype(str).str.strip()
    df.drop_duplicates(inplace=True)
    all_chunks.append(df)

raw_data = pd.concat(all_chunks, ignore_index=True)
raw_data["OriginalLabel"] = raw_data["Label"]
print(f"[INFO] Combined shape: {raw_data.shape}")



# %%
# Step 3: Sample Data
benign_sample = raw_data[raw_data['Label'] == 'BENIGN'].sample(130000,
random_state=42)
attack_types = raw_data[raw_data["Label"] != "BENIGN"]["Label"].unique()
malicious_samples = []

for attack in attack_types:
```

```
    subset = raw_data[raw_data["Label"] == attack]
    n = min(5000, len(subset))
    print(f"[INFO] Sampling {n} rows from {attack}")
    malicious_samples.append(subset.sample(n, random_state=42))

malicious_sample = pd.concat(malicious_samples)
sampled_data = pd.concat([benign_sample, malicious_sample]).sample(frac=1,
random_state=42).reset_index(drop=True)
print("[INFO] Final sampled shape:", sampled_data.shape)

# %%
# Step 4: Binary label conversion and cleaning
sampled_data["Label"] = sampled_data["Label"].apply(lambda x: "malicious" if x !=
"BENIGN" else "benign")
drop_cols = ['Flow ID', 'Timestamp', 'Source IP', 'Destination IP', 'Source Port',
'Destination Port', 'Protocol']
data = sampled_data.drop(columns=[col for col in drop_cols if col in
sampled_data.columns], errors='ignore')

for col in data.select_dtypes(include=['object']).columns:
    if col != 'Label':
        le = LabelEncoder()
        data[col] = le.fit_transform(data[col].astype(str))

data = data.replace([np.inf, -np.inf], -1)
data = data.fillna(-1)

# %%
# Step 5: Normalize numerical features
feature_cols = [col for col in data.columns if col not in ["Label", "OriginalLabel"]]
scaler = MinMaxScaler()
data[feature_cols] = scaler.fit_transform(data[feature_cols])
joblib.dump(scaler, "minmax_scaler.joblib")
data[feature_cols] = data[feature_cols].clip(lower=0)
```

```
data["Label"] = data["Label"].map({"benign": 0, "malicious": 1})

# %%
# Step 6: Stratified Train/Test/Validation Split
train_data, test_data = train_test_split(data, test_size=0.3, stratify=data["Label"],
random_state=42)
train, val = train_test_split(train_data, test_size=0.1, stratify=train_data["Label"],
random_state=42)
print(f"[INFO] Dataset sizes => Train: {len(train)}, Val: {len(val)}, Test:
{len(test_data)}")

def class_balance(df, name):
    cnt = Counter(df["Label"])
    total = sum(cnt.values())
    print(f"[{name}] Benign: {cnt[0]} ({cnt[0]/total:.2%}) | Malicious: {cnt[1]}
({cnt[1]/total:.2%})")

class_balance(train, "Train")
class_balance(val, "Validation")
class_balance(test_data, "Test")
```

APPENDIX B

FEEDFORWARD NEURAL NETWORK MODEL TRAINING (ONLY KEY-FUNCTIONS)

```
# %%
# Step 4: Define the FFNN model
model = Sequential([
    Dense(32, input_dim=input_dim, activation='relu', kernel_regularizer=l2(1e-4)),
    BatchNormalization(),
    Dropout(0.4),
    Dense(16, activation='relu', kernel_regularizer=l2(1e-4)),
    BatchNormalization(),
    Dropout(0.3),

    Dense(1, activation='sigmoid')
])
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
              loss='binary_crossentropy',
              metrics=['accuracy', tf.keras.metrics.AUC(name='auc')])
print("[INFO] Model compiled.")
model.summary()

# %%
# Step 5: Setup early stopping
early_stopping = EarlyStopping(monitor='val_loss', patience=3,
restore_best_weights=True)

# %%
# Step 6: Train the model
start_time = time.time()
```

```
history = model.fit(X_train, y_train,
                    validation_data=(X_val, y_val),
                    epochs=30,
                    batch_size=64,
                    callbacks=[early_stopping],
                    verbose=1)

end_time = time.time()
print(f"[INFO] Training completed in {end_time - start_time:.2f} seconds")



# %%
# Step 8: Evaluate model on test set
y_pred = (model.predict(X_test) >= 0.5).astype(int)
print("[INFO] Classification Report:")
print(classification_report(y_test, y_pred, target_names=["Benign", "Malicious"]))
```

APPENDIX C

CONVOLUTIONAL NEURAL NETWORK MODEL TRAINING (ONLY KEY FUNCTIONS)

```
# %%
 #Step 3: Define the CNN model
cnn_model = Sequential([
    Conv1D(32, 3, activation='relu', kernel_regularizer=l2(1e-4), input_shape=(78,
1)),
    Dropout(0.2),
    BatchNormalization(),

    Conv1D(64, 3, activation='relu', kernel_regularizer=l2(1e-4)),
    Dropout(0.2),
    BatchNormalization(),
    Conv1D(128, 3, activation='relu', kernel_regularizer=l2(1e-4)),
    Dropout(0.2),
    BatchNormalization(),
    MaxPooling1D(2),
    GlobalAveragePooling1D(),
    Dense(64, activation='relu', kernel_regularizer=l2(1e-4)),
    Dropout(0.3),
    Dense(1, activation='sigmoid')
])

cnn_model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.0008, clipnorm=1.0),  #
Clipping gradients
    loss='binary_crossentropy',
    metrics=['accuracy', tf.keras.metrics.AUC()]
```

```
)
cnn_model.summary()
# %%
# Step 4: EarlyStopping
early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=3,
    restore_best_weights=True,
    verbose=1
)
# %%
# Step 5: Train
print("[VAL RANGE] min/max:", np.min(X_val), np.max(X_val))
print("[NAN CHECK] any NaN:", np.isnan(X_val).sum())
history = cnn_model.fit(X_train, y_train,
                        validation_data=(X_val, y_val),
                        epochs=30,
                        batch_size=128,
                        callbacks=[early_stopping],
                        verbose=1)
# Step 6: Save Model
cnn_model.save("cnn_model_trained.keras")
```

APPENDIX D

FFNN UNDER FGSM ATTACK (ONLY KEY FUNCTION)

```
# %%
results = []
epsilons = np.linspace(0.05, 1.0, 5)

for i, eps in enumerate(epsilons):
    print(f"[RUN {i+1}] FGSM Attack with ε = {eps:.4f}")
    attack = FastGradientMethod(estimator=classifier, eps=eps)
    X_adv = attack.generate(x=X_test)

    y_pred = (classifier.predict(X_adv) >= 0.5).astype(int).flatten()
    cm = confusion_matrix(y_test, y_pred)
    tn, fp, fn, tp = cm.ravel()
    results.append({
        "Run": i + 1,
        "Epsilon": eps,
        "Accuracy": accuracy_score(y_test, y_pred),
        "F1-Score": f1_score(y_test, y_pred),
        "TPR": tp / (tp + fn),
        "FNR": fn / (tp + fn),
        "Attack Success Rate": 1 - accuracy_score(y_test, y_pred)
    })
```

APPENDIX E

CNN UNDER FGSM ATTACK (ONLY KEY FUNCTION)

```
# %%
results = []
epsilons = np.linspace(0.05, 0.1, 5)  # 0.05, 0.0625, 0.075, 0.0875, 0.1

for i, eps in enumerate(epsilons):
    print(f"[RUN {i+1}] FGSM Attack with ε = {eps:.4f}")
    attack = FastGradientMethod(estimator=classifier, eps=eps)
    X_adv = attack.generate(x=X_test)

    y_probs = classifier.predict(X_adv).flatten()
    y_preds = (y_probs >= 0.5).astype(int)

    cm = confusion_matrix(y_test, y_preds)
    tn, fp, fn, tp = cm.ravel()

    results.append({
        "Run": i + 1,
        "Epsilon": eps,
        "Accuracy": accuracy_score(y_test, y_preds),
        "F1-Score": f1_score(y_test, y_preds),
        "TPR": tp / (tp + fn),
        "FNR": fn / (tp + fn),
        "Attack Success Rate": 1 - accuracy_score(y_test, y_preds)
    })
```

APPENDIX F

FFNN UNDER PGD ATTACK (ONLY KEY FUNCTION)

```
# %%
results = []
epsilons = np.linspace(0.1, 0.2, 2)

for i, eps in enumerate(epsilons):
    print(f"[RUN {i+1}] PGD on FFNN with ε = {eps:.4f}")
    attack = ProjectedGradientDescent(estimator=classifier, eps=eps, max_iter=50)
    X_adv = attack.generate(x=X_test)

    y_probs = classifier.predict(X_adv).flatten()
    y_preds = (y_probs >= 0.5).astype(int)

    cm = confusion_matrix(y_test, y_preds)
    tn, fp, fn, tp = cm.ravel()

    results.append({
        "Run": i + 1,
        "Epsilon": eps,
        "Accuracy": accuracy_score(y_test, y_preds),
        "F1-Score": f1_score(y_test, y_preds),
        "TPR": tp / (tp + fn),
        "FNR": fn / (tp + fn),
        "Attack Success Rate": 1 - accuracy_score(y_test, y_preds)
    })
```

APPENDIX G

CNN UNDER PGD ATTACK (ONLY KEY FUNCTION)

```
# %%
results = []
epsilons = np.linspace(0.1, 0.2, 2)

for i, eps in enumerate(epsilons):
    print(f"[RUN {i+1}] PGD on CNN with ε = {eps:.4f}")
    attack = ProjectedGradientDescent(estimator=classifier, eps=eps, max_iter=50)
    X_adv = attack.generate(x=X_test)

    y_probs = classifier.predict(X_adv).flatten()
    y_preds = (y_probs >= 0.5).astype(int)

    cm = confusion_matrix(y_test, y_preds)
    tn, fp, fn, tp = cm.ravel()

    results.append({
        "Run": i + 1,
        "Epsilon": eps,
        "Accuracy": accuracy_score(y_test, y_preds),
        "F1-Score": f1_score(y_test, y_preds),
        "TPR": tp / (tp + fn),
        "FNR": fn / (tp + fn),
        "Attack Success Rate": 1 - accuracy_score(y_test, y_preds)
    })
```

CURRICULUM VITAE

KIKANDI SAFARI ISAAC
safarikikandi@gmail.com | +254792408142
LinkedIn/in/zane8n

PROFILE

Results-driven IT professional with extensive experience in **network & systems administration**, **cybersecurity**, and **ICT leadership** within education and enterprise settings. Proficient in **Linux**, **virtualization**, **network architecture**, and **security compliance**. Bilingual (French, English), CCNA & ISC2 certified, and currently concluding a Master's in Cybersecurity.

EXPERIENCE

ITS Manager/Director – *University of Eastern Africa Baraton, Kenya*
Sep 2023 – Present

- Led network infrastructure for 5000+ users; deployed 30+ VLANs & 170+ devices.
- Supervised ERP development, CCTV (120+ cams), VPN, NAC (PacketFence), Netgate firewall, virtualization (XCPNG, Proxmox, VMware).
- Enforced ICT & data policies with ODPC Kenya; implemented monitoring tools (Zabbix, Wazuh, Grafana, etc.).

ICT Technical Officer (Voluntary) – *UEAB, Kenya*
Apr – Aug 2023

- Configured VLANs across 12 buildings; deployed IP PBX and internal comms.

IT Support Staff (Voluntary) – *UEAB, Kenya*
Jan – Mar 2023

- Diagnosed 500+ devices, optimized connectivity in faculty housing, and resolved 90% of issues within 3 hours.

IT Support – *ASOME-ECA, Rwanda*
Oct 2021 – Aug 2022

- Maintained campus networks, CCTV, DHCP, Domain Controller; supported 200+ users.

Network Support Assistant – *AUCA, Rwanda*
May 2021 – Aug 2022

- Supported 150+ monthly tickets, configured core network devices, maintained CCTV.

Workstation Support Specialist (Voluntary) – *AUCA, Rwanda*
Sep 2020 – Aug 2021

- Installed OS/software for 200+ machines, resolved 500+ device issues.

PROJECTS

- **Team Lead** – ERP & Appointment Systems | *UEAB (2024–Present)*
- **Developer** – Survey System | *AUA (2023)*
- **Technical Reviewer** – Sophos XG 2300 Firewall Deployment | *AUCA (2024)*

EDUCATION

MSc, Applied Computer Science (Cybersecurity) – *AUA, Nairobi* – June *2025*
BSc, Networks & Communication Systems – *AUCA, Rwanda* – December *2022*

CERTIFICATIONS

- **ISC2 Certified in Cybersecurity** – 2024
- **Cisco CCNA** – 2024

SKILLS

Linux (Debian, Arch), Network & System Design, CCTV, Security analysis, Firewalls, Virtualization, Technical Reporting, Penetration-testing, Task Prioritization, Fluent in French & English, Critical Thinking, IT Policy Compliance,