Threat Analysis of NGAVs, Insider Threats and Case Studies of Success and Failure of NGAVs

Rahul Ravi Hulli
M.E. Information Systems Security
Concordia University
Montreal, Quebec, Canada
r_hulli@mail.concordia.ca,
rahulravi.hulli@gmail.com

***Abstract*—Please consider the following indices to meet the milestones as defined below:**
1: The background and the problem to be solved in the paper: Securing ML Systems: Addressing Vulnerabilities and Defense Strategies, The New Advancements in NGAVs and ML Based Security.
2: The approach proposed by the paper to address the problem: Methodology for analyzing ml threats and mitigations – A: Data collection and processing (Data based AI tool), B: Host based Detection (Host and Heuristics based AI tool).
3: The performance of the proposed approach, as reported in the paper: There are no specific performance-based details in the paper, but there are two case studies which would help understand where AI based NGAVs fail.
4: Your critical review of the potential application, advantage, and limitation of the solution/approach: Conclusion and Case Study.

I. Introduction

Machine learning plays a crucial role in enhancing safety-critical domains and improving cybersecurity. It addresses complex problems in domains where safety is crucial. It also plays a critical role in determining and catching potentially damaging activities such as spam emails, malicious files, etc. ML (Machine Learning) is known to study real-time data and differentiate between normal and questionable patterns, which helps to protect computer systems and networks from menaces. It bolsters the defense mechanisms against cyber threats. It allows systems to reply to incoming data, including regular and potentially damaging data, and learn to determinate and intercept malicious behavior. However, ML also presents challenges associated to the adaptability of threat actors who can exploit ML models and data to their benefit. The threat actors can manipulate the black-box nature of the ML models. Black-box models are intricate and difficult to diagnose, making it hard to understand how they make judgments. The attackers can also acclimate and refine their methods to circumvent ML-based security measures. They can do this by discovering vulnerabilities in the ML models or carefully preparing input data to manipulate weaknesses.

Networked infrastructures that connect computers and devices across departments and networks are crucial to the day-to-day operations of governments, businesses, and organizations. These networked systems enable efficient operations by facilitating data accessibility and the sharing of computer resources. However, it is crucial to protect these infrastructures against insider threats, which are among the many cyberthreats.

II. Securing ML Systems: Addressing Vulnerabilities and Defense Strategies

It is important here to emphasize the growing significance of addressing security concerns of ML based systems, spanning various stages of the ML lifecycle and various layers of deployment. Take a glimpse over the ML lifecycle. The phases in this lifecycle include input data, training, interference (using the trained model to make predictions) and monitoring the deployed ML system. This broad approach is necessary for managing vulnerabilities at each stage of an ML-based system's operation. At the software stage, ML-based systems are exposed to attacks on the underlying operating system (OS). Attackers can manipulate vulnerabilities and bugs in the OS to compromise the entire system. ML-based systems are deployed on either on-site or on cloud service providers. Deploying on the cloud can expand the attack surface, making these systems vulnerable to various old-school attacks at multiple layers. Note that it is easier to evaluate the security features on how they reside on each layer of the OSI / TCP model. When we address the Network-Level vulnerabilities, some of the main concerns include the Denial of Service (DoS) attacks, botnets, and ransomware attacks, which can disrupt the availability and functionality of the ML system. ML-based systems can be attacked at the system level. One such example is the memory-based attack or CPU side-channel attacks. These attacks target the hardware and software interaction, potentially compromising the confidentiality and integrity of the ML system. ML Threat actors deploy tactics such as data poisoning, evasion, extraction, inference, and poising, which deceive or compromise the ML-based systems, and they require specific defense mechanisms to counteract. There is also a deficiency of concrete threat inspection applications in the ML field. There are fewer applications that have gone through more comprehensive and practical approaches to access and mitigate ML threats, vulnerabilities, and solutions. A threat assessment framework, TTP (Threat Tactics, techniques, and procedures) helps organizations and security professionals comprehend and respond to ML-related threats effectively.

III. The New Advancements in NGAVs and ML Based Security

Let us take a deeper dive into TTPs. The authors of this paper [1] have recognized new threat tactics, techniques, and procedures by pulling information from two important sources: the AI (Artificial Intelligence) incident database and pertinent literature. These sources supply insights into real-world incidents and recorded threats associated to AI and ML (Machine Learning). These newly identified TTPs intend to

complement the MITRE's ATLAS database - a widely used resource for comprehending adversarial tactics and techniques. We can map these recognized threat TTPs to various phases of the ML lifecycle, specific ML models, and the tools used in ML development and deployment. This mapping aids evaluate the potential influence of threats on various components of ML systems, making it easier to comprehend where vulnerabilities exist and how they might impact various aspects of ML projects. The authors have conducted a vulnerability investigation of ML repositories. This research concerns scrutinizing popular libraries and resources used in ML development to determine vulnerabilities. Also note that, this research [1] evaluates the dependencies associated with these ML libraries to foresee and prepare for potential threats arising from a particular ML library or its dependencies. This report [1] offers a multi-layer ML mitigation matrix, a structured framework created to manage and mitigate ML threats effectively. This matrix leverages security recommendation from various sources, including standard databases, such as those referenced in [2], [3], [4], as well as resources from the National Institute of Standards and Technology (NIST) and the Cloud Security Alliance.

A. Understanding ML Threats: A Targeted Investigation

The prior goal of this study is to understand threats to ML systems. To accomplish this goal, the researchers [1] piloted a practical investigation. They also characterize ML threats - define various threats that involve ML systems. This includes comprehending how these threats impact various segments of ML, including the steps of ML projects, ML models, and the tools employed in ML development. The study results demonstrate that deep neural networks (DNNs), particularly convolutional neural networks (CNNs), are often targeted in attack scenarios. This means that attackers usually focus their efforts on exploiting vulnerabilities within these specific models. ML repositories like TensorFlow, OpenCV and Notebook, as well as libraries like NumPy were found to have significant vulnerability reputation. Specific dependencies, such as pickle, jiblib, numpy116, python3.9.1 and log4j, were identified as pushing severe vulnerabilities. DoS attacks and buffer overflow attacks were the most typical types of threats observed in ML repositories. The study revealed that testing, inference, training, and data collection are the phases of ML projects that are most frequently targeted by threat actors. These are critical stages in the ML lifecycle and are susceptible to several types of attacks. The research presents several mitigation techniques to address these vulnerabilities and threats. These include 'adversarial defenses' and 'traditional defenses' such as software updates and cloud security patches (e.g., zero trust). The results have functional applications for ML red/blue teams, allowing them to leverage understanding of the recognized threats and tactics for executing effective attacks and defenses. ML practitioners can use these results to contain vulnerabilities and threats throughout the lifecycle of ML products. Researchers can build upon these findings to generate theories and algorithms to strengthen ML protection.

B. Understanding Insider Threats

A person who has privileged access to a company's network, systems, and data and deliberately uses that access to compromise the confidentiality, integrity, or availability of the company's information and ICT infrastructures is referred to as an insider threat in the CERT Coordination Centre's (CERT/CC) technical report [14]. Earlier, we called these nefarious insiders "traitors." Insider risks also include people who pose a threat to the organization by using "masqueraders" to act like someone else. Insider threats can also be caused by authorized users who accidentally commit crimes, sometimes known as "unintentional perpetrators." It is vital to remember that both intentional and unintentional insider acts depart from typical behavioral patterns, even though masqueraders may access ICT systems using compromised devices or credentials obtained from authorized users. This study recognizes the significance of identifying and mitigating any anomalous or harmful behavior within organizations' networked infrastructures by focusing on all sorts of insider threats without separating out intent.

1. **Categorization of Inside Threats:** Differentiating between various insider dangers, such as traitors, imposters, and other potential harmful insiders, is involved in this categorization. By doing this, we develop a formal framework for dealing with these challenges.
2. **Data Sources:** It states that a variety of data sources will be examined as part of the survey, all of which are crucial for developing successful anomaly-based intrusion detection systems (IDS). Host-based, network-based, and contextual data sources are the three basic types of data sources.
3. **Technical Perspective:** In technical viewpoint the poll will take a technical stance. This indicates that it will explore practical issues like data analytics methods and detection algorithms in addition to offering a theoretical perspective. This strategy is essential for closing the knowledge gap between academic research and practical cybersecurity procedures.
4. **Contributions:** Some of these contributions are described, including the equitable treatment of several types of insider threats and the introduction of the APT intrusion kill chain as a framework. The poll also recognizes the growing significance of contextual data-based analytics.

C. Assets and Vulnerabilities in Computer Security and ML

Let us clarify the concepts of assets and vulnerabilities in the world of computer security and ML systems. We would need to highlight that assets encompass a wide range of features critical to an organization, while vulnerabilities can be displayed at different levels and can be exploited by threat actors to compromise the security and integrity of these assets.

Definition of Assets: In computer security, an "asset" refers to any valuable logical, or physical object owned by an organization. Assets can include a wide range of items, both

tangible and digital. These are critical to an organization's operations and need protection.

Types of ML Assets: Assets are categorized into several levels. Each level has a significance:

Data Level: ML assets at the data level include access keys (such as tokens, usernames/passwords, private/public keys, and certificates), datasets, models, source code, and libraries.

Software Level: Assets at the software level contain elements like Model as a Service (MaaS) APIs, ML applications in production, containers, and virtual machines (VMs) used to deploy ML apps.

Storage Level: ML assets at the storage level contain databases, objects (e.g., buckets), files, and blocks used to store ML training datasets, models, and code.

System Level: Assets at the system level contain physical infrastructure like servers, racks, data centers, and clusters used to host ML applications.

Network Level: ML assets at the network level contain network devices like firewalls, routers, gateways, switches, and load balancers that secure ML systems.

Definition of Vulnerabilities: A "vulnerability" is a flaw, whether in software or hardware, that could be exploited by malicious actors (threat actors) to carry out dangerous actions, such as unauthorized access, data theft, or erasure of data or systems.

Types of Vulnerabilities: Vulnerabilities can manifest at different stages within an organization's infrastructure. This has been explained above. One easy way to look at it is through the OSI / TCP model:

Data Level: Vulnerabilities at the data level can lead to threats like model stealing, backdooring, poisoning, injection attacks, evasion, and inference attacks, which may compromise the integrity of ML data.

Software Level: Vulnerabilities at the software level include errors or bugs within ML applications, such as buffer overflows, exposed credentials, and security misconfigurations. Exploiting these can result in unauthorized access or control.

Storage Level: ML databases and cloud storage are susceptible to vulnerabilities like weak authentication, improper backup procedures, and SQL injection attacks, which can lead to data breaches.

System Level: Threat actors may target hardware vulnerabilities, such as firmware issues and CPU side-channel attacks, to launch attacks like Denial of Service (DoS), impacting the cloud infrastructure where ML applications run.

Network Level: Vulnerabilities can also exist at the network level due to inappropriate configurations, making networks vulnerable to distributed DoS attacks and botnet-based attacks.

Standards for Vulnerabilities: Various standards and databases, including Common Weaknesses Exposure (CWE) Top 25, OWASP Top 10, National Vulnerability Database (NVD), and Common Vulnerability and Exposures (CVE), are typically used to record and organize vulnerabilities, making it more effortless to understand and manage them systematically

D. Comprehensive Threat Analysis in Computer Security and ML

Let us develop a comprehensive understanding of threats. We will look at their types, phases, and techniques, in both – traditional computer security and the emerging field of machine learning security.

Definition of Threats: A "threat" is a possible danger or risk that exploits a vulnerability to cause destruction to a specific asset. These threats can be broadly categorized into two types: insider threats and outsider threats.

Insider Threats: Insider threats arise within the organization's inner systems, run by trusted entities, such as employees, who misuse their access privileges.

Outsider Threats: Outsider threats come from external sources and target the organization's assets from remote locations.

Traditional Threat Phases:

Pre-Attack Phase: Reconnaissance and resource development tactics. Here, attackers gather information about the target, such as open ports and OS versions, using techniques such as network scanning. Resource development includes formulating resources to support command and control (C2) operations, purchasing compromised systems (botnets), and developing tools for attacks.

Attack Phase: After the pre-attack phase, attackers try to gain initial access to the target system through various means, including delivering malicious files, exploiting vulnerabilities, and manipulating software dependencies. Once accessed, they execute malicious code, persist on the target system, escalate privileges, and hide their actions from security tools. They can move laterally to other systems within the network as well

Machine Learning Threats:

ML Pre-Attack Phase: ML pre-attack tactics are like traditional threats but adjusted to ML systems. Attackers conduct reconnaissance by searching for unrestricted research materials and ML artifacts. They may also obtain adversarial ML attack implementations.

ML Attack Phase: ML systems are vulnerable to old-school and adversarial attacks, where attackers aim to turn normal ML behaviors into malicious ones. These attacks can target confidentiality, integrity, and availability (CIA) of data. Attackers may have varying levels of knowledge about the target ML system, ranging from white-box (full knowledge) to black-box (no knowledge). Attack techniques include poisoning, evasion, extraction, and inference. These attacks exploit ML models' behavior or extract sensitive information from them.

Adversarial Models for ML Attacks: Many adversarial models used in ML attacks, such as the Fast Gradient Sign Method (FGSM), DeepFool, Carlini and Wagner (C&W), Jacobian-based Saliency Map (JSMA), Universal Adversarial Perturbations, Basic Iterative Method (BIM), One Pixel, Iterative Least-Likely Class Method (ILCM), and Adversarial Transformation Networks (ATNs). These methods generate adversarial inputs that can deceive ML models.

E. Threat assessment techniques to ML-Based Systems.

Threat assessment is a constant process in computer security that interests identifying, analyzing, evaluating, and

mitigating threats to an organization's systems and assets. There have been no concrete applications of this process to ML-based systems until recently.

The ATLAS framework is the first comprehensive real-world endeavor at ML threat inspection. It was created collaboratively by the MITRE Corporation with organizations like Microsoft and Palo Alto Networks [6]. This framework provides a structured method to evaluate threats. ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge) is a publicly available knowledge database that catalogs adversarial tactics, techniques, and procedures (TTPs). It has does threat assessment in both traditional and ML contexts. Kumar et al. [8] determined gaps in ML security during development, deployment, and when ML systems were under attack. They listed ML security aspects and proposed fixes, including those from ATT&CK, Cloud Security Alliance, and NIST security standards. Lakhdhar et al. [7] mapped recently discovered vulnerabilities to potential attack tactics in ATT&CK. They used features like CVSS stringency scores and vendor information to train a model. However, this approach was limited to the ATT&CK database and focused on mapping vulnerabilities to TTPs. Kuppa et al. [9] also mapped Common Vulnerabilities and Exposures (CVEs) to ATT&CK techniques utilizing a multi-head joint embedding neural network. Like Lakhdhar et al., their work was limited to mapping vulnerabilities to ATT&CK TTPs.

The research [1] leverages numerous sources, including ATLAS, ATT&CK, and the AI Incident Database, to deliver a complete set of TTP reports to describe ML threats. It also contains a vulnerability assessment approach to identify, analyze, evaluate, and mitigate vulnerabilities in ML-based systems. In addition to these databases, the research combines defense frameworks such as ATT&CK mitigations and MITRE D3FEND, as well as security guidelines from Cloud Security Alliance and NIST. This is used to conduct an end-to-end threat assessment of ML assets, providing a more comprehensive understanding of threats and vulnerabilities in the ML lifecycle. The research [1] aims to show how ATLAS TTPs impact various components of ML systems at different phases of the ML lifecycle. Additionally, it examines how traditional threats and vulnerabilities affect ML repositories.

F. Insider Threat Actions, Communication Channels, and Attack Types

Persistence and Communication Channels: Attackers contact compromised hosts (also known as "bots") via a Command and Control (C2) channel [15]. Attackers can instruct the compromised hosts using this channel, which is created by the Remote Access Trojans (RATs) or backdoors that have been deployed. **Organization of the botnet:** There are two paradigms, Client-Server (CS) and Peer-to-Peer (P2P) which are used for grouping hacked sites into botnets [16] [17]. To avoid detection in a CS model, botnets routinely modify the IP (Internet Protocol) address of their server utilizing dynamic Domain Name System (DDNS) services [18]. A P2P model-based botnet, on the other hand, is more durable since each node functions as both a bot master and a bot client [19]. **Destructive Attacks Using Botnets:** Botnets can be used to launch destructive attacks, such as DDoS attacks [20] [21] that overwhelm victims' networks with incoming traffic and email spam attacks that bombard victims' email systems with unwanted or malicious emails [22]. **Exploitation for Profitable Activities:** Attackers can make use of compromised hosts' resources for lucrative endeavors like click fraud and bitcoin mining [23] [24]. These actions are taken so that the attackers can profit financially. **Outside Attackers Changing into Malicious Insiders:** Attackers change into malicious insiders during the APT intrusion death chain's "actions on objectives" phase [25]. Without undergoing the whole infiltration effort, traitors and unintended criminals might offer the same hazards as harmful insiders.

Various forms of insider threats:
Exfiltration of Data: unlawful access to business data, unintended disclosure of sensitive data, and theft of intellectual property [28] [29] are just a few examples of behaviors that fall under the category of unlawful copying, transferring, or retrieving of data from a computer or server. **Data availability or integrity violations:** Changes to file extensions, the encryption and decryption of sensitive data, and file manipulation [30] are all examples of actions that might cause data to disappear or be damaged and prevent the availability of proper data copies. Ransomware is also mentioned as an example [31]. **Sabotage of ICT Systems:** An example of an insider using ICT to damage a person, or an organization would be to conduct a DDoS assault from within the company [32].

G. Analyzing Behavioral Biometrics: Command, Keyboard, and Mouse

Introduction to Behavioral Biometrics: This includes identifying and detecting masqueraders by exploiting a user's behavioral patterns. It highlights that for this purpose, *nix shell commands, keyboard keystrokes, and mouse dynamics are important sources of behavioral data. Various approaches employed in system call-based analytics may be repurposed for analyzing *nix shell commands. It should be noted, however, that these approaches are often used inside a user-profiling framework, in which user profiles are constructed based on past command sequences. For analysis, statistical and machine learning methods are often utilized. **Lane and Brodley's Early strategy :** It examines an early strategy devised by Lane and Brodley to detect masqueraders by assessing the similarity of command sequences. This system advanced beyond binary true/false metrics by introducing a numerical measure of similarity [46]. It also employed the least-recently-used (LRU) pruning approach and a greedy clustering algorithm to maximize data storage space. "Sequence Model" is used to describe strategies like the n-gram model but using a fixed-size window. Approaches that assess abnormality using the Hellinger distance and utilize one-class SVM to categorize anomalous command subsequences.

One-step transition matrices, high-order Markov chains, implicative mixture transition distribution (MTD), and other statistical methods are used to depict user profiles. Based on command uniqueness and popularity, several approaches

employ likelihood-ratio testing, Bayesian categorization, and simpler test statistics. **Keyboard and Mouse Dynamics:** Keyboard and mouse dynamics can potentially be viable data sources for behavioral biometrics analysis. It offers Ahmed and Traore's example approach for characterizing keyboard dynamics using dwell time and flight time, as well as mouse dynamics using variables such as average speed and movement direction. To detect discrepancies in these dynamics, an artificial neural network (ANN) is trained.

Threat type	Model	Tech category	Algorithm	Data source
Intrusion		statistical	ensemble [84]	Win performance monitor logs
Malware	n-gram	statistical	LR [85]	Win audit logs
Insider threats		statistical	GMM, KDE [58]	Win system logs
		machine learning	SVM [58]	

Table 4: Taxonomy of Host Log Based Analytics

IV. Host Logs for Analysing Host Behavior

Host logs are records of numerous events and messages that occur within an operating system or other software applications. These logs provide a wealth of information that may be used to audit and trace a host's activities. Host logs may be used to identify intrusions, malware, and malevolent insiders. Because of their complexity and redundancy, collecting usable information from host logs can be difficult. Host logs can include massive volumes of data, and sorting through this data to find valuable information can be time-consuming and difficult. There is no standard model for evaluating host logs, such as n-grams (sequence) or frequency analysis. Statistical and machine learning techniques, on the other hand, have discovered the greatest application cases in this field. An intrusion detection system that focuses on examining windows performance monitor logs is introduced [47]. This employs numerous detectors to handle individual characteristics derived from these logs. Each detector fits data with a distribution selected from a collection of candidates. To reach a final choice on intrusion detection, a weighted-majority voting procedure is used. To reduce false alarms, the detectors are gradually calibrated using a labeled data set. Another approach uses ordinary windows audit logs to identify malware. The data collector records file and registry actions, as well as process launching [48]. These events are categorized by process IDs, and each process is represented by n-grams. All the n-grams are combined into a feature vector representing the complete raw log. To detect malicious feature vectors, a logistic regression (LR) classifier is developed. A behavioral biometrics-driven system is developed for identifying harmful insiders and preventing masqueraders [49]. It collects crucial system events such as process creations, registry key changes, and file system actions using a Windows-specific sensor. To define a user's behavior, eighteen characteristics are collected from these occurrences. For their discriminative capabilities, these characteristics are evaluated using the Fisher's approach and the Fisher Linear Discriminant (FLD). The Gaussian mixture model (GMM), support vector machine (SVM), and kernel density estimation (KDE) are evaluated, with GMM outperforming the others.

Host log-based solutions are not prospering owing to data collecting issues, redundancy, and the enormous quantity of logs. It does, however, emphasize host logs' unique capacity to give a clear and efficient means of defining host or user behavior. Furthermore, it states that host log-based analytics are not limited to *nix operating systems, since equivalent data can be gathered from Windows event logs on Windows operating systems.

V. Methodology for Analyzing ML Threats and Mitigations

Let us look at the methodologies used in the research [1]. This paper analyzes ML (Machine Learning) threat behaviors and their impact on various ML components, including phases, models, tools, and related dependencies. The study also leverages ML threat knowledge from sources like ATLAS, ATT&CK, and the AI Incident Database, as well as vulnerabilities from ML repositories, to anticipate potential threats associated with specific packages or libraries.

Here is a breakdown of the key elements and research questions addressed in the study [1]:
The study [1] views 89 real-world ML attack scenarios, including 15 from ATLAS, 60 from the AI Incident Database, and 13 from the literature. It also concerns the analysis of 854 ML repositories, with 845 from GitHub and 9 analysis reports from PyPA.

RQ1: Prominence and Common Entry Points of Threat TTPs: This question seeks to recognize the prominence and common entry points of threat TTPs manipulated in ML attack methods. It aims to comprehend how ML threat TTPs execute and the series of TTPs used in attacks. The goal is to report better defense strategies against these threats.

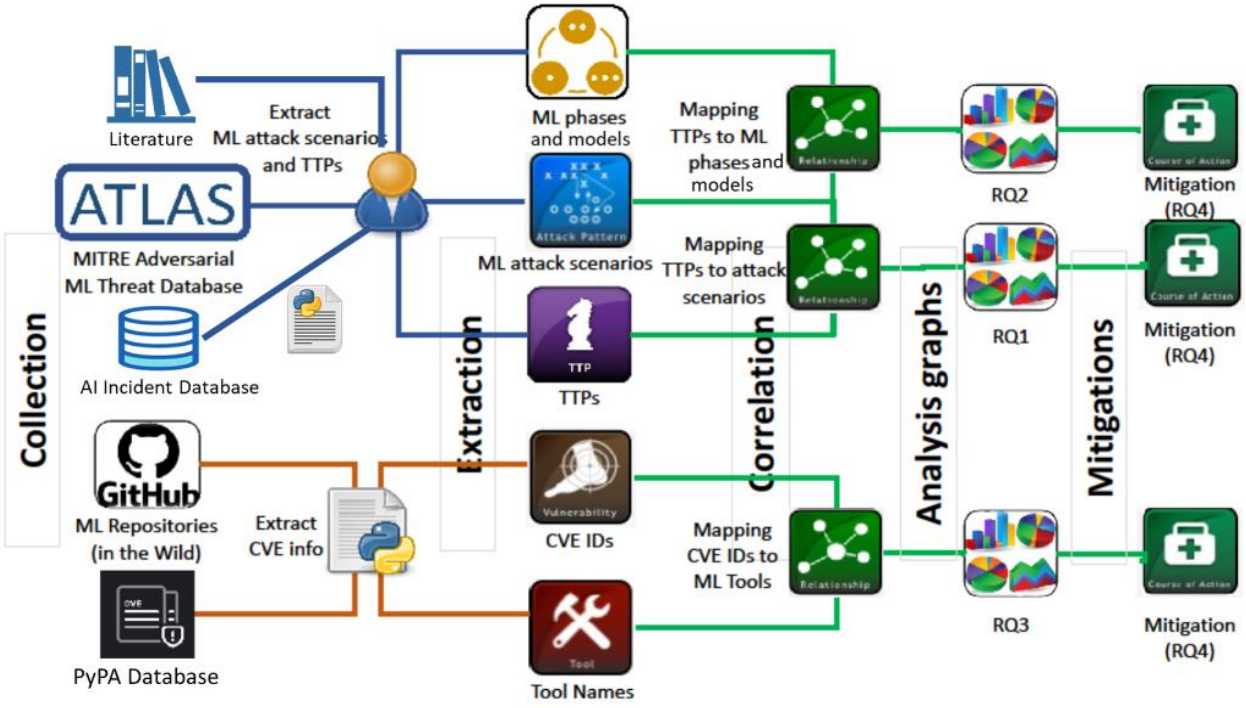

Figure 1: Study Methodology – a Roadmap

RQ2: Effect of Threat TTPs on ML Phases and Models: This question concentrates on understanding how threat TTPs affect different ML phases and models. It aims to supply insights into securing individual components of the ML lifecycle. It also identifies the ML phases frequently targeted and the TTPs commonly used in these phases.

RQ3: Identification of New Real-World Threats: This question assesses whether there are new real-world threats documented in the AI Incident Database, the publications, and ML repositories that were not previously documented in the ATLAS database. It aims to evaluate the totality of ATLAS and recognize potential security hazards that may have been overlooked. The investigation also looks into vulnerable ML repositories, their dependencies, and the most common vulnerabilities in these repositories.

RQ4: Mitigation Solutions for ML Stakeholders: This question aims to deliver end-to-end mitigation solutions to improve the security of ML assets. It seeks to propose recommendations for securing ML assets during development, from data-level security to cloud-level security. The analysis draws from existing security frameworks and guidelines such as those in references [2], [3], [10], and [11].

Methodology Roadmap: Figure 1 represents a roadmap of the study's methodology. It traces the series of steps and research activities that answer the research questions and achieve the study's objectives [1].

A. Data Collection and Processing for ML Threat Analysis

Let us describe the threat model used in the study, outline the goals of the attackers, their knowledge and specificity in conducting ML (Machine Learning) threats. We will also oversee a structured framework to understand how attackers aim to compromise ML assets and what they can and cannot access.

Attackers compromise the confidentiality, integrity, and availability (CIA) of ML-related assets, including data (e.g., training data, features, models). Poisoning attacks mostly target data integrity, while extraction attacks concentrate on stealing models or features, thus impacting data confidentiality.

Note: The following has been directly adopted from paper [1].

Let a $\in$ A be an asset, where A is a set of assets from the system S. An asset a can be owned or accessed by an entity (e.g., user, user group, program, set of programs), denoted E. E denotes the set of all entities and e $\in$ E. Let

$$ACS : A \times E \rightarrow R$$

be a function, that defines the level of privilege that an entity E has on an asset a or an asset group Ag $\subseteq$ A, under the system S.

R is a set of right access and it can take values R = {none, user, root} meaning that entities can have either no privilege (none), user access on A (user), and full access on A (root); or R = {none, read, write} meaning that entities can have no privilege (none), read access on A (read), and write access on A (write).

When AC_{AWS} ('model.pkl',' ml api') = root, it means that the Amazon Web Service (AWS) ML API service ml api have full access on the pickled model file model.pkl. When AC_{VM} ('training.csv','john') = write, it means that user john can

modify or delete the training data file training.csv in the virtual machine VM .

Let P_1, ..., P_n be a set of premises and C the goal to achieve. This relation is represented by

$$(P_1, ..., P_n) / C$$

It also means that C can succeed when properties P_1, ..., P_n are satisfied. Based on [54], we define the following notations. The notation

$$--a \rightarrow E$$

means that a is E's asset. Given $k_E \in K$ a protection property (e.g., encryption key, certificate, token), the notation $\{a\}k_E$ means that the protection k_E is enforced on asset a by an entity E. Let E1, E2 $\in$ E be two entities that share an asset a. The notation

$$E1 \leftarrow a \rightarrow E2$$

means that a is shared by E1 and E2. The sharing is satisfied when E1 send a to E2 and E2 send a to E1 as follows,

$$(E1 -a \rightarrow E2, E2 -a \rightarrow E1) / (E1 \leftarrow a \rightarrow E2)$$

Let m: A→C be a model function that takes data in A and return decisions in C based on the inputs. C can be two classes (i.e., {c1, c2}) or multiple classes (i.e., {c1, c2, ..., cn}), where c1, c2, ..., cn $\in$ C.

Note that, in black-box settings, an attacker AT does not have direct access to ML assets AV of the target victim V (i.e., model, executing code, datasets), i.e., $\forall aV \in AV$, ACV (aV , AT) = none [30]. They have only access to the ML inference API using an access token kV obtained as a legitimate user from the victim's platform V , i.e.,

$$a_{AT} \rightarrow AT$$

where $a_{AT} \in$ A are data crafted offline by attacker AT to be send via API. During the attack, AT performs queries using the victim's ML inference API and observes outputs. To do so, AT sends an online request with the crafted data aAT using access token kV , i.e.,

$$AT \ --\{aAT\}kV \ \rightarrow V$$

Then, AT will receive prediction responses and analyze it to further improve its data for attack, i.e.,

$$V --\{mV(aAT)\} \rightarrow AT$$

where mV is the executed model behind the ML inference API of the target victim V. In white-box settings, attacker AT may have an internal access to some ML assets $\tilde{AV} \subseteq AV$ of the target victim V (e.g., model, training data), i.e.,

$$\forall aV \in \tilde{AV}, ACV(aV, AT) \in \{read, write\}$$

Then, AT can perform several state-of-the-art attack techniques such as poisoning, evasion, extraction, and inference.

Specificity. In adversarial settings, ML threats can target a specific class/sample for misclassification (adversarially targeted) or any class/sample for misclassification (adversarially untargeted). The goal of AT is to maximize the loss L so that model mV misclassifies input data,

$$arg\ max\ L(mV(a), c)$$

where a $\in$ A is an input data, c $\in$ C is a target class, and m_V (a) is the predicted target data given a. To achieve the goal, AT can execute a targeted or untargeted attack affecting integrity and confidentiality of data [12], [13].

When attack is targeted, AT substitutes the predicted class c by adding a small perturbation θu(a, c) so that

$$mV(aAT) = c$$

where $a_{AT} = a + \theta u(a, c)$ is an adversarial sample.

In untargeted attack, AT adds a small perturbation θt(a) to input a so that

$$mV(aAT) \ != \ mV(a)$$

where $a_{AT} = a+\theta t(a)$ is an adversarial sample. ML threats can also leverage traditional TTPs to achieve goal.

Note: This marks the end of the adopted information.

In black-box settings, attackers (AT) cannot directly access ML assets of the target victim (V). They can only access the ML inference API using a legitimate access token 'kV' obtained from the victim's platform. Attacker AT crafts data 'aAT' offline to be transmitted via the API and performs queries using 'kV.'

In white-box settings, attacker AT may have inner access to some ML assets of the target victim, allowing read and write access. Attackers can utilize various attack techniques, including poisoning, evasion, extraction, and inference.

In adversarial settings, attackers may perform targeted attacks (adversarially targeted) or untargeted attacks (adversarially untargeted). Targeted attacks drive the model to misclassify a specific class or sample, while untargeted attacks make the model misclassify any class or sample. Traditional threats can be targeted, where attackers vigorously pursue a specific target, or untargeted, where they spread through the network without a select target.

In traditional attacks, threat actors can target assets like user accounts, servers, virtual machines, databases, and networks. Their goal is to circumvent authentication and firewalls to gain full control of ML assets, causing potential damage.

The data collection methodology, used in the study to gather information related to ML threats, vulnerabilities, and attacks, involves selecting appropriate data sources based on criteria such as recency, consistency, and reputation.

Threat actors use many tactics to deploy ML attacks, including 'Reconnaissance', 'Resource Development', 'Initial Access', 'ML Model Access', 'Execution, Persistence', 'Defense Evasion, Discovery', 'Collection', 'ML attack staging', 'Exfiltration', and 'Impact'. These tactics includes gathering information, setting up command and control (C2) infrastructure, gaining access to ML infrastructure, accessing model internals, running malicious code, evading detection, collecting data, and preparing for the attack's impact.

The three main criteria to select data sources are recency, consistency, and reputation. **Recency:** Data sources should contain recent information about ML vulnerabilities and threats. **Consistency:** The data sources should supply information on vulnerabilities and threats. **Reputation:** The reputation of the maintaining organizations, the number of stars for GitHub repositories, and the frequency of updates denote the best data sources.

Many data sources are chosen based on the criteria mentioned above: **ATLAS Database:** Contains ML attack scenarios from 2019 to 2021. **ATT&CK Database:** Contains enterprise attacks from 2018 to 2021, used to complement TTP definitions in ATLAS. **AI Incident Database:** Contains real-world AI incident information from 2003 to 2022. **Literature:** Contains 14 threats from academic papers published between 2010 to 2021. The associations with well-known organizations and media outlets bolster the reputation of these sources. **GitHub Repositories:** GitHub repositories are picked based on the reputation criteria. A search query identifies repositories related to machine learning with a significant number of stars (>1000) and sorted by stars in descending order. This search returns 845 repositories, which are deemed reputable based on their star counts. **PyPA Database:** Nine ML repositories are selected from the Python Packaging Authority (PyPA) database. These repositories are picked based on their applicability to the study. **Confidence Level:** The search query used for GitHub repositories returns a selection with a confidence level of 99% and a small error margin of 4.44%.

Data Processing is a study that involves three main phases: extraction, correlation, and metric computation. Below is the explanation to each of these phases: **Extraction:** Threat information is manually extracted from many data sources, including the ATLAS database, the literature, and the AI Incident database. Extraction involves reading ML attack scenarios, analyzing the tactics used in these scenarios, and referencing tactic characterizations provided in ATLAS or ATT&CK. The AI Incident database is studied to gather recent real-world ML attacks. Keywords such as "attack" are used to specify relevant information. **Correlation:** Correlation involves organizing and connecting the extracted threat information. Information from different sources and datasets is correlated to create a comprehensive view of ML threats and vulnerabilities. **Metric Computation:** Metrics related to ML threats, vulnerabilities, and attacks are computed based on the extracted and correlated data. These metrics provide insights into the prevalence, impact, and characteristics of ML threats and vulnerabilities.

In the data extraction phase of the study, information from various sources, including the ATLAS database, the AI Incident database, the literature, and ML repositories, is collected. **ATLAS Database:** Information is taken from the ATLAS database, which contains 15 million attack scenarios along with their Tactics, Techniques, and Procedures (TTP) definitions. The study extracts definitions of each ML attack scenario and looks for associated TTP definitions in ATLAS and ATT&CK. The attack goal, knowledge level, and specificity for each scenario are also extracted based on the defined threat model. **AI Incident Database:** Data is extracted from the AI Incident database, specifically 60 recent real-world ML attacks that occurred between 2018 and 2022. The study uses a crawler to gather this information, and any duplicate entries are deleted. Manual reading of the attack definitions is performed to identify potential TTPs equivalent to those in ATLAS/ATT&CK, as well as details such as the models used, attack goals, knowledge levels, and explicitness. **Literature:** Information is taken from literature, specifically from research papers related to ML threats and vulnerabilities. These papers are reviewed to gather relevant data, including depictions of threats, attack tactics, and techniques.

The extracted data provides understandings into ML threats, vulnerabilities, and attacks, including the purposes of attacks, the level of knowledge attackers have, and whether attacks are targeted or untargeted.

This data extraction process aids in creating a comprehensive dataset for further analysis in the study. For example, the study [1] has an attack scenario executed against the Microsoft Azure Service, which involved several phases targeting the confidentiality, integrity, and availability of the ML system. The attack's knowledge level is categorized as white-box, and its explicitness is adversarially untargeted. Additionally, new threats pulled from the AI Incident database that were not documented in ATLAS are identified, contributing to a more complete understanding of ML threats in the study.

B. Analyzing ML Threats and Vulnerabilities: Methodology

Title	Tactics and Techniques
India's Tek Fog Shrouds an Escalating Political War	***Resource Development***(Establish Accounts), ***Initial Access*** (Valid Accounts), ***ML Attack Staging*** (Create Proxy ML model: Use Pre-Trained Model), ***Exfiltration*** (Exfiltration via Cyber Means)
Meta says it's shut down a pro-Russian disinformation network, warns of a social hacking operation	***Resource Development*** (Establish Accounts), ***Initial Access*** (Valid Accounts), ***Exfiltration*** (Exfiltration via Cyber Means)
Libyan Fighters Attacked by a Potentially Unaided Drone, UN Says	***Reconnaissance*** (Active Scanning), ***Impact*** (Cost Harvesting)
Fraudsters Cloned Company Director's Voice In $35M Bank Heist, Police Find	***ML Attack Staging*** (Create Proxy ML Model: Use Pre-trained Model), ***Impact*** (Cost Harvesting)
Poachers evade KZN park's high-tech security and kill four rhinos for their horns	***Defense Evasion*** (Evade ML Model), ***Impact*** (Evade ML Model)
Tencent Keen Security Lab: Experimental Security Research of Tesla Autopilot	***ML Attack Staging*** (Craft Adversarial Data), ***Impact*** (Evade ML Model, Cost Harvesting)
Three Small Stickers in Intersection Can Cause Tesla Autopilot to Swerve Into Wrong Lane	***ML Attack Staging*** (Craft Adversarial Data), ***Impact*** (Evade ML Model, Cost Harvesting)
The DAO Hack - Stolen $50M & The Hard Fork.	***Impact*** (Cost Harvesting, Denial of Service)

Table 1: ML real-world attack TTPs from the AI Incident database

The following outlines the analysis process to understand and catalog machine learning-related security threats and vulnerabilities using data extracted above (GitHub repositories and databases containing CVE data).

Table 1 and ML Real-World Attack (Tek Fog): In Table 1, there is a reference to an ML real-world attack called "Indian Tek Fog Shrouds an Escalating Political War." This attack involves using an ML-based bot app named "Tek Fog." Tek Fog's main objectives contain misinterpreting public opinion by creating temporary email addresses, bypassing authentication systems on various platforms (WhatsApp, Facebook, Instagram, Twitter, Telegram), sending fake news, hijacking trends, phishing inactive WhatsApp accounts, spying on personal information, and building a database of citizens for harassment. The bot may utilize a Transformer model like GPT-2 to develop coherent text-like messages. This attack targets the confidentiality (by bypassing authentication systems and spying on confidential information) and integrity (by posting counterfeit news) of the ML system.

Table 3 and Carlini et al. [12] Attack: In Table 3, there is a statement of the "Carlini et al. [12] attack." This attack uses a GPT-2 pre-trained proxy model to extract training examples by querying the target ML model. It focuses on compromising the privacy and confidentiality of the training data used by the ML model. It operates in black-box mode, trying to recover data to train the model. The attack is indiscriminate and does not target certain pieces of training data.

GitHub Data Mining: The authors mined titles and comments from issues in 845 machine-learning repositories on GitHub. They applied two filters - a remark filter and a label filter, to extract appropriate information about vulnerabilities and threats. Keywords such as "cve" for CVE codes, "vuln" for vulnerabilities, "threat" for threats, "attack" for attacks, "secur" for security, and "attacker" were utilized to identify appropriate information. They also extracted CVE IDs (Common Vulnerabilities and Exposures) from the data. Some labels and remarks contained threat descriptions without CVE IDs but included external links to corresponding sources. The authors browsed these links and sources to extract hidden CVE IDs.

Title	Tactics and Techniques
Carlini *et al.* [4]	***ML Attack Staging*** (Create Proxy ML model: Use Pre-Trained Model), ***Exfiltration*** (Exfiltration via ML Inference API: Extract ML Model)
Biggio *et al.* [34]	***ML Attack Staging*** (Craft Adversarial Data), ***Defense Evasion*** (Evade ML Model), ***Impact*** (Evade ML Model)
Barreno *et al.* [55]	***ML Attack Staging*** (Craft Adversarial Data), ***Defense Evasion*** (Evade ML Model), ***Impact*** (Evade ML Model)
Carlini *et al.* [56]	***ML Attack Staging*** (Craft Adversarial data), ***Defense Evasion*** (Evade ML Model), ***Impact*** (Evade ML Model)
Wallace *et al.* [57]	***ML Attack Staging*** (Create Proxy ML model: Use Pre-Trained Model), ***Exfiltration*** (Exfiltration via ML Inference API: Extract ML Model)
Abdullah *et al.* [33]	***ML Attack Staging*** (Craft Adversarial Data), ***Defense Evasion*** (Evade ML Model), ***Impact*** (Evade ML Model)
Chen *et al.* [31]	***ML Attack Staging*** (Craft Adversarial Data: Insert Backdoor Trigger), ***Persistence*** (Backdoor ML Model: Inject Payload)
Choquette-Choo *et al.* [37]	***ML Attack Staging*** (Craft Adversarial Data), ***Exfiltration*** (Exfiltration via ML Inference API: Infer Training Data Membership)
Papernot *et al.* [58]	***ML Attack Staging*** (Craft Adversarial Data), ***Defense Evasion*** (Evade ML Model), ***Impact*** (Evade ML Model)
Goodfellow *et al.* [59]	***ML Attack Staging*** (Craft Adversarial Data), ***Defense Evasion*** (Evade ML Model), ***Impact*** (Evade ML Model)
Papernot *et al.* [60]	***ML Attack Staging*** (Craft Adversarial Data), ***Defense Evasion*** (Evade ML Model), ***Impact*** (Evade ML Model)
Cisse *et al.* [61]	***ML Attack Staging*** (Craft Adversarial Data)
Athalye *et al.* [62]	***ML Attack Staging*** (Craft Adversarial Data), ***Defense Evasion*** (Evade ML Model), ***Impact*** (Evade ML Model)
Jagielski *et al.* [17]	***Reconnaissance*** (Search for Victim's Publicly Available Research Materials), ***ML Attack Staging*** (Craft Adversarial Data), ***Exfiltration*** (Exfiltration via ML Inference API: Extract ML Model)

Table 2: ML real-world attack TTPs from the literature

CVE Database and PyPA Database: The extracted CVE IDs were entered into the National Vulnerability Database (NVD) and the CVE database to acquire precise information. This information contained the severity level of the vulnerability, the dependency liable for it, the version affected, and potential threats resulting from the vulnerability. The PyPA database was also used to extract CVE IDs. Duplicate CVE IDs found in both GitHub data and the PyPA database were ignored. In total, 226 CVE IDs were extracted for analysis.

Building an Attack Matrix: The extracted data, containing CVE IDs and associated information, was utilized to semi-automatically build an attack matrix for analysis.

Let us create an attack correlation matrix (CCM) to understand the relationships between various aspects of attacks and ML components. Using this, we will answer certain specific research questions related to the tactics used in attacks, their impact on ML phases, and the ML models targeted or exploited in these attacks.

Attack Correlation Matrix (CCM): After extracting information regarding attack vectors, including vulnerabilities and threats, the next step is to develop an attack correlation matrix. The objective of the CCM is to establish relationships between the elements of an attack vector and the elements of an element-of-interest (EOI),

which can be an ML component (such as phases, models, or tools) or a stage within an attack scenario. The CCM helps to understand how various attack vectors affect specific components within the ML ecosystem.

Threat CCM (Cross-Correlation Matrix): The CCM has two categories: threat CCM and vulnerability CCM. The threat CCM maps threat Tactics, Techniques, and Procedures (TTPs) to EOIs like attack scenarios and ML phases.

Mapping between TTP Features and Attack Scenarios: To address research question RQ1, the authors link threat features (goals, knowledge, specificity, capability/tactics) of ML threats to attack scenarios. The threat matrix shows coefficients describing the strength of relationships between threat features and attack scenarios. Coefficients can take many values, including strings like "Black box," "stage 0," or empty values indicating no relation or "N/A" for unknown or unmentioned relations. The "stage i" notation means the execution of a specific tactic at a certain step in the attack execution flow.

Mapping between Tactics and ML Phases: To address research question RQ2, the tactics used in attacks are mapped to ML phases. This mapping is done by reviewing the descriptions of tactics and associated methods in the ATLAS and ATT&CK databases to find ML phase signatures. Keywords and associations with ML phases identify the effect of tactics on different ML phases. The resulting information is recorded in a threat CCM for analysis.

Mapping between Attack Scenarios and ML Models: Another element of the analysis is to figure out which ML models are targeted or manipulated by attack scenarios. This mapping is achieved by digging for information concerning the targeted model in ATLAS TTP descriptions, AI Incident database descriptions, or by performing web searches when no information is found. For instance, if an attack scenario says, "Convolutional Neural Network," it is recorded in the threat CCM as an impact on that model. The CCM captures the relationships between attack scenarios and targeted/exploited ML models.

RQ1: Prominence and Common Entry Points of Threat TTPs Exploited in ML Attack Scenarios: This research question has two parts: the prominence of threat TTPs exploited in attack scenarios and the common entry points. The most notable threat tactic in ML attack scenarios is "ML Attack Staging," which occurs 30 times across 89 ML attack techniques. During this stage, threat actors organize the attacks by preparing adversarial data, training proxy models, or poisoning target models. Other noteworthy tactics include "Impact," which occurs 21 times, and "Resource Development," which occurs 15 times. Common entry points in attack methods are "Reconnaissance" and "ML Attack Staging." Attackers manipulate public resources like research materials, ML artifacts, and adversarial ML attack implementations during reconnaissance. ML Attack Staging involves preparing attack data or models.

RQ2: Impact of Threat TTPs Against ML Phases and Models: This research question examines the influence of threat TTPs on ML phases and models. The most targeted ML phases are "Testing," "Inference," "Training," and "Data Collection." Tactics such as "Reconnaissance," "Impact," "ML Attack Staging," and "Resource Development" are often used across various ML phases. These findings supply insights into which ML phases are more powerless to threats and which tactics are commonly employed across phases.

RQ3: Finding New Threats Not Documented in ATLAS: This question aims to determine new threats in the AI Incident database, literature, and ML repositories not documented in ATLAS. From the AI Incident database and literature, 9 techniques and 7 tactics were identified in 8 ML attacks. These contain tactics like "Create Proxy ML model," "Exfiltration via Cyber Means," and "Evade ML Model." These newly identified threats were not documented in ATLAS and can be used to improve ATLAS case studies and knowledge of ML security threats.

C. Host-Based Analytics: Leveraging Individual Host Data

There are three primary types of data sources: host-based, network-based, and contextual. The emphasis is on host-based analytics, which entails working with data gathered from specific hosts. Host-based analytics make use of a variety of information gathered from unique hosts. This information consists of both high-level application data and low-level operating system data, such as system calls, shell command lines, mouse/keyboard dynamics, *nix syslog, and Windows event log. These data sources provide information on both the host's conduct and the way that guests interact with the host. System calls occur when a computer application seeks a service from the kernel of the operating system. To gather system calls, many methods such as the *nix operating system's built-in Auditing System or the strace/ptrace tools can be used. Analyzing these calls can aid in the detection of different cybersecurity issues, such as breaches and malware. For example, privileged process execution analysis has proven useful against such a threat.

- **Commands and Keyboard/Mouse Dynamics:** User orders frequently form sequences, but keystroke/mouse dynamics need the adoption of specific models for characterization [34]. Due to privacy considerations, gathering this user input data might be difficult. It might be determined by whether the host has an in-built command logger or a third-party command/ keystroke/ mouse recorder. These data sources are especially effective for detecting masqueraders since they provide information that may be used to identify real users based on behavioral biometrics.
- **Operating System Logging:** Operating system's built-in logging capabilities captures different system events, such as authentication, system daemon activity, kernel messages, process-related events, and policy changes [33]. This is referred to as syslog in *nix operating systems, and the event log in Windows. These logs include useful information; however, they can be excessively long owing to duplication entries.
- **Processing and Feature Extraction:** Raw logs might be complicated and redundant, making them less useful for

identifying breaches [35]. There are two alternative strategies to fix this. One approach is to devote time and effort to analyzing raw logs and extracting characteristics in order to address specific cybersecurity concerns. The alternative approach is to reengineer the logging capabilities to capture only information relevant to certain dangers, as exemplified by the "RUU" logger, which monitors system events relating to process, registry, and file operations.

D. Network Logs for Analyzing Network Traffic

Network core equipment such as routers, switches, load balancers, and firewalls, as well as operational servers such as Proxy, DHCP, DNS, Active Directory (AD), and Email may capture network traffic data, which has long been seen as a valuable source for identifying intrusions ([50]]). In addition to network traffic, servers and devices may be configured to generate network logs. According to the text, various works have identified the significant potential of network logs in tackling insider threats ([51]). Network logs are more application-specific and may accurately represent how users or hosts behave inside a network, especially in the context of a given service or application. Network traffic is data traveling over a network at a specific point in time, often packed in packets ([52]). It differentiates network traffic from network logs by emphasizing that network traffic provides more information at a lower network/transport layer.

Data Acquisition Categories: Thre are three important technology categories for gathering network traffic data:

- NetFlow, for example, "Cisco NetFlow" and "sFlow."
- Simple Network Management Protocol (SNMP)-based applications such as "MRTG" and "Cricket."
- Packet sniffers, for example, "snoop" and "tcpdump."

Network Traffic Analytical approaches: Numerous analytical approaches may be used to deal with cyber security concerns utilizing network traffic data. It demonstrates these techniques:

- Regression analysis may be used to create a time series of traffic volumes between two entities (for example, a host and a domain name). This enables the detection of abnormal volumes that deviate greatly from their predicted values.
- Analysis of the periodicity of communication patterns between a host and a domain name can reveal the presence of beaconing behavior.
- A method for detecting anomalous or malicious user/host activity is connectivity analysis. It can, for example, detect users/hosts connecting to questionable IP addresses or failing to connect to expected destinations.

Network logs can be used as an alternate data source to network traffic in some instances. Network logs frequently reflect specific application layer protocols or features, such as AD logs from a Windows domain controller, which record user logon/off and permissions checks. In general, network logs are more organized and formatted, requiring less effort to get information. Furthermore, many network traffic analysis techniques may be extended to analyze network logs with small changes.

VI. Challenges in Detecting Threats

Despite almost two decades of study devoted to identifying and thwarting insider threats, these initiatives have fallen behind the quick development of contemporary networks. As a result, malicious insiders continue to cause victims to sustain substantial losses. This persistent issue is exacerbated by several major difficulties:

1. **Late Detection:** First, late detection Current methods frequently fall short of spotting early signs of dangerous insider activity. Normally, they only provide alerts after harmful activities have already taken place.
2. **Limited Data Sources:** Many solutions only use specific audit data sources, which only offer a limited amount of information about potential threats.
3. **Over-Reliance on Domain Knowledge:** Excessive reliance on domain expertise Traditional data analytics rely on domain expertise to develop rules and extract features. This strategy reduces their ability to counter developing threats.

The following are the research challenges

1. **Big Dirty Data Storage and Management:** Insider threat research frequently entails dealing with large amounts of data that are high in volume, velocity, and diversity. Complexity is introduced through the usage of many operating systems, data gathering methods, hardware, and software. There is a requirement for a specific big data platform to manage these data. However, creating such a platform may be time-consuming and expensive. One possible answer is to organize and process the data using open-source technologies such as Apache Hadoop. "Pre-processing of big dirty data," which entails cleaning and preparing the data for analysis, is another key difficulty. Data cleaning is an important step in identifying and correcting faulty or incorrect data.
2. **Knowledge Extraction and Management:** Data evaluation, transformation, and correlation are all part of the knowledge extraction process. It is also difficult to effectively manage the retrieved information for reusability. To increase the reusability of extracted knowledge, the Common Information Model (CIM) is proposed as a feasible option.
3. **Intelligent Decision Making:** Insider threat detection decision-making frequently depends on human review by domain experts and is prone to high false positive rates (FPRs). Manual examination is impractical due to the large volume of data. The importance of automated decision-making in lowering FPRs and reducing dependency on human labor is underlined. To solve these issues, the paragraph advises using machine/deep learning frameworks and ensembles of many detectors. Machine learning has the potential to automate information extraction and lessen reliance on prior knowledge, but it may also result in reduced

interpretability. By incorporating past knowledge into the feature extraction process, a balance between automation and interpretability may be achieved. It can assist minimize FPRs by taking into account semantic properties. A final choice is reached by aggregating data from several specified use cases, which can improve interpretability. However, relying heavily on past information might limit reusability and extendability.

VII. Conclusion

The research concludes various software-level and network-level vulnerabilities found in ML repositories, both from GitHub and the PyPA database. These can lead to a wide range of potential threats, including code execution, DoS, and data breaches. Hence the developers and maintainers must address these vulnerabilities and ensure the security and integrity of their projects.

Impact on ML Repositories: TensorFlow, OpenCV, Ray, and NNI are among the repositories with the most vulnerabilities. Note that the number of vulnerabilities does not indicate the repository's security posture. Some repositories may have more vulnerabilities disclosed due to active security practices and frequent updates.

Dependencies Causing Vulnerabilities: TensorFlow was impacted by several dependencies, some with high severity and others with medium severity. The severity levels are determined using the Common Vulnerability Scoring System (CVSS) v3/v2 scores.

Critical Severity Dependencies:
Pickle Dependency: This dependency had a critical severity and impacted several repositories, including Pytorch, NNI, Pandas, Numpy, and Scikit-Learn. The severity indicates that vulnerabilities in the Pickle dependency could usher significant security issues in these repositories.

Python391 (version 3.9.1 of python): This version had a critical severity and affected the stated repositories, indicating that vulnerabilities in the Python interpreter could have severe consequences for ML projects.
High Severity Dependencies:

Numpy116 (version 1.16.0): This version of Numpy had an increased severity and impacted multiple repositories, including Pytorch, NNI, Pandas, Numpy, and Scikit-Learn. Increased severity implies that vulnerabilities in this Numpy version could lead to heavy security risks for these repositories.

Deeplearning4j Dependencies: Deeplearning4j was affected by three dependencies with high severity:

- commons-compress118 (Apache version 1.18): This Apache Commons Compress version had a high severity and impacted Deeplearning4j.
- jackson-bind100 (version 1.0.0 beta7): This version of Jackson Bind had a high severity and affected Deeplearning4j.
- snakeyaml124 (version 1.24): This SnakeYAML version also had a high severity and impacted Deeplearning4j.

Ray and MLflow Dependencies: Both Ray and MLflow were impacted by the Log4j dependency, which had an increased severity. This implies that vulnerabilities in Log4j could pose significant security risks to these repositories.
Other High Severity Dependencies: Other repositories like OpenCV and Mxnet were respectively impacted by the libpng and requests dependencies, both with increased severity.

Most Frequent Vulnerabilities (GitHub Repositories):

- **Denial-of-Service (DoS):** This vulnerability was the most frequent and occurred in multiple repositories, including TensorFlow, OpenCV, Deeplearning4j, Ray, Scikit-Learn, and Gym.
- **Arbitrary Code Execution:** Occurred in TensorFlow, NNI, and Pytorch.
- **Remote Code Execution (RCE):** Found in OpenCV, Ray, and MLflow.
- **Use-After-Free (UAF):** Detected in TensorFlow, OpenCV, and Ray.
- **Buffer Overflow:** Occurred in TensorFlow, OpenCV, and Pytorch.
- **Other Vulnerabilities:** Out-of-Bounds, Code Injection, Null-Pointer Dereference, and Credential Sniffing were found in various combinations across repositories.

Most Frequent Vulnerabilities (PyPA Database Repositories):

- **Arbitrary Code Execution:** Occurred in Notebook, Pandas, and Numpy.
- **Cross-Site Scripting (XSS):** Found in Notebook and Jupyter-lab.
- **Denial-of-Service (DoS):** Detected in Numpy, NLTK, and Protobuf.
- **Buffer Overflow:** Occurred in Numpy and NLTK.
- **Open Redirect:** Found in Notebook and Jupyter-server.

Mitigation Strategies: These strategies are based on security guidance from various sources including MITRE ATT&CK, MITRE D3FEND, NIST security guidelines, and the Cloud Security Alliance.

Data-Level Mitigations:

- Adversarial Defenses: Implement adversarial defense techniques to protect against adversarial attacks on data.
- Code Signing Certificates: Use code signing certificates to ensure the authenticity and integrity of data sources.
- Data Protection: Enforce data protection measures in transit (e.g., TLS encryption), at rest (e.g., AES-based keys), and in use (e.g., dynamic analysis).
- Identity and Access Management (IAM): Implement IAM practices based on the principle of least privilege to restrict unauthorized access to ML artifacts.

Software-Level Mitigations:

- Vulnerability Scanning: Regularly scan ML libraries, pipeline, and model source code for vulnerabilities using tools like GitHub code scanning, Tsunami, OSS-fuzz, and SonaQube.

- IAM for Running Processes: Enforce IAM with least privilege on running processes, applications, containers, and virtual machines to restrict access.
- Database Security: Proper IAM configuration for databases, regular backups kept separate from the corporate network, and secure database access controls.

System-Level Mitigations:
- OS Hardening: Harden the operating system with practices such as automatic updates, Secure Boot, access permission limitations, and patch management.
- Endpoint Detection and Response (EDR): Implement EDR solutions or antivirus software for threat detection.
- Logging and Auditing: Implement robust logging and auditing mechanisms to monitor system activities.

Network-Level Mitigations:
- Network Access Control Lists (NACLs): Apply NACLs to virtual private clouds (VPCs) to control traffic access.
- Firewalls and IPS: Install firewalls and Intrusion Prevention Systems (IPS) as the first line of defense against malicious traffic.
- Encrypted Tunnels: Use encrypted tunnels (e.g., IPSec VPN) for end-to-end security of remote accesses.

Cloud-Level Mitigations:
- IAM in the Cloud: Implement IAM in cloud environments with policies based on least privilege, role-based permissions, and multi-factor authentication (MFA).
- Cloud Security Solutions: Enforce cloud security policies using solutions like cloud access security brokers (CASB), Zero Trust, and secure access service edge (SASE).
- Endpoint Protection: Utilize endpoint protection solutions like EDRs.
- Network Prevention and Detection: Implement network prevention and detection systems.
- Auditing and Logging: Enable auditing and logging in cloud environments for monitoring.

My Observation and Critical Review:
It is important to note here that the AI based NGAVs are evolving in two different directions. One direction is where you analyze a plethora of information on how not to secure a system – gather all potential live system activities that can compromise the system and act against it. The other direction is mostly procedural, you use the existing hardening techniques onto the system and orchestrate it to be able to close the doors against a malware invasion through heuristic based detection. However, it's important to note that the latter method can be bypassed over time and excellency (which will be explained through a case study below). Heuristics can work as an amazing fix against potentially harmful and complicated attacks but fail to act against very simple ones.

AI on NGAVs will work based on what objectives and mitigation direction it was tailored for, if you want it to eliminate threats through analysis of how system is performing at a given time (Like a doctor checking a patient for flu), you may follow a SOAR system which works based on a database of system MIS. On the other hand, if you want the solution to be based on the kind of behaviors the invading malware exhibit (Like closely detecting a microorganism), the solution would be built upon a database of heuristics of the malware. The latter needs to be refreshed more often; we will see why below.

VIII. Case Study I (NGAV Success) – The EDR Killer

On May 21, a new threat actor, self-identifying as 'Spyboy', appeared on the Russian Anonymous Marketplace (RAMP) with a tool known as Terminator EDR Killer. This Terminator tool could incapacitate 23 antivirus, endpoint detection and response (EDR), and extended detection and response (XDR) products and controls. Among the affected products were EPP Solutions from Sophos, CrowdStrike, Kaspersky, McAfee, BitDefender, Malwarebytes, ESET, and even Windows Defender on Windows 7 and later [70].

During 2021, Spyboy priced this tool at USD 3,000 for a comprehensive bypass or USD 300 for targeting a specific AV/EDR/XDR.

Additionally, the threat actor released videos showing the tool's capabilities against Sophos and CrowdStrike solutions. Spyboy disclaimed any responsibility for ransomware or locker usage.

A. Analyzing the Terminator EDR Killer

Spyboy's Terminator tool uses the Bring Your Own Vulnerable Driver (BYOVD) attack method. This approach involves deploying drivers that possess legitimate signatures, allowing them to load onto Windows systems. These drivers enable attackers to disable security solutions and take control of the target machine.

In a Reddit post [71], research engineer Andrew Harris revealed that Terminator specifically deploys and loads a vulnerable version of the Zemana AntiMalware kernel driver. This driver exploits a security vulnerability tracked as "CVE-2021-31728" signed by "Zemana Ltd." It is placed in the C:\Windows\System32\drivers\ folder with a random name assigned, typically consisting of four to ten characters.

Once the vulnerable driver is written to the disk, and the user approves the User Account Control (UAC) prompt, the Terminator tool activates the driver. This driver, with its kernel-level privileges, is then employed to terminate security software processes, which are usually protected.

The use of BYOVD attack techniques has been on the rise, with various malicious entities, including ransomware groups and state-sponsored cyber-espionage outfits like the North Korean hacking group Lazarus, adopting it. Well known threat actors like BlackByte use vulnerable drivers to execute malicious code within the kernel context, granting them elevated privileges within the system.

B. How was this mitigated?

Dmitry Bestuzhev, BlackBerry's Senior Director of Cyber Threat Intelligence, highlights that the tool cannot run autonomously and requires manual user action. He notes that

"The end-user must be a local administrator and must also accept the UAC prompt." In terms of removal, if the user has administrator rights, they can simply uninstall the program through the "add/remove programs" feature in Windows or via an official anti-malware uninstallation tool.

To mitigate risks associated with tools claiming to disable anti-malware defenses:

- Users should have restricted user rights in the system.
- Company administrators should implement device control to limit the usage of undesirable USB-like devices.
- Organizations should enable self-protection features in their anti-malware products to ensure that even if the tool is downloaded inadvertently or with malicious intent, the antivirus solution remains intact.
- Visibility over Endpoints can help. Deploying MDMs such as ManageEngine MDM / Workspace One could be helpful.

This is one such case study where BlackBerry Cylance AI NGAV tool was able to perform better. One thing to note is that this mitigation requires a lot of protocol-oriented analysis, and detection is purely based on heuristics. One such example would be as below:

```
event_platform=Win event_simpleName=PeFileWritten "drivers"
"system32"
| regex
FilePath="^\\\Device\\\HarddiskVolume\d+\\\Windows\\\System32
\\\drivers\\\$"
| regex FileName="^[a-zA-Z]{4,10}\.sys$"
| stats count(aid) as writeCount by SHA256HashData, FileName,
FilePath
| where writeCount < 5
```

The solution falls under the two software solutions of Blackberry Cylance AI.

IX. A Case Study II (NGAV Failure) – The Universal Bypass

Skylight Cyber researchers from Australia have revealed the first universal technique to deceive an "artificial intelligence-based antivirus," making it perceive malware as harmless software, thus enabling it to run on seemingly protected machines. The AI antivirus under scrutiny is CylancePROTECT, which BlackBerry acquired the previous year to incorporate Cylance's AI antivirus technology into its Spark communications platform for the Internet of Things (IoT).

The Skylight researchers revealed that, after examining Cylance's engine and neural net model, they discovered that the AI antivirus heavily relied on string analysis and exhibited a particular bias. **Note that this is where the heuristic idea (mentioned above) fails miserably.** This bias was instrumental in the security researchers' ability to exploit the antivirus's vulnerability. They achieved this by appending a predetermined list of strings to any malicious file, which typically would have triggered detection but instead remained undetected. Their method exhibited a excellent success rate, effectively bypassing detection for 100% of the top 10 malware samples (for May 2019) and 90% of a more extensive sample consisting of 384 malware variants.

Cybercriminals consistently create new malware by modifying existing code to evade contemporary antivirus defenses, although this typically demands substantial effort for each new malware iteration. By exposing the AI antivirus's weakness, malicious actors can effortlessly infiltrate machines guarded by such antivirus software with a massive amount of slightly modified malware strains.

With this single research endeavor, Skylight's cybersecurity experts showcased the profound susceptibility of AI-based antivirus tools, such as CylancePROTECT, before they had widespread popularity in the cybersecurity world.

The researchers put forth an accessible analogy to elucidate the ease with which AI-based antivirus solutions can be manipulated.

Suppose an AI is trained to distinguish between birds and humans. It may eventually learn that a fundamental difference between the two is that birds have beaks while humans do not. The AI vendor might exclaim that this is where the AI scope ends. The vendor might presume that if the AI can effectively discern humans from birds, it should also be capable of distinguishing one from the other in images. This is the logic behind AI antivirus systems.

If an AI antivirus can scrutinize thousands of known malware samples and correctly identify the vast majority as malware, the vendor may assume it is highly efficient at detecting similar threats.

However, malware creators actively counteract antivirus measures and employ clever straightforward tactics to implement and can utterly confuse AI systems. In the earlier bird versus human analogy, a human wearing a mask with a bird beak could easily be mistaken for a bird.

X. Appendix

Software-Level Threats (GitHub Repositories): Arbitrary Code Execution: This vulnerability allows attackers to execute arbitrary code on the affected system.

Buffer Overflow: Attackers can exploit this vulnerability to overwrite the memory buffer's boundaries, leading to potential crashes or code execution.

Denial-of-Service (DoS): This vulnerability disrupts the normal functioning of a system, causing unavailability.

Out-of-Bounds (Read/Write): Attackers can read or write data outside the bounds of allocated memory.

Use-After-Free (UAF): This vulnerability arises when a program uses memory after it has been deallocated, leading to potential crashes or code execution.

Code Injection: Attackers can inject malicious code into a system.

Null-Pointer Dereference: This vulnerability occurs when a program attempts to read or write to a null pointer.

Untrusted Deserialization: Attackers can exploit this vulnerability to execute malicious code during the deserialization process.

Improper Validation: This vulnerability arises when input data is not properly validated, leading to security issues.

Excessive Iteration: This vulnerability can lead to resource exhaustion and system instability.

Divide-by-Zero: Attackers can exploit this vulnerability to cause divide-by-zero errors.

Reachable Assertion: This vulnerability occurs when an assertion is reached, indicating a programming error.

Crash-Insufficient Information: This vulnerability causes crashes due to insufficient error information.

Network-Level Threats (GitHub Repositories):
Remote Code Execution (RCE): Attackers can execute code remotely on the affected system.

Cross-Site Scripting (XSS): This vulnerability allows attackers to inject malicious scripts into web applications viewed by other users.

Credentials Discovery Sniffing: Attackers can sniff and discover credentials transmitted over a network.

Software-Level and Network-Level Threats (PyPA Database): Directory Transversal: This vulnerability can allow attackers to access files outside of the intended directory.

Cross Site Request Forgery (CSRF): Attackers can perform actions on behalf of an authenticated user.

Cross Site Inclusion: This vulnerability allows attackers to include external content into web pages.

Open Redirect: Attackers can redirect users to malicious websites.

Symlink: This vulnerability can allow attackers to manipulate symbolic links.

Command Execution: Attackers can execute commands on the affected system.

Cleartext Storage: Sensitive information may be stored in cleartext.

Inefficient Regex Complexity: Inefficient regular expressions can lead to performance issues.

APT Intrusion Kill Chain: This framework aids in comprehending the progression of a cyberattack.

Vulnerabilities and Attack Vectors: Reconnaissance, weaponization, delivery, exploitation, and installation are among the common attack vectors that attackers may use.

Delivery and Exploitation: Attackers utilize techniques like social engineering-based phishing and rogue websites to spread malware to victims' hosts. It also covers "exploitation," the process by which hackers use holes in the operating system and software to spread malware.

Intervention in the Victim Environment: Describes how the attacker takes over the victim's host after the malware has been transmitted and installed. This control gives the attacker the ability to carry out tasks like accessing confidential information and covertly spreading the virus.

System Calls: Communication channel via which a program interacts with the operating system's kernel. It emphasizes the thorough examination of system calls for detecting various sorts of cybersecurity risks, including intrusions, malware, and insider threats.

Sequence of System Calls: The following is an example of a subsequence of system calls recorded from a program (sendmail). System calls such as open, read, remap, and close are included in the sequence. It exemplifies the nature of system call sequences examined in cybersecurity.

System Call Analysis Technical Categories: Most extant techniques analyze sequences of system calls, with some schemes digging further into the parameters of individual system calls. It offers two popular technical categories for studying system call sequences:

- **Sequence-based Analysis:** It generates n-grams (sequences of n system calls) from the raw system call sequence and analyzes these n-grams using various models.
- **Frequency-based Analysis:** System call sequences are converted into fixed-length frequency vectors depending on the occurrence of each system call. Anomalies are then detected using statistical or machine learning methods.

Analytical Approaches of Various Types ([37], [38], [39], [40], [41]): Numerous analytical methodologies are used to analyze system calls, including:

- **Markov Chains and Hidden Markov Models (HMMs):** These models use joint probabilities to characterize the sequential transitions of system calls [37].
- **ANN-based Schemes:** ANNs are used to identify non-linear connections between input (n-grams) and output (normal or abnormal) [38] [39].
- **Deep Learning-based Schemes:** These schemes use deep neural networks to extract features from n-grams, however they have higher processing costs [40] [41] .

Early Pioneering Work ([36], [42]): The analysis of system calls for intrusion detection was pioneered in the 1990s by Forrest and Hofmeyr. Early schemes like "lookahead" focused on sequences of system calls related to privileged processes (e.g., sendmail and ftp). These schemes compared testing sequences to a baseline database of n-grams, flagging anomalies when the mismatch percentage exceeded a defined threshold. Later schemes improved accuracy using Markov chains, HMMs, and ANNs.

Algorithms for Frequency-Based Analysis and Classification: Frequency-based analysis involves transforming system call sequences into frequency vectors [43]. To distinguish anomalous frequency vectors from normal ones, classification methods such as k-nearest neighbor (kNN), k-means clustering (kMC) [44], support vector machine (SVM), and others are utilized. For decision-making, a sophisticated scheme is presented that integrates many detectors and incorporates algorithms such as signature-based, multinomial log-likelihood ratio test (LLRT), SVM, and logistic regression (LR) [45].

Introduction to Network Logs: Because most application layer services rely on network connectivity, network logs contain application-specific information extracted from network traffic. In essence, network logs, like network traffic data, may be utilized to address cybersecurity concerns.

Network Logs vs. Network Traffic: There are technological similarities between network logs and network traffic-based analytics. Both forms of data may be analyzed using statistical and machine learning (deep learning) methods. Network logs, on the other hand, give information at a higher tier and are usually more organized, making them suited for use in a full enterprise-level network monitoring system.

Myers et al.'s Web Server Logs Framework: Myers et al. provide a conceptual framework for detecting malevolent insiders that abuse corporate web servers. This framework integrates monitoring and detection capabilities into an existing log management system in order to expose illegal access and automated activity in web server logs.

Graph-Based Correspondence Pattern Algorithms: Some techniques analyze email and mobile phone data to identify insider threats using graph-based algorithms. These algorithms examine correspondence patterns and raise an event of interest when test patterns differ from normative patterns.

Hanley and Montelibano's Rule-Based Approach: Hanley and Montelibano present hands-on rules for identifying data exfiltration using proxy, email, and Active Directory (AD) logs. Their criteria are designed to detect anomalous insider behavior in email communications, such as sending large amounts of data to receivers who do not exist in the corporate namespace.

Franc's Proxy Log system: Franc's proxy log system focuses on identifying fraudulent network activity via proxy logs. Flows are created from proxy log events depending on source and destination IP addresses, ports, and protocol. To detect abnormal communication between users and domains, features are gathered and methods such as SVM and multiple instance learning (MIL) are utilized.

Integration of numerous Network Log Types: Correlating and analyzing numerous types of network logs at the same time to give enterprise-level prevention and detection capabilities is a new trend in insider threat detection. Prototype systems have been created to integrate a suite of detectors for several network log kinds into an ensemble or to evaluate features derived from numerous network log types using a machine learning framework.

PRODIGAL (PROactive Detection of Insider Threats with Graph Analysis and Learning) System: PRODIGAL (PROactive Detection of Insider Threats with Graph Analysis and Learning) is an example of a system that extracts over 100 features from various network logs, such as email, proxy, and LDAP. Multiple detectors use different algorithms to work with subsets of these characteristics. Not only may the PRODIGAL create warnings for aberrant observations, but also for uncommon patterns and circumstances, allowing for manual examination.

The Beehive System: The Beehive system, like the PRODIGAL, uses a machine learning architecture with logs from proxy, DHCP, VPN, and LDAP. It collects 15 characteristics for thousands of hosts and using PCA and k-means clustering to detect anomalies in regular corporate network activity.

Graph-Based System: To combat insider threats, a graph-based system is presented that models users' interactions with devices utilizing information from LDAP, proxy, email logs, and auxiliary sources. The isolation forest (IF) technique is used to detect suspicious users using graphs and sub-graphs.

Tuor et al.'s Deep Learning Scheme: Tuor et al. demonstrate the efficacy of deep learning in detecting insider threats. Their method pulls features from email and proxy logs and detects them in real time using deep neural networks (DNNs) and recurrent neural networks (RNNs).

REFERENCES

[1] Threat Assessment in Machine Learning based Systems: https://arxiv.org/abs/2207.00091

[2] "Atlas framework," MITRE Corporation. [Online]. Available: https://atlas.mitre.org/matrix

[3] M. S. K. W. K. D. P. M. C. W. S. M. W. R. D. W. B. E. Strom, J. A. Battaglia, "Finding cyber threats with att&ck-based analytics," The MITRE Corporation, Tech. Rep., 2017 https://www.mitre.org/sites/default/files/publications/16-3713-finding-cyber-threats%20with%20att%26ck-based-analytics.pdf

[4] L. N. Tidjon and F. Khomh, "Never trust, always verify: a roadmap for trustworthy ai?" 2022. [Online]. Available: https://arxiv.org/abs/2206.11981

[5] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson et al., "Extracting training data from large language models," in 30th USENIX Security Symposium (USENIX Security 21), 2021, pp. 2633–2650

[6] "Altas framework," MITRE Corporation. [Online]. Available: https://atlas.mitre.org/matrix

[7] Y. Lakhdhar and S. Rekhis, "Machine learning based approach for the automated mapping of discovered vulnerabilities to adversial tactics," in 2021 IEEE Security and Privacy Workshops (SPW), 2021, pp. 309–317

[8] R. S. Siva Kumar, M. Nystr ¨om, J. Lambert, A. Marshall, M. Goertzel, A. Comissoneru, M. Swann, and S. Xia, "Adversarial machine learning-industry perspectives," in 2020 IEEE Security and Privacy Workshops (SPW), 2020, pp. 69–75.

[9] A. Kuppa, L. Aouad, and N.-A. Le-Khac, "Linking cve's to mitre att&ck techniques," in The 16th International Conference on Availability, Reliability and Security, ser. ARES 2021. New York, NY, USA: Association for Computing Machinery, 2021

[10] "D3fend framework," MITRE Corporation. [Online]. Available: https://d3fend.mitre.org/

[11] R. Mogull, J. Arlen, F. Gilbert, A. Lane, D. Mortman, G. Peterson, and M. Rothman, "Security guidance for critical areas of focus in cloud computing v4.0," Cloud Security Alliance, 2017.

[12] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and counter-measures for regression learning," in 2018 IEEE Symposium on Security and Privacy (SP). IEEE, 2018, pp. 19–35.

[13] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, "The security of machine learning," Machine Learning, vol. 81, no. 2, pp. 121– 148, 2010.

[14] Detecting and Preventing Cyber Insider Threats: A Survey: https://ieeexplore.ieee.org/abstract/document/8278157

[15] S. S. C. Silva, R. M. P. Silva, R. C. G. Pinto, and R. M. Salles, "Botnets: A survey," Comput. Netw., vol. 57, no. 2, pp. 378–403, 2013.

[16] H. Choi, H. Lee, H. Lee, and H. Kim, "Botnet detection by monitoring group activities in DNS traffic," in Proc. IEEE Comput. Inf. Technol., Aizuwakamatsu, Japan, 2007, pp. 715–720.

[17] D. Zhao et al., "Botnet detection based on traffic behavior analysis and flow intervals," Comput. Security, vol. 39, pp. 2–16, Nov. 2013.

[18] J. B. Grizzard, V. Sharma, C. Nunnery, B. B. Kang, and D. Dagon, "Peer-to-peer botnets: Overview and case study," in Proc. HotBots, vol. 7. Cambridge, MA, USA, 2007, p. 1.

[19] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "BotMiner: Clustering analysis of network traffic for protocol-and structure-independent botnet detection," in Proc. USENIX Security Symp., vol. 5. San Jose, CA, USA, 2008, pp. 139–154.

[20] X. Sun, R. Torres, and S. Rao, "DDoS attacks by subverting membership management in P2P systems," in Proc. IEEE Workshop Secure Netw. Protocols, Beijing, China, 2007, pp. 1–6.

[21] Y. Xiang, K. Li, and W. Zhou, "Low-rate DDoS attacks detection and traceback by using new information metrics," IEEE Trans. Inf. Forensics Security, vol. 6, no. 2, pp. 426–437, Jun. 2011.

[22] A. Pathak, F. Qian, Y. C. Hu, Z. M. Mao, and S. Ranjan, "Botnet spam campaigns can be long lasting: Evidence, implications, and analysis," in Proc. ACM SIGMETRICS Perform. Eval. Rev., Seattle, WA, USA, 2009, pp. 13–24.

[23] H. Haddadi, "Fighting online click-fraud using bluff ads," ACM SIGCOMM Comput. Commun. Rev., vol. 40, no. 2, pp. 21–25, 2010.

[24] D. Plohmann and E. Gerhards-Padilla, "Case study of the miner botnet," in Proc. Int. Conf. Cyber Conflict, 2012, pp. 1–16.

[25] E. M. Hutchins, M. J. Cloppert, and R. M. Amin, "Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains," presented at the Leading Issues Inf. Warfare Security Res., vol. 1, 2011, p. 80.

[26] A. Liu, C. Martin, T. Hetherington, and S. Matzner, "A comparison of system call feature representations for insider threat detection," in Proc. 6th Annu. IEEE SMC Inf. Assurance Workshop, West Point, NY, USA, 2005, pp. 340–347.

[27] N. Virvilis and D. Gritzalis, "The big four—What we did wrong in advanced persistent threat detection?" in Proc. IEEE Availability Rel. Security (ARES), Regensburg, Germany, 2013, pp. 248–254.

[28] D. L. Costa et al., "An insider threat indicator ontology," SEI, Pittsburgh, PA, USA, Rep. CMU/SEI-007, 2016.

[29] M. Collins, Common Sense Guide to Mitigating Insider Threats, Carnegie-Mellon Univ., Pittsburgh, PA, USA, 2016.

[30] Y. Yu and T.-C. Chiueh, "Display-only file server: A solution against information theft due to insider attack," in Proc. 4th ACM Workshop Digit. Rights Manag., Tallinn, Estonia, 2004, pp. 31–39.

[31] X. Huang et al., "Cost-effective authentic and anonymous data sharing with forward security," IEEE Trans. Comput., vol. 64, no. 4, pp. 971–983, Apr. 2015.

[32] J. Baek, Q. H. Vu, J. K. Liu, X. Huang, and Y. Xiang, "A secure cloud computing based framework for big data information management of smart grid," IEEE Trans. Cloud Comput., vol. 3, no. 2, pp. 233–244, Apr./Jun. 2015.

[33] M. Garg. (2006). Sysenter Based System Call Mechanism in Linux 2.6. Accessed: Nov. 6, 2017. [Online]. Available: http://articles.manugarg.com/systemcallinlinux2_6.html

[34] A. A. E. Ahmed and I. Traore, "Anomaly intrusion detection based on biometrics," in Proc. 6th Annu. IEEE SMC Inf. Assur. Workshop, West Point, NY, USA, 2005, pp. 452–453.

[35] Y. Song, M. B. Salem, S. Hershkop, and S. J. Stolfo, "System level user behavior biometrics using fisher features and Gaussian mixture models," in Proc. IEEE Security Privacy Workshops (SPW), 2013, pp. 52–59.

[36] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A sense of self for Unix processes," in Proc. IEEE Security Privacy, 1996, pp. 120–128

[37] E. Eskin, W. Lee, and S. J. Stolfo, "Modeling system calls for intrusion detection with dynamic window sizes," in Proc. IEEE DARPA Inf. Survivability Conf. Expo. II, vol. 1. Anaheim, CA, USA, 2001, pp. 165–175.

[38] J. Hu, X. Yu, D. Qiu, and H.-H. Chen, "A simple and efficient hidden Markov model scheme for host-based anomaly intrusion detection," IEEE Netw., vol. 23, no. 1, pp. 42–47, Jan./Feb. 2009.

[39] A. K. Ghosh, J. Wanken, and F. Charron, "Detecting anomalous and unknown intrusions against programs," in Proc. IEEE Comput. Security Appl. Conf., Phoenix, AZ, USA, 1998, pp. 259–267.

[40] A. K. Ghosh, A. Schwartzbard, and M. Schatz, "Learning program behavior profiles for intrusion detection," in Proc. Workshop Intrusion Detection Netw. Monitor., Santa Clara, CA, USA, 1999, pp. 51–62.

[41] B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep learning for classification of malware system call sequences," in Proc. Aust. Joint Conf. Artif. Intell., 2016, pp. 137–149.

[42] S. A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion detection using sequences of system calls," J. Comput. Security, vol. 6, no. 3, pp. 151–180, 1998.

[43] R. Canzanese, S. Mancoridis, and M. Kam, "System call-based detection of malicious processes," in Proc. IEEE Softw. Qual. Rel. Security (QRS), Vancouver, BC, Canada, 2015, pp. 119–124.

[44] Y. Liao and V. R. Vemuri, "Use of k-nearest neighbor classifier for intrusion detection," Comput. Security, vol. 21, no. 5, pp. 439–448, 2002.

[45] Y. Liao and V. R. Vemuri, "Using text categorization techniques for intrusion detection," in Proc. USENIX Security Symp., vol. 12, 2002, pp. 51–59.

[46] T. Lane and C. E. Brodley, "An application of machine learning to anomaly detection," in Proc. 20th Nat. Inf. Syst. Security Conf., vol. 377. Baltimore, MD, USA, 1997, pp. 366–380.

[47] J. Shavlik and M. Shavlik, "Selection, combination, and evaluation of effective software sensors for detecting abnormal computer usage," in Proc. 10th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min., 2004, pp. 276–285.

[48] K. Berlin, D. Slater, and J. Saxe, "Malicious behavior detection using windows audit logs," in Proc. 8th ACM Workshop Artif. Intell. Security, 2015, pp. 35–44.

[49] X. Sun, R. Torres, and S. Rao, "DDoS attacks by subverting membership management in P2P systems," in Proc. IEEE Workshop Secure Netw. Protocols, Beijing, China, 2007, pp. 1–6.

[50] B. Mukherjee, L. T. Heberlein, and K. N. Levitt, "Network intrusion detection," IEEE Netw., vol. 8, no. 3, pp. 26–41, May/Jun. 1994.

[51] K. Wang and S. J. Stolfo, “Anomalous payload-based network intrusion detection,” in Proc. Int. Workshop Recent Adv. Intrusion Detection, 2004, pp. 203–222.

[52] (1999). SIGKDD Blog. KDD Cup 1999: Computer Network Intrusion Detection. Accessed: Jun. 11, 2017. [Online]. Available: http://www.kdd.org/kdd-cup/view/kdd-cup-1999

[53] M. Tavallaee, E. Bagheri, W. Lu, and A.-A. Ghorbani, “A detailed analysis of the KDD cup 99 data set,” in Proc. 2nd IEEE Symp. Comput. Intell. Security Defence Appl., Ottawa, ON, Canada, 2009, pp. 1–6.

[54] J. Bau, E. Bursztein, D. Gupta, and J. Mitchell, “State of the art: Automated black-box Web application vulnerability testing,” in Proc. IEEE Security Privacy (SP), Berkeley, CA, USA, 2010, pp. 332–345.

[55] R. Villamarín-Salomón and J. C. Brustoloni, “Identifying botnets using anomaly detection techniques applied to DNS traffic,” in Proc. IEEE Consum. Commun. Netw. Conf., 2008, pp. 476–481.

[56] G. Gu, J. Zhang, and W. Lee, “BotSniffer: Detecting botnet command and control channels in network traffic,” in Proc. 15th Annu. Netw. Distrib. Syst. Security Symp., 2008.

[57] W. T. Strayer, R. Walsh, C. Livadas, and D. Lapsley, “Detecting botnets with tight command and control,” in Proc. IEEE Conf. Local Comput. Netw., Tampa, FL, USA, 2006, pp. 195–202.

[58] W. T. Strayer, D. Lapsely, R. Walsh, and C. Livadas, “Botnet detection based on network behavior,” in Botnet Detection. Boston, MA, USA: Springer, 2008, pp. 1–24.

[59] A. Al-Bataineh and G. White, “Analysis and detection of malicious data exfiltration in Web traffic,” in Proc. Malicious Unwanted Softw. (MALWARE), 2012, pp. 26–31.

[60] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, and C. Kruegel, “Disclosure: Detecting botnet command and control servers through large-scale netflow analysis,” in Proc. 28th Annu. Comput. Security Appl. Conf., Orlando, FL, USA, 2012, pp. 129–138.

[61] Y. Liu et al., “SIDD: A framework for detecting sensitive data exfiltration by an insider attack,” in Proc. Syst. Sci., 2009, pp. 1–10.

[62] T. Fawcett, “ExFILD: A tool for the detection of data exfiltration using entropy and encryption characteristics of network traffic,” Ph.D. dissertation, Dept. Elect. Comput. Eng., Univ. Delaware, Newark, DE, USA, 2010.

[63] B. E. Binde, R. McRee, and T. J. O’Connor, “Assessing outbound traffic to uncover advanced persistent threat,” Singapore, SANS Inst., White Paper, 2011.

[64] N. Villeneuve and J. Bennett, Detecting APT Activity With Network Traffic Analysis, Trend Micro Incorporat., Tokyo, Japan, 2012

[65] J. Zhang, C. Chen, Y. Xiang, W. Zhou, and Y. Xiang, “Internet traffic classification by aggregating correlated naive Bayes predictions,” IEEE Trans. Inf. Forensics Security, vol. 8, no. 1, pp. 5–15, Jan. 2013.

[66] Y. Wang et al., “Internet traffic classification using constrained clustering,” IEEE Trans. Parallel Distrib. Syst., vol. 25, no. 11, pp. 2932–2943, Nov. 2014.

[67] The CERT Division. The Cert Insider Threat Database | The Cert Division. Accessed: Apr. 11, 2017. [Online]. Available: https://www.cert.org/insider-threat/research/database.cfm

[68] Computer Immune Systems—Data Sets and Software, Dept. Comput. Sci., Univ. New Mexico, Albuquerque, NM, USA, accessed: Apr. 11, 2017. [Online]. Available: https://www.cs.unm.edu/immsec/systemcalls.htm

[69] G. Creech and J. Hu. (2013). The ADFA Intrusion Detection Datasets. Accessed: Jun. 11, 2017. [Online]. Available: https://www.unsw.adfa.edu.au/australian-centre-for-cyber-security/cybersecurity/ADFA-IDS-Datasets/

[70] BlackBerry’s Cylance AI Prevents Terminator EDR Killer: https://blogs.blackberry.com/en/2023/06/blackberry-cylance-ai-prevents-terminator-edr-killer

[71] 2023-05-31 // SITUATIONAL AWARENESS // Spyboy Defense Evasion Tool Advertised Online: https://www.reddit.com/r/crowdstrike/comments/13wjrgn/20230531_situational_awareness_spyboy_defense/

[72] BlackBerry's Cylance AI Antivirus Defeated by Embarrassing Universal Bypass: https://www.tomshardware.com/news/blackberry-cylance-ai-antivirus-security-vulnerability,39984.html