Results in Engineering 25 (2025) 104078

Contents lists available at ScienceDirect

Results in Engineering

journal homepage: www.sciencedirect.com/journal/results-in-engineering

Research paper

# Enhancing cybersecurity incident response: AI-driven optimization for strengthened advanced persistent threat detection

Gauhar Ali *, Sajid Shah, Mohammed ElAffendi

*EIAS Data Science Lab, College of Computer and Information Sciences, Prince Sultan University, Riyadh, 11586, Saudi Arabia*

ARTICLE INFO

*Keywords:*
Cybersecurity incident response
Machine learning
Security information and event management

ABSTRACT

The Cyber Security Incident Response Team executes its critical duties in the Centralized Security Operation Centers. Its primary target is to protect organizational resources from all types of Advanced Persistent Threats (APTs) and attacks while making the correct judgments at the right time. Despite the great efforts and the efficient tools, the incident response mechanism faces two significant challenges. The generation of large amounts of alerts with huge rates of false positives per unit of time and the time-consuming manual expert engagement in the post-alert decision phase. The main aim of this research study is to investigate the potential of Machine Learning techniques in solving the above challenges. This study proposes six event detection modules to identify 14 different types of malicious behavior using Splunk. The available APT dataset is extended by adding more alert types. Then, the machine learning techniques i.e., Random Forest, and XGBoost, are applied to improve the decision process and reduce the false positive alerts of the Security Information and Event Management system. The proposed model outperforms the state-of-the-art techniques by predicting an accuracy of 99.6%.

## 1. Introduction

The cyber incident is analogous to a catastrophe, which is unexpected, sudden, and crucial. If the cyber-incident is not quickly alleviated, it causes severe damage. These damages often worsen when a cyber incident is unmanageable for longer periods. For example, the adversary can destroy or withdraw confidential data. Moreover, in a recently published report [1], 20 percent of the companies report a loss in revenue, business opportunities, and customer decrease due to security incidents.

Companies implement incident response procedures to identify, maintain, and remove cybersecurity incidents. The incident response consists of a plan of action preparation, incident detection, limiting the adversary, removal of the adversary, healing from the incident, and post-accident tasks [2]. The incident Response Frameworks i.e., SysAdmin, Audit, Network, and Security (SANS) [3], and National Institute of Standards and Technology (NIST) [4] help companies in the formation of systematic incident response procedures. However, companies use traditional Security Information and Event Management (SIEM) systems to gather and evaluate security events in their Security Operations Centers (SOCs) [5]. These tools often struggle to detect and respond efficiently to multi-staged threats like Advance Persistent Threat (APT) [6]. Unlike conventional attacks, APT uses sophisticated strategies, covert methods, and persistent focus on the victim for an extended period [7].

The existing incident and response approaches face significant challenges: 1) the generation of huge streams of alerts per unit of time by Security Operation Centers (SOC), making it hard or impossible to handle APT attacks promptly. Thus, many attacks remain unidentified or detected too late; 2) a large number of false positive APT attack alerts consume resources, delay genuine threat detection, overload security experts, and ruin trust in incident and response approaches, and negatively impact its efficacy; 3) the manual intervention by experts, in the post-alert phase of SOC, can delay the decision process and raise the chance of human error [8].

ML methods have demonstrated potential in detecting APT attacks. An APT attack consists of various stages i.e., reconnaissance, initial compromise, persistence, lateral movement, data exfiltration, and command & control. Understanding this lifecycle is crucial for effective detection and response to APT attacks. However, the unavailability of extensive labeled datasets covering the whole APT lifecycle is a challenge [9–11]. Several research studies have developed ML-based methods to detect APT. Most of the research studies, [12–14], have focused on detecting four stages of APT attacks, while some studies, [15,16], focused on detecting 1 stages of APT attack. Some studies recommend the use of

* Corresponding author.
*E-mail addresses:* gali@psu.edu.sa (G. Ali), sshah@psu.edu.sa (S. Shah), affendi@psu.edu.sa (M. ElAffendi).

https://doi.org/10.1016/j.rineng.2025.104078
Received 24 October 2024; Received in revised form 5 January 2025; Accepted 17 January 2025

Available online 21 January 2025

multi-layer analysis techniques, combining various detection methods to improve accuracy and decrease false positive alerts [17].

A novel framework is proposed to enhance security event identification and reduce false positive alerts. The proposed framework includes many core elements to solve the above-mentioned problems. Firstly, six event detection modules are proposed to identify more malicious behaviors using a specific set of Indicators of Compromises (IoCs). These modules can detect 14 different types of malicious behavior. Secondly, the Advance Persistent Threat (APT) alert dataset [12] is extended by adding additional alert types, giving the system a large number of APT indicators. Thirdly, Machine Learning (ML) techniques i.e., Random Forest and Extreme Gradient Boosting (XGBoost) are trained and tested using the extended APT dataset. The proposed model improved the APT detection accuracy up to 99.6% and reduced false positive alerts generated by SIEM. The main contributions of the research study are as follows.

- To detect APT in the early stage, we implemented behavior detection modules and extended the available dataset with new malicious behavior alerts.
- To reduce false positive APT alarms, we developed a sophisticated ML-based prediction model, resulting in a high accuracy of 99.6% and a significantly lower False Positive Rate (FPR).
- To automate the incident response process by reducing manual intervention in the post-alert decision using ML techniques.

The rest of the paper is structured as follows. The preliminaries and related literature are summarized in Section 2. In section 3, we detailed the proposed framework. In Section 4, we present the implementation and evaluation of the result in detail. Similarly, the conclusion of this work is drawn in section 5.

## 2. Preliminaries and related literature

The following subsections discuss SIEM, ML, and literature related to CyberSecurity Incident Response optimization Using ML techniques.

### 2.1. Overview of security information and event management

SIEM was introduced by Mark Nicolett and Amrit Williams in the technical report of Gardner in 2005 [18]. SIEM combines Security Information Management (SIM) and Security Event Management (SEM). The SIM is used to store, analyze, and report log events. Similarly, SEM is used to monitor and correlate real-time events. Therefore, SIEM can be defined as a tool that gathers, analyzes, and validates log data [19]. Moreover, SIEM uses either rule-based or statistical events correlation and provides the log data in human-readable format i.e., reports and alerts. The core functionalities of SIEM are discussed in the following subsection.

- Log Collection: Network Devices like switches, hosts, and firewalls, generate log data. There are two methods defined for log data collection i.e., agent-based and agent-less [20]. In the agent-based method, an intermediary agent collects log data and then forwards it to the next module. Similarly, in the Agent-less method, there is no intermediate agent and the server gets data directly from the servers.
- Log Aggregation: It is a procedure for gathering, analyzing, and capturing structured data from the log data generated by network devices. In the literature [20], two methods are used for log data collection i.e., push and pull. i) Push: The network devices like switches, and firewalls, send logs to the servers. ii) Pull: The servers drag the log data from the network devices.
- Parsing: It is a procedure for transforming the log data into structured data. In a network environment, multiple parsers are used to transform log data received from multiple devices such as firewalls and server switches.
- Normalization: SIEM receives log data from multiple sources and converts it into one standard reduced format. It removes the redundant logs from the log data.
- Threat analysis: SIEM identifies adversaries by comparing log data with the records in a known threat database. During threat analysis, SIEM uses statistical methods to identify different threat patterns.
- Response: SIEM is a very efficient tool because it can detect malicious activity quickly before it can cause serious damage. It has built-in predefined real-time alerts and a notification system.
- Reporting: SIEM tools contain different custom reporting templates for data visualization. Moreover, the data collected by SIEM is available to data analysts and experts for investigation.

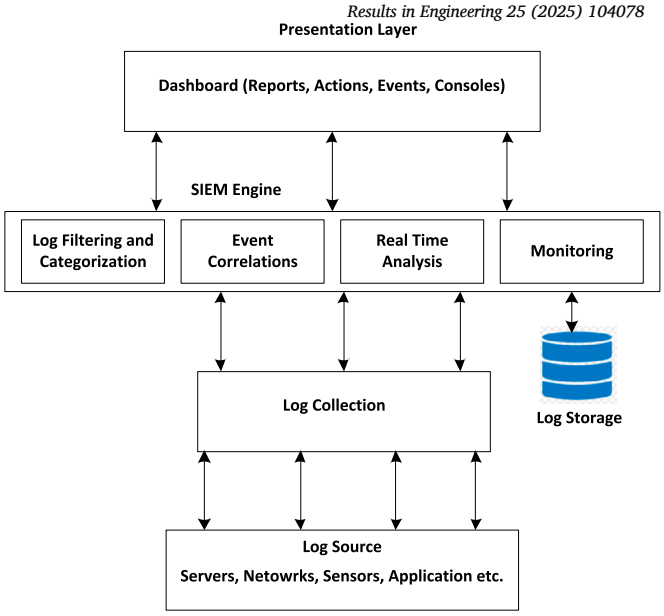


**Fig. 1.** High-level SIEM Architecture.

#### 2.1.1. SIEM's architecture and components

The layered architecture of SIEM is depicted in **Fig. 1**. It comprises a SIEM Engine, log collection, log sources, and presentation layer [20]. The SIEM architecture collects logs generated by different sources like servers, firewalls, and applications. Then, the SIEM engine filters the duplicate events and correlates the filtered events to find malicious activities. In addition, the analysis result is displayed in the form of a report or alert in the presentation layer. The SIEM architecture also stores the log data for future analysis. The SIEM provides functionalities such as threat detection, log storage, incident response, and threat reporting.

Gartner's annual report on the SIEM Magic Quadrant [21] has classified SIEM solutions as niche, challengers, leaders, or visionary. Splunk is a well-known SIEM tool recognized as a leader by the Gartner report.

#### 2.1.2. Splunk: implementations of SIEM

Splunk processes log data and present them in a human-readable format. It consists of two modules i.e., the processing module and management module [22]. A high-level Splunk architecture is present in **Fig. 2**. Furthermore, the processing module consists of forwarders, Indexers, and search heads [23]. The forwarders gathered log data from the network devices and forwarded it to other Splunk processing components. Similarly, an indexer is used to index and store log data. Furthermore, the search head analyzes the log data and generates visual graphs and reports. Similarly, the deployment server, license master, Indexer cluster master node, search head cluster deployer, and monitoring console are the sub-components of the management module.

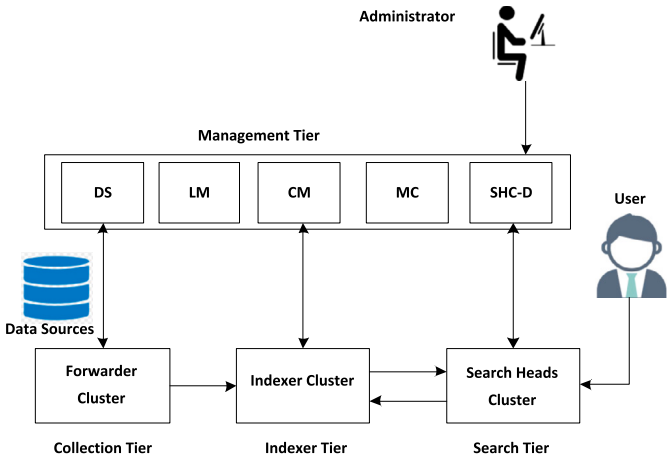


**Fig. 2.** Splunk Architecture.

Splunk can be deployed in two ways i.e., single server deployment and distributed deployment [24]. The single server deployment is used either to test Splunk components or for learning purposes. Similarly, in a multi-server deployment, the Splunk components are installed on several servers in a distributed manner.

### 2.2. Introduction to machine learning

ML is an Artificial Intelligence (AI) application that enables a system to learn from its experiences. ML identifies patterns in a given data and generates predictions. ML has gained much attention due to its applications in diversified domains like Image processing, digital forensics, and incident response. [25]. ML is also known as learning from data. Conventional softwares are based on deduction methods whereas ML is based on induction methods. Therefore, an ML algorithm automatically discovers rules by looking at many examples, and these practices can then be applied to other similar data. The following are the well-known ML models in the existing literature.

#### 2.2.1. Random forest

Random Forest [26] is a powerful ML classifier that combines the predictions of multiple decision trees. The working of random forest can be divided into the following three steps. 1) Random Forest begins by extracting random samples from the original data by using either sample with replacement or random feature selection method. This randomness in the training phase protects the model from overfitting. 2) Then, a decision tree is generated for each sample. 3) Finally, the Random Forest combines the predictions of each decision tree using either classification or regression. The Random Forest is well known because of its simplicity, usability, and potential to deal with large-size datasets.

#### 2.2.2. XGBoost

XGBoost [27] is an improved version of the gradient boosting technique to reduce the loss function. It is an ensemble ML model that combines numerous decision trees to generate a strong prediction model. Unlike random forests, which build trees separately, boosting builds trees consecutively, with each new tree attempting to remedy the flaws committed by the preceding ones. Moreover, it uses L1 and L2 regularization techniques to avoid overfitting. XGBoost classifier has attracted researchers due to parallel processing, accuracy, and ability to work with large-size datasets.

### 2.3. Related works

Cybersecurity has strongly accepted ML in various fields i.e., botnet identification [28], malware [29], and intrusion detection [30][31][32]. This research focuses on the application of ML to enhance and automate decision-making in the post-alert stage of SIEM systems. The related works are discussed in the following paragraphs.

In [12], the authors have developed an efficient ML-based model to detect APT attacks quickly and accurately. The model consists of three core phases i.e., APT attacks detection, correlation, and prediction. The model detected the APT attacks with a true positive rate of 81.8% and a false positive rate of 4.5%. However, the accuracy is still low and needs to be improved. Similarly, the authors, in [13], have proposed a framework to protect organization workstations and servers from APTs. Agents are used to monitor these computers. The agents perform two functions i.e., security violation identification and alert generation. The proposed TerminAPTor module correlates the alerts generated by agents using Information Flow Tracking (IFT). This IFT is used to detect APTs. However, the authors simulated only two APT scenarios which resulted in high false positive alerts. Similarly, the proposed context-based framework introduces a conceptual attack model called the attack pyramid [14]. It has defined sensitive data at the top level and the lower levels consist of a platform for recording events related to the APTs. However, sufficient knowledge is required to set up the framework.

The authors, in [33], discussed the deployment of ML to categorize and predict SOC alerts. The proposed solution aims to reduce human analyst intervention by automating the process of threat detection using an ML model. The proposed ML model depends on the response from five senior SOC experts. Each expert shares his experience in handling the alerts. They used a Random Forest Classifier while the dataset was generated using SIEM. Although the aim was to automate the decision-making concerning false positive alerts, the final decision was still taken by the experts. Similarly, the authors, in [34], have proposed an efficient ML-based threat detection model with minimized false positive alerts. To select standardized threat use cases, several MITRE's taxonomies (ATT&CK, CAPEC, and MAEC) were examined. KMeans is used to analyze uploaddownload data traffic, because of its simplicity and fast grouping ability. Linear regression and Random Forest were used to identify malicious activities while SVM was used to identify malicious child and parent activities. Moreover, the authors have claimed that existing intrusion detection mechanisms are inefficient and proposed context-aware IDS by integrating ML and DRL models [35]. The robustness of the proposed model is enhanced by integrating the denoising autoencoder. The proposed model is trained and tested using NSL-KDD, UNSW-NB15, and AWID datasets. The authors claimed that the proposed model showed better accuracy and less FPR as compared to their counterparts. However, the proposed model is not tested in real-world scenarios. Moreover, a quick ML-based incident response approach is proposed in [36]. The authors generated their APT dataset by analyzing 2.5 million log events. The dataset consists of a thousand vectors, where each vector contains nine features labeled as either positive or negative. ZeroR, OneR, SVM, NaiveBayes, J48, and Random Forest classifiers were trained and tested. The Random Forest attained a high accuracy of 95.54% using the extracted APT dataset.

Moreover, in [15], the authors have proposed a spam and junk mail detection mechanism based on Bayesian spam filtering. They identified APT's behavioral patterns and proposed a novel self-destructive method to protect individual or organizational data from APTs. Furthermore, several classification models i.e., random forest, KNN, XGBoost, are used on four different datasets i.e., e CSE-CIC-IDS2018, CIC-IDS2017, NSL-KDD, and UNSW-NB15, to detect APT at the early stage [16]. However, Random Forest and XGBoost were selected as the base classifiers. Then, the simple classifier aggregator is used to merge the predictions of the two classifiers. The proposed hybrid model was then trained and tested on the four different datasets. The summary of the existing related works is presented in **Table 1**.

Finally, the state-of-the-art APT detection mechanisms face several challenges i.e., high false positive rate, detection of APT in early stages,

**Table 1**
The summary of the Existing Related Works.

| References | APT Life Cycle Detection | AI Models | Lowest FPR | Highest Accuracy | Limitations |
|---|---|---|---|---|---|
| [12] | Complete | Decision Tree, SVM, KNN, Ensemble Classifier | 4.5% | 84.8% | The blacklist based detection modules require continuous update. Moreover, the accuracy is lower. |
| [13] | Complete | No ML model used | High | Not calculated | Information Flow Tracking is used to detect APTs, however, the FPR is high. |
| [14] | Complete | No ML model used | 27.88% | Not calculated | sufficient knowledge is required to setup the mechanism. |
| [33] | No | Random Forest | Not calculated | 98.5% | The post-alert decision is not automated, it involves the intervention of security experts. |
| [34] | Partial | SVM, Random Forest | Not calculated | Not calculated | The process anomaly detection is presented formally using One-Class SVM. However, the paper lacks ML based experimental results. |
| [35] | No | Deep Q-network | 0.35% | 96.12 | The proposed model enable to detect APTs. |
| [36] | No | ZeroR, OneR, NaiveBayes, SVM, J48, RandomForest | Not calculated | 95.5% | The authors did not discuss the FPR of the proposed model. |
| [15] | Partial | NaiveBayes | Not calculated | 87% | The authors did not discuss the FPR of the proposed model. |
| [16] | Partial | Random Forest, XGBoost | 0.12% | 99.91% | This study used the existing datasets poses a challenge, because these datasets might not contain the latest attack scenarios. |

**Table 2**
Notations.

| Notation | Description |
|---|---|
| $\mathcal{IP}_j$ | jth host IP address. |
| $\mathcal{USR}_i$ | ith user account. |
| $\mathcal{EVENT}_k$ | kth event, recorded in the log file. |
| $\mathcal{ALERT}_i$ | alert generated by ith infected host. |
| $\mathcal{MD}$ | Malicious Domain. |
| $\mathcal{MIP}$ | Malicious IP address. |
| $\mathcal{MDNS}$ | Malicious DNS. |
| $\mathcal{MSSL}$ | Malicious SSL Certificate. |
| $\mathcal{ME}$ | Malicious Execution File. |
| $\mathcal{UFT}$ | Unusual File Transfer. |
| $\mathcal{MH}$ | Malicious Hash. |
| $\mathcal{SL}$ | Various failed attempt before successful login. |
| $\mathcal{SDA}$ | Sensitive Data Access. |
| $\mathcal{CUP}$ | Change User Privileges |
| $\mathcal{NS}$ | Network Scan |
| $\mathcal{UNT}$ | Unusual Network Traffic |
| $\mathcal{TOR}$ | Tor connection with the remote server |
| $\mathcal{USB}$ | Unusual User Behavior |
| $\mathcal{MDS}$ | Set of Malicious Domain. |
| $\mathcal{MIP}$ | Set of Malicious IP address. |
| $\mathcal{USBS}$ | Set of Unusual User Behavior. |
| $\mathcal{UFBS}$ | Set of Unusual File Behavior. |
| $\mathcal{UNTS}$ | Set of Unusual Network Traffic. |
| $\mathcal{USBS}$ | Set of Unusual System Behavior. |
| $S\_IP$ | Source IP address. |
| $D\_IP$ | Destination IP address. |
| $S\_Port$ | Source Port Number. |
| $D\_Port$ | Destination Port Number. |

new APT scenarios, and behavior. To address those challenges, a novel mechanism is proposed for APT detection.

## 3. Overview of the proposed framework

The proposed architecture uses an advanced ML Technique for Cybersecurity Incident Response (CIR) optimization. The proposed architecture consists of an Event Logging, Event Detection Module, Alert Correlation Module, and Reporting as shown in **Fig. 3**.

### 3.1. Notations

The notations used in this section are given in **Table 2**.

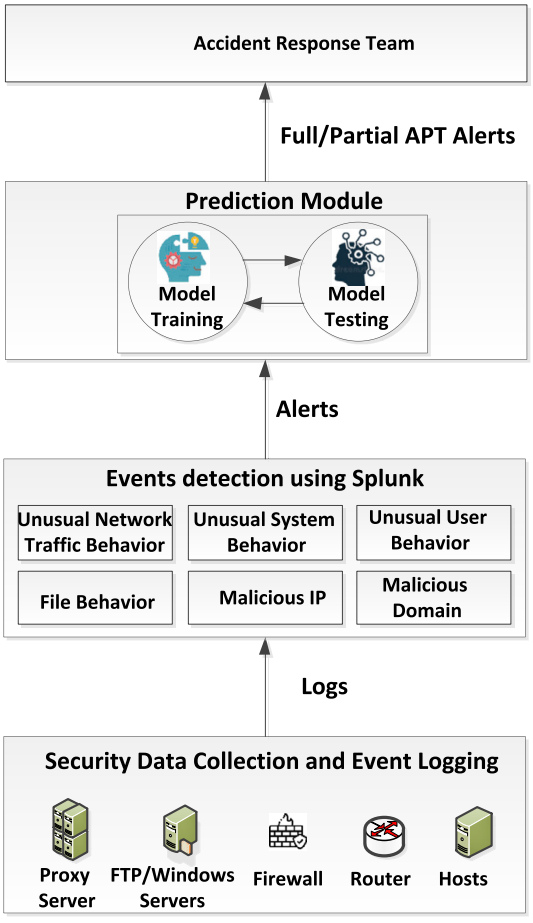


**Fig. 3.** Proposed High-level Architecture.

### 3.2. Event logging

The following logs were collected from the hosts. Splunk collects log files from different sources i.e., Syslog, MySQL Logs, Apache Access Logs, DNS logs, FTP server Logs, and Authentication Logs. The components of an event are given in **Eq. (1)**.

**Table 3**
Splunk query to Detect Unusual Traffic.

```
index = * source = "C:\\firewall.log"
((S_Port NOT IN ("21", "22", "80", "443", "445", "135")) OR
(proto NOT IN ("TCP", "UDP")) AND
D_IP IN (compromised_IPs.txt))
| stats count by S_IP, D_IP, D_Port, proto
| where count > 5
| sort - count
| head 10
```

$$\mathcal{EVENT}_j = \{S\_IP, D\_IP, S\_port, D\_port, timestamp, type, infected\_host\} \quad (1)$$

Formally, we defined a log file as a set of events i.e., $\mathcal{LOG}$ set records all the events generated in a system as shown in **Eq. (2)**.

$$\mathcal{LOG} \supseteq \cup_{j=1}^{n} Event_j \quad (2)$$

### 3.3. Event detection module

The event detection module is proposed to detect malicious behavior in a log of events. Splunk is used to collect log files from different sources like firewalls, routers, hosts, and servers. Splunk is selected due to its ability for real-time threat detection and response, customization, and advanced analytics. Splunk examines these logs against malicious behavior indicators to detect malicious behavior. If a malicious behavior is detected then Splunk generates an alert.

We proposed six sub-modules for malicious behavior detection i.e., unusual network traffic, unusual system behavior, user behavior, file behavior, malicious IP, and malicious domain as shown in **Fig. 3**. Each module generates an alert when a specific behavior is detected.

#### 3.3.1. Unusual network traffic (UNT) behavior detection

UNT behaviors include port scan, presence of tor connection, and data transfer. The Splunk query is used to detect Unusual Network Traffic. A sample Splunk query as shown in **Table 3**. Unusual Network Traffic detection can be an indicator of several stages in the APT life cycle i.e., Command and Control, Exfiltration, and lateral movement. 1) Command and Control (C2): An unusual network traffic pattern can be produced when the adversary uses non-standard communication methods to reach the compromised host. 2) Exfiltration: Extracting sensitive data from the compromised host can increase network traffic. 3) Lateral Movement: A spike in network traffic can occur due to unusual network activity used by attackers to propagate across the network. We defined a Unusual Network Traffic Set ($\mathcal{UNTS}$) that contains all those events that cause unusual network traffic as shown in **Eq. (3)**.

$$\mathcal{UNTS} = \{\mathcal{UNT}_0 \cup \mathcal{UNT}_1 \cup \mathcal{UNT}_2 \cup .....\mathcal{UNT}_n\} = \cup_{i=1}^{n} \mathcal{UNT}_i \quad (3)$$

#### 3.3.2. Unusual system behavior detection

Unusual system behaviors i.e., unexpected system crashes, error messages, and modifications to system configuration are IoCs for many stages in the APT life cycle. The following indicators were used to detect unusual system behavior of APT attacks. 1) Indicators for the Data Ex-filtration phase: Malicious activities can result in unusual system behaviors like unexpected system crashes and error messages. 2) Indicators for lateral movement phase: Adversaries may install malicious software like backdoors, and rootkits, which can result in a modification to system configurations or new process creation. A sample Splunk query is given in **Table 4**. It uses event id 101 to detect system errors i.e., INACCESSIBLE_BOOT_DEVICE, KERNEL_MODE_EXCEPTION_NOT_HANDLED, PAGE_ F- AULT_IN_NONPAGED_AREA, UNEXPECTED_KERNEL_M- ODE_TRAP, DRIVER_IRQL_NOT_LESS_OR_EQUAL. An attacker can crash a system by exploiting vulnerabilities to execute these error codes. $\mathcal{USB}$ is a superset of all the events of unusual system behaviors as shown in **Eq. (4)**.

**Table 4**
Splunk query to Detect System Behavior.

```
index = * sourcetype = wineventlog
| where EventCode = 1001
| rex field = EventData "(? < Error_Code > 0x[A-Fa-f0-9]+ )"
| where Error_Code IN ("0x0000007B", "0x0000008E", "0x00000050",
"0x0000007F", "0x000000D1")
```

**Table 5**
Splunk query to Detect Five Failed Attempts before Successful Login.

```
index = * source = "C:\\xampp\\FileZillaFTP\\Logs\\FileZilla
Server.log"
| stats count(eval(login = 530)) as failures count(eval(login = 230))
as successes by host
| where (successes ≤ 1 AND failures ≥ 5)
```

**Table 6**
Splunk Query to Detect File Creation and Deletion.

```
index = * source = "Microsoft-windows-sysmon/operational"
(Eventcode IN ("11", "23", "29") AND host = "fileserver")
| table host file_creation_time location
| sort _time
```

$$\mathcal{USB} = \{\mathcal{USB}_0 \cup \mathcal{USB}_1 \cup \mathcal{USB}_2 \cup .....\mathcal{USB}_n\} = \cup_{j=1}^{n} \mathcal{USB}_j \quad (4)$$

#### 3.3.3. Unusual user behavior detection

It recognized unusual user behavior from the logs collected by Splunk using the following indicators. 1) Indicators at Reconnaissance Phase: Unusual network scans. 2) Indicators at Initial Compromise Phase: At this phase, the following indicators are used i.e., Unusual pattern of failed login attempts, Failed login attempts from the same IP address into many accounts, and successfully logging into an account followed by numerous failed login attempts. 3) Indicators at Lateral Movement Phase: Unusual activity from privileged accounts, change in user privileges, and access to sensitive data. A sample Splunk query is given in **Table 5** for detecting failed attempts before successful login. The $\mathcal{UUB}$ is a superset of all the events of unusual user behavior as shown in **Eq. (5)**.

$$\begin{aligned}\mathcal{UUB} = \{&(\mathcal{SL}_0 \cup \mathcal{SL}_1 \cup \mathcal{SL}_2 \cup .....\mathcal{SL}_n) \\ &\cup (\mathcal{SBA}_0 \cup \mathcal{SBA}_1 \cup \mathcal{SBA}_2 \cup .....\mathcal{SBA}_n)\} \\ &\cup (\mathcal{CUP}_0 \cup \mathcal{CUP}_1 \cup \mathcal{CUP}_2 \cup .....\mathcal{CUP}_n)\} \\ &\cup (\mathcal{NS}_0 \cup \mathcal{NS}_1 \cup \mathcal{NS}_2 \cup .....\mathcal{NS}_n)\} \\ &= \{(\cup_{i=1}^{n} \mathcal{SL}_i) \cup (\cup_{j=1}^{n} \mathcal{SBA}_j) \cup (\cup_{k=1}^{n} \mathcal{CUP}_k) \cup \\ &\quad (\cup_{l=1}^{n} \mathcal{NS}_l)\}\end{aligned} \quad (5)$$

#### 3.3.4. File behavior detection

The following indicators of file behavior are used to identify an APT. 1) File execution: APTs often use malicious files that execute upon being opened. 2) File creation and deletion: APTs may create or delete new unexpected files. 3) File encryption: APTs may use encryption to keep their task undetected. The system logs are monitored for unexpected file encryption to detect APT. 4) File transfer: The system logs are monitored for unexpected file transfers between network hosts. A sample Splunk query is given in **Table 6** for the detection of malicious file creation and deletion by a particular host. The Unusual File Behavior Set $\mathcal{UFBS}$ is defined as a superset of all the events generated in response to unusual File behavior i.e., malicious domain, unusual file transfer, malicious hash as given in **Eq. (6)**.

**Table 7**
Splunk Query to Detect Malicious Domain.

```
index = * source = "C:\\Windows\System32\dns"
| stats count by reformattedDomain
| lookup malwareDomain.csv domain AS reformattedDomain
| eval domainmatch = if(reformatedDomain = = domain, "malicious",
"benign")
```

$$\begin{aligned}\mathcal{UFB} =& \{(\mathcal{MD}_0 \cup \mathcal{MD}_1 \cup \mathcal{MD}_2 \cup .....\mathcal{MD}_n) \\ & \cup(\mathcal{UFT}_0 \cup \mathcal{UFT}_1 \cup \mathcal{UFT}_2 \cup .....\mathcal{UFT}_n)\} \\ & \cup(\mathcal{MH}_0 \cup \mathcal{MH}_1 \cup \mathcal{MH}_2 \cup .....\mathcal{MH}_n)\} \\ =& \{(\cup_{j=1}^{n}\mathcal{MD}_i) \cup (\cup_{j=1}^{n}\mathcal{UFT}_j) \cup (\cup_{k=1}^{n}\mathcal{MH}_k)\}\end{aligned} \tag{6}$$

### 3.3.5. Malicious IP detection

This module uses the following indicators for malicious IP addresses. 1) Known malicious IP address. 2) Unusual increase in network traffic from an IP. 3) Communication with C2 server. A malicious IP Set $\mathcal{MIPS}$ is defined that contains blacklisted IP addresses as shown in **Eq. (7)**.

$$\begin{aligned}\mathcal{MIP} =& \{\mathcal{MIP}_0 \cup \mathcal{MIP}_1 \cup \mathcal{MIP}_2 \cup .....\mathcal{MIP}_n\} \\ =& \cup_{j=1}^{n}\mathcal{MIP}_j\end{aligned} \tag{7}$$

### 3.3.6. Malicious domain detection

The proposed framework used the following indicators to detect malicious Domains. 1) Similar domains that were used in previous APT attacks. 2) Only TOR-accessible domains. 3) malicious DNS. 4) Newly registered domains. 5) Domains having malicious SSL certificates. A sample Splunk query is given in **Table 7**. A Malicious Domain Set ($\mathcal{MD}$) is defined as a superset of all the events generated in response to malicious domain behavior i.e., malicious domain, malicious DNS, malicious SSL Certificate as shown in **Eq. (8)**.

$$\begin{aligned}\mathcal{MD} =& \{(\mathcal{MD}_0 \cup \mathcal{MD}_1 \cup \mathcal{MD}_2 \cup .....\mathcal{MD}_n) \\ & \cup(\mathcal{MDNS}_0 \cup \mathcal{MDNS}_1 \cup \mathcal{MDNS}_2 \cup .....\mathcal{MDNS}_n)\} \\ & \cup(\mathcal{MSSL}_0 \cup \mathcal{MSSL}_1 \cup \mathcal{MSSL}_2 \cup .....\mathcal{MSSL}_n)\} \\ =& \{(\cup_{i=1}^{n}\mathcal{MD}_i) \cup (\cup_{j=1}^{n}\mathcal{MDNS}_j) \cup (\cup_{k=1}^{n}\mathcal{MSSL}_k)\}\end{aligned} \tag{8}$$

### 3.3.7. Alert ($\mathcal{ALERT}$)

In the proposed architecture $\mathcal{ALERT}$ consists of the following attributes as shown in **Eq. (9)**. However, **Eq. (10)** shows the different types of alerts generated by the proposed system.

$$\begin{aligned}\mathcal{ALERT} =& \{S\_IP, D\_IP, S\_port, D\_port, timestamp, activity, \\ & alter\_type, C\_hostand stage\}\end{aligned} \tag{9}$$

$$\begin{aligned}alert.type =& \{alert_{MD} \vee alert_{ME} \vee alert_{MH} \vee alert_{MDNS} \\ & \vee alert_{SL} \vee alert_{IP} \vee alert_{SSL} \vee alert_{NS} \\ & \vee alert_{SDA} \vee alert_{CUP} \vee alert_{UNT} \vee alert_{USB} \\ & \vee alert_{tor} \vee alert_{UFB}\}\end{aligned} \tag{10}$$

### 3.3.8. Alert set ($\mathcal{AS}$)

$\mathcal{AS}$ contains all the alerts generated by the system in response to unusual behaviors.

$$\mathcal{AS} \supseteq \cup_{j=1}^{\infty}\mathcal{ALERT}_j \tag{11}$$

### 3.3.9. APT life cycle graph ($\mathcal{APTLCG}$)

$\mathcal{APTLG}$ is a superset of all the APT attacks sequences. It is a directed graph where nodes represent alerts and the edges between nodes represent the sequence of alerts i.e., ($\mathcal{ALERT}_i$, Sequence of $\mathcal{ALERT}_i$).

$$\mathcal{APTLCG} = \{\mathcal{ALERT}_i, \mathcal{ALERT}_j, \mathcal{ALERT}_k, ....\} \tag{12}$$

### 3.3.10. APT attack sequence ($\mathcal{APTAS}$)

$\mathcal{APTAS}$ It contains all sequences of APT attack alerts.

$$\begin{aligned}\mathcal{APTLAS} =& \{\mathcal{APTLAS}_1 \cup \mathcal{APTLAS}_2 \cup .....\mathcal{APTLAS}_n\} \\ =& \cup_{i=1}^{n}\mathcal{APTLAS}_i\end{aligned} \tag{13}$$

### 3.3.11. APT incomplete attack sequence ($\mathcal{APTIAS}_i$)

It contains an incomplete sequence of APT attacks on host i.

$$\begin{aligned}\mathcal{APTIAS}_i \subset& \{(alert.stage == S1) \cup (alert.stage == S2) \\ & \cup(alert.stage == S3) \cup (alert.stage == S4)\}\end{aligned} \tag{14}$$

### 3.3.12. APT incomplete attack sequence $\mathcal{APTCAS}_i$

It contains a complete sequence of APT attacks on host i.

$$\begin{aligned}\mathcal{APTCAS}_i =& \{(alert.stage == S1) \cap (alert.stage == S2) \\ & \cap(alert.stage == S3) \cap (alert.stage == S4)\}\end{aligned} \tag{15}$$

### 3.3.13. Event detection function

The *Event_Detection ()* takes logs of events as input and detects malicious events based on the type of event stored in the log file as shown in **Eq. (16)**. The different types of events are discussed in **Table 8**.

$$Event\_Detection(\mathcal{LOG}) = \begin{cases} \mathcal{EVENT}.type == \mathcal{MD}, & \text{if the domain is blacklisted} \\ \mathcal{EVENT}.type = \mathcal{MIP}, & \text{if the IP address is blacklisted} \\ \mathcal{EVENT}.type = \mathcal{MDNS}, & \text{if the IP associated with a domain is changing rapidly} \\ \mathcal{EVENT}.type = \mathcal{MSSL}, & \text{if the attached SSL certificate is malicious} \\ \mathcal{EVENT}.type = \mathcal{ME}, & \text{if the received file is malicious} \\ \mathcal{EVENT}.type = \mathcal{UFT}, & \text{if the files transfer are unusual} \\ \mathcal{EVENT}.type = \mathcal{MH}, & \text{if the hash value of the file is tampered} \\ \mathcal{EVENT}.type = \mathcal{SL}, & \text{if there are many failed attempts before successful login} \\ \mathcal{EVENT}.type = \mathcal{SDA}, & \text{if fail or success attempts to sensitive data detected} \\ \mathcal{EVENT}.type = \mathcal{CUP}, & \text{if the privileges of a user account has being changed} \\ \mathcal{EVENT}.type = \mathcal{NS}, & \text{if sequential port numbers being accessed from a specific host} \\ \mathcal{EVENT}.type = \mathcal{UNT}, & \text{if huge traffic toward a particular host is detected} \\ \mathcal{EVENT}.type = \mathcal{TOR}, & \text{if there is a tor connection to a host} \\ \mathcal{EVENT}.type = \mathcal{USB}, & \text{if there is unexpected system error} \end{cases} \tag{16}$$

**Table 8**
APT Attack Detectable Stages and Alerts.

| APT Stages | Unusual behavior Events | Alerts | Alert Description |
|---|---|---|---|
| S1: Initial Compromise | Malicious Domain | $alert_{MD}$ | Malicious Domain Alert |
| | Malicious File Execution | $alert_{ME}$ | Malicious File Execution alert |
| | Malicious Hashes | $alert_{MH}$ | Malicious Hash Alert |
| | Malicious DNS | $alert_{MDNS}$ | Alert generated in response of a malicious DNS detection |
| | Successful login after several failed attempts | $alert_{SL}$ | Alert generated in response of illegal login attempts |
| S2: C and C Communication | Malicious IP addresses | $alert_{IP}$ | Alert generated when malicious IP is detected |
| | Malicious SSL certificate | $alert_{ssl}$ | Alert generated when malicious SSL certificate is detected |
| S3: Asset Discovery | Network scanning | $alert_{NS}$ | Network ports scanning |
| | Assessing sensitive data | $alert_{SDA}$ | Illegal access to sensitive data |
| | Change to user privileges | $alert_{CUP}$ | Alert generated when there is a change in user privileges |
| S4: Data Ex-filtration | Unusual Network Traffic | $alert_{UNT}$ | Alert generated due to huge increase in network traffic |
| | Unexpected System Errors | $alert_{USB}$ | Alert generated due to system errors |
| | Tor connection with C and C server | $alert_{tor}$ | Alert generated when tor connection is detected |
| | File transfer, creation, or deletion | $alert_{UFB}$ | Alert generated due to file transfer, deletion, or creation |

### 3.3.14. Implementation of malicious behavior detection algorithm

The proposed architecture is designed to detect malicious behavior in a system using input logs as shown in **Algorithm 1**. The **Algorithm 1** takes a set of logs as input which consists of one or more events. As an output, the **Algorithm 1** generates an alert or error message. An Event_Detection() function is used to extract the type of each event from the input logs. The **Algorithm 1** examines the type of each event and divides it based on specific types of malicious behavior. The **Algorithm 1** generates an alert if the event type matches the malicious behavior. The details of the malicious behaviors and alerts are given in **Table 8**.

**Algorithm 1** Malicious Behavior Detection.

Input: $\mathcal{LOG} \supseteq \cup_{j=1}^{\infty} \mathcal{EVENT}_i$
Output: Alert or Error
Event_Detection(): $\mathcal{LOG}$–> $\mathcal{EVENT}_i.type$
**if** $\{\mathcal{EVENT}_i.type = (\mathcal{UNT}$ or $\mathcal{TOR})\}$ **then**
  $\mathcal{UNTS} = \mathcal{UNTS} \cup \mathcal{EVENT}_i$
  Alert():$\mathcal{EVENT}_i$–> $\{alert_{UNT}$ or $alert_{TOR}\}$
  $\mathcal{AS} = \{\mathcal{AS} \cup \{alert_{UNT}$ or $alert_{TOR}\}$
  $\mathcal{APTLCG} = \mathcal{APTLCG} \cup \{alert_{UNT}$ or $alert_{TOR}\}$
**else if** $\{\mathcal{EVENT}_i.type = (\mathcal{MD}$ or $\mathcal{MDNS}$ or $\mathcal{MSSL})\}$ **then**
  $\mathcal{MDS} = \mathcal{MDS} \cup \mathcal{EVENT}_i$
  Alert():$\mathcal{EVENT}_i$–> $\{alert_{MD}$ or $alert_{MDNS}$ or $alert_{MSSL}\}$
  $\mathcal{AS} = \mathcal{AS} \cup \{alert_{MD}$ or $alert_{MDNS}$ or $alert_{MSSL}\}$
  $\mathcal{APTLCG} = \mathcal{APTLCG} \cup \{alert_{MD}$ or $alert_{MDNS}$ or $alert_{MSSL}\}$
**else if** $\{\mathcal{EVENT}_i.type = (\mathcal{ME}$ or $\mathcal{UFT}$ or $\mathcal{MH})\}$ **then**
  $\mathcal{UFBS} = \mathcal{UFBS} \cup \mathcal{EVENT}_i$
  Alert():$\mathcal{EVENT}_i$–> $\{alert_{ME}$ or $alert_{UFT}$ or $alert_{MH}\}$
  $\mathcal{AS} = \mathcal{AS} \cup \{alert_{ME}$ or $alert_{UFT}$ or $alert_{MH}\}$
  $\mathcal{APTLCG} = \mathcal{APTLCG} \cup \{alert_{ME}$ or $alert_{UFT}$ or $alert_{MH}\}$
**else if** $\{\mathcal{EVENT}_i.type = \mathcal{MIP}\}$ **then**
  $\mathcal{MIPS} = \mathcal{MIPS} \cup \mathcal{EVENT}_i$
  Alert():$\mathcal{EVENT}_i$–> $alert_{MIP}$
  $\mathcal{AS} = \mathcal{AS} \cup alert_{MIP}$
  $\mathcal{APTLCG} = \mathcal{APTLCG} \cup \{alert_{MIP}\}$
**else if** $\{\mathcal{EVENT}_i.type = (\mathcal{SL}$ or $\mathcal{SDA}$ or $\mathcal{CUP}$ or $\mathcal{NS})\}$ **then**
  $\mathcal{USBS} = \mathcal{UFBS} \cup \mathcal{EVENT}_i$
  Alert():$\mathcal{EVENT}_i$–> $\{alert_{SL}$ or $alert_{SDA}$ or $alert_{CUP}$ or $alert_{NS}\}$
  $\mathcal{AS} = \mathcal{AS} \cup \{alert_{SL}$ or $alert_{SDA}$ or $alert_{CUP}$ or $alert_{NS}\}$
  $\mathcal{APTLCG} = \mathcal{APTLCG} \cup \{alert_{SL}$ or $alert_{SDA}$ or $alert_{CUP}$ or $alert_{NS}\}$
**else if** $\{\mathcal{EVENT}_i.type = \mathcal{USB}\}$ **then**
  $\mathcal{USBS} = \mathcal{USBS} \cup \mathcal{EVENT}_i$
  Alert():$\mathcal{EVENT}_i$–> $alert_{USB}$
  $\mathcal{AS} = \mathcal{AS} \cup alert_{USB}$
  $\mathcal{APTLCG} = \mathcal{APTLCG} \cup \{alert_{USB}\}$
**else**
  Error (No known malicious behavior found)
**end if**

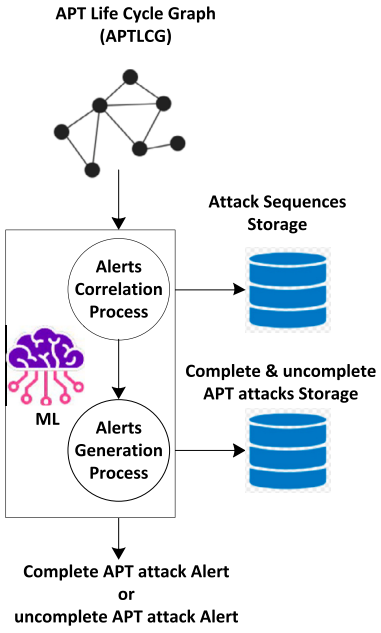


**Fig. 4.** Alerts Correlation and Generation.

Then, it adds the generated alert to the $\mathcal{APTLCG}$ Graph. However, if the event type does not match any of the malicious behaviors then an error message is generated which shows that malicious behavior is not found.

## 3.4. Alert correlation module

The alert correlation module is proposed to correlate the alerts generated by different modules and predict the exact attack type. This module accepts the APT Life Cycle Graph as input. Then this module uses supervised ML techniques like Random Forest and XGBoost, to correlate and predict the exact attack type alert. It reduces the false positive alerts and filters the repeated alerts generated by the proposed six modules. The main functionalities of the alert correlation module are depicted in **Fig. 4**.

The IoCs for different kinds of attacks can often overlap because adversaries frequently use identical tactics and techniques. For example, several attacks may use IoCs like malicious URLs, IP addresses, file signatures, and malicious behavior patterns. To effectively identify and respond to attacks, the proposed framework using AI techniques detects the exact attack IoCs associated with each attack. The APT attack detectable stages and alerts are given in **Table 8**.

### 3.4.1. Alert function

The *Alert ()* takes events as input and generates the corresponding alert based on the event type as shown in **Eq. (17)**. The *Alert ()* compares the *event.type* with pre-defined malicious behavior. If a match is found, then it generates the corresponding alert. The different types of alerts are discussed in **Table 8**.

$$Alert(\mathcal{EVENT}_i) = \begin{cases} alert_{MD}, & \text{if } \mathcal{EVENT}.type == \mathcal{MD} \\ alert_{MIP}, & \text{if } \mathcal{EVENT}.type == \mathcal{MIP} \\ alert_{UNT}, & \text{if } \mathcal{EVENT}.type == \mathcal{UNT} \\ alert_{MDNS}, & \text{if } \mathcal{EVENT}.type == \mathcal{MDNS} \\ alert_{MSSL}, & \text{if } \mathcal{EVENT}.type == \mathcal{MSSL} \\ alert_{ME}, & \text{if } \mathcal{EVENT}.type == \mathcal{ME} \\ alert_{UFT}, & \text{if } \mathcal{EVENT}.type == \mathcal{UFT} \\ alert_{MH}, & \text{if } \mathcal{EVENT}.type == \mathcal{MH} \\ alert_{SL}, & \text{if } \mathcal{EVENT}.type == \mathcal{SL} \\ alert_{SDA}, & \text{if } \mathcal{EVENT}.type == \mathcal{SDA} \\ alert_{CUP}, & \text{if } \mathcal{EVENT}.type == \mathcal{CUP} \\ alert_{NS}, & \text{if } \mathcal{EVENT}.type == \mathcal{NS} \\ alert_{TOR}, & \text{if } \mathcal{EVENT}.type == \mathcal{TOR} \\ alert_{USB}, & \text{if } \mathcal{EVENT}.type == \mathcal{USB} \end{cases} \quad (17)$$

### 3.4.2. Alert correlation function

The *Alert_correlation()* takes alerts store $\mathcal{APTAS}$ as input and add it to $\mathcal{APTIAS}_i$ or $\mathcal{APTCAS}_i$ set. In other words, it correlates the alerts and classifies them into complete APT Attacks and incomplete APT Attacks. The alert correlation function is shown in **Eq. (18)**.

$$Correlation(\mathcal{APTAS}) = \begin{cases} ||\mathcal{APTIAS}_i = \mathcal{APTIAS}_i \cup alert_i, \\ if\{(\mathcal{APTIAS}_i \subset (\emptyset \cup S1 \cup S2 \cup S3 \cup S4))\} \\ ||\mathcal{APTCAS}_i = \mathcal{APTCAS}_i \cup alert_i, \\ if\{(\mathcal{APTCAS}_i == (S1 \cap S2 \cap S3 \cap S4))\} \end{cases} \quad (18)$$

### 3.4.3. Implementation of alerts correlation algorithm

The proposed architecture correlates alerts and classifies them into complete APT Attacks and incomplete APT Attacks as shown in **Algorithm 2**. The **Algorithm 2** takes $\mathcal{APTLC}$ as input and adds it to either the complete APT alert set or the incomplete APT alert set i.e., $\mathcal{APTCAS}_i$, $\mathcal{APTIAS}_i$. In step 3, the **Algorithm 2** checks if the APT attack sequence i.e., $\mathcal{APTAS}$, for host i is empty. In step 4, if empty, it adds the alert from $\mathcal{APTLC}$ to $\mathcal{APTAS}$ of host i. In step 6, if $\mathcal{APTAS}$ is not empty, the **Algorithm 2** calls a loop that iterates through the remaining alerts in $\mathcal{APTLC}$. In step 7, the **Algorithm 2** executes a correlation check between the current alert ($\mathcal{ALERT}_i$) and all alerts already in $\mathcal{APTAS}$. The correlation check examines three parameters i.e., stage, source IP address, and timestamp. Firstly, it inspects whether the current alert's stage is greater than or equal to the stage of any existing alert in $\mathcal{APTAS}$. It shows a potential development in the attack. Secondly, it inspects whether the current alert's source IP address matches any existing alert in $\mathcal{APTAS}$. It indicates a potential connection between the events. Lastly, it inspects whether the current alert's timestamp is greater than or equal to any existing alert in $\mathcal{APTAS}$. It aids in maintaining chronological order within the APT sequence. In step 8, if the correlation check passes the current alert is added to $\mathcal{APTAS}$. Moreover, it invokes the $Alert_Correlation()$, which returns either an incomplete APT attack sequence i.e., $\mathcal{APTIAS}_i$ or a complete APT attack sequence i.e., $\mathcal{APTCAS}_i$. In step 9, if the correlation check fails, the **Algorithm 2** returns "similar alert already exists" and starts the execution of a new alert in the loop.

**Algorithm 2** Alerts correlation Algorithm.

1: Input: $\mathcal{APTLC}$
2: Output: *complete_APT_alert* or *uncomplete_APT_alert*
3: **if** {$\mathcal{APTAS} == \emptyset$} **then**
4: $\mathcal{APTAS} = \mathcal{APTAS} \cup \mathcal{ALERT}_i$
5: **else**
6: **while** {Length($\mathcal{APTLC}$)} **do**
7: **if** {{$\mathcal{ALERT}_i$.stage $\geq$ $\mathcal{ALERT}_j$.stage} $\wedge$ {$\mathcal{ALERT}_i$.S_IP == $\mathcal{ALERT}_j$.S_IP} $\wedge$ {$\mathcal{ALERT}_i$.timestamp $\geq$ $\mathcal{ALERT}_j$.timestamp}} **then**
8: $\mathcal{APTAS} = \mathcal{APTAS} \cup \mathcal{ALERT}_i$
9: $Correlation() : \mathcal{APTAS} -> \{\mathcal{APTIAS}_i$ or $\mathcal{APTCAS}_i\}$
10: **else**
11: Similar alert already added
12: **end if**
13: **end while**
14: **end if**

### 3.4.4. APT reporting

APT offers substantial issues due to its stealth and long-lasting nature. Early diagnosis and rapid intervention are critical for minimizing damage. This module reports two types of APT alerts i.e., complete APT attack alert and incomplete APT attack alert. This reporting allows the incident and response team to make swift and appropriate decisions based on the information provided by the proposed system.

## 4. Experimental setup and results evaluation

The experimental setup and obtained results are discussed in the following sections.

### 4.1. System setup

A VMWare ESXi physical server is used to install the virtual machines with different operating systems and services. The vulnerable web applications i.e., Damn Vulnerable Web Application (DVWA), X-operating system, Apache, Mysql, Php, Perl (XAMPP) server are installed on the first VM. Similarly, the Domain Name System (DNS) Server and file server are installed on the other two VMs. The adversary attacks the servers from the Public Network. Each server has a log-capturing capacity. The log server is used for filtering and storing logs. Moreover, the networks and server logs are periodically transferred to the Splunk server as shown in **Fig. 5**.

### 4.2. Dataset collection

The dataset, in [12], is extended by adding more alert types as shown in **Table 8**. This dataset contains the source IP address, destination IP

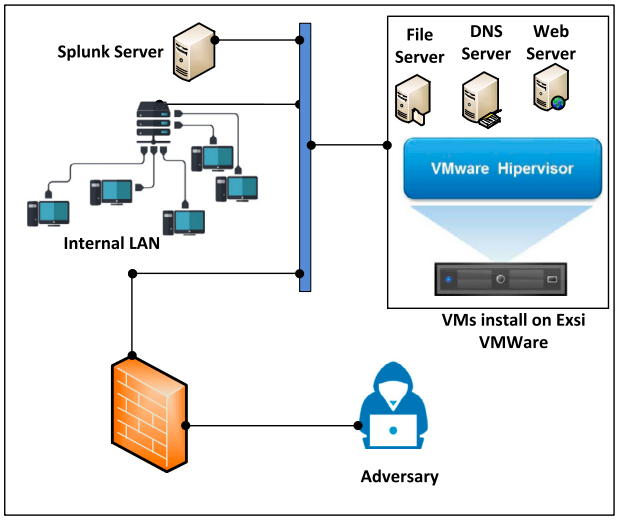


**Fig. 5.** System Setup.

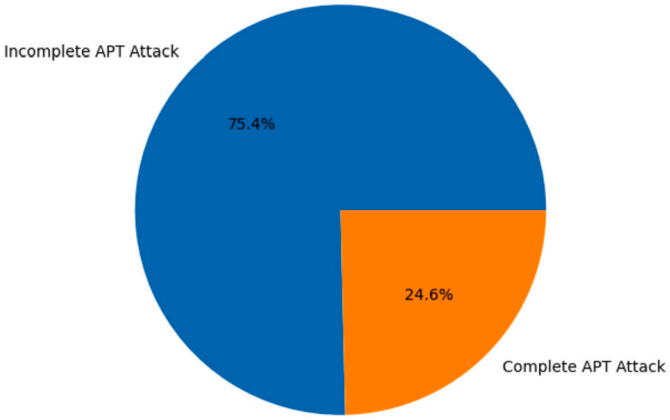


**Fig. 6.** Class Distribution.

address, source port number, destination port number, timestamps, alert types, infected hosts, and attack steps. Moreover, it is used as the basis for ML models to detect and investigate complete and incomplete APT attacks.

### 4.3. Data pre-processing and encoding

Since the original dataset is messy and it is not suitable for machine learning algorithms, we have performed preprocessing. Processing plays a vital role in improving data quality. Initially, our dataset had 3886 records with 9 attributes as discussed in section B. After careful analysis of the attributes, the alter_id attribute is discarded because it carries no information for the ML algorithm. Furthermore, the source IP attribute is deleted as it duplicates the IPs of the infected host attribute.

In the next step, the dataset is annotated into two classes i.e., 'complete APT attack' and 'incomplete APT attack'. Therefore, 1031 incomplete APT attacks and 337 complete APT attacks are identified as shown in **Fig. 6**.

In the next step, data encoding is performed. Initially, infected host IPs and destination IPs are converted into their respective IP classes i.e., A, B, and C. In the dataset, alert type, step, destination IP and infected host IP are categorical attributes. Therefore, a one-hot encoding technique is used to transform these attributes. There are 13 distinct values for alert type, therefore, after one hot encoding there are 13 columns each representing a different alert type. Similarly, there are 3 columns for each class of IP address. Since we have converted our dataset into binary classification, while in the original dataset, while in the original dataset, each record represents a single APT step. For the 'complete APT attack', the data of four steps is aggregated, while less than four steps correspond to an 'incomplete APT attack'. The aggregation criteria for the alert type, destination IP address, and infected host IP address attributes are set to sum. Moreover, the time stamp in the data is a continuous attribute. Therefore, 'mean' as the aggregation criteria used for the time stamp. The destination port and source port are nominal attributes, therefore 'median' is used as the aggregation criteria. Following data pre-processing and encoding, the resulting dataset contains 1368 records (rows) and 24 attributes (columns).

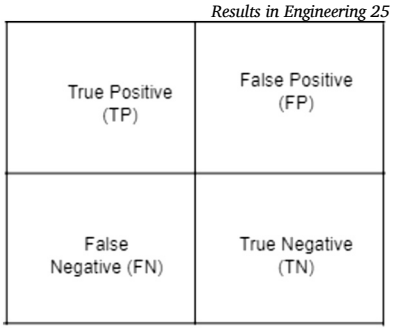


**Fig. 7.** Confusion Matrix.

The Random Forest and XGBoost classifiers are selected because the majority of the attributes in the dataset are categorical. It is worth mentioning that a decision tree-based classifier performs well on categorical attributes. Both, XGBoost and Random Forest are tree-based ensemble classifiers. Random Forest is a powerful ML homogeneous ensemble classifier that is composed of multiple decision trees. Each tree is built using random sub-samples of the training data. The final decision is made using majority vote criteria. Random Forest is useful because it reduces overfitting, and increases the robustness of the model. An overfitted model poorly performs on unseen data or cannot generalize, unlike random forest, XGBoost implements gradient boosting where each new tree is trained to work on the errors produced by preceding trees. Moreover, XGBoost uses a regularization mechanism in tree formation, that prevents the model from overfitting.

### 4.4. Results and discussion

The training dataset is split into training and testing datasets with a 20:80 ratio. The out-of-sample experiments also called hold-out testing or external validation are performed on the dataset. The testing data is unseen for the model to provide a realistic assessment of the model. Since, the dataset is imbalanced, therefore, only accuracy is not enough to validate the model. So, we added other performance metrics as well such as True Positive Rate (TPR), True Negative Rate (TNR), False Negative Rate (FNR), False Positive Rate (FPR), F1 score, and Precision (P). All these metrics are calculated from the confusion matrix shown in **Fig. 7**.

False Positive in a real-world environment generates a false alarm and interrupts daily operations, while False Negative is a real threat to the system. We aim to reduce both. Accuracy is the total correctly predicted examples from all classes (both positive and negative classes in our case) out of the total examples in the dataset. Accuracy is calculated by using the formula given in **Eq. (19)**.

$$Acc = \frac{TP + TN}{TP + FP + TN + FN} \tag{19}$$

TPR is also called recall which measures how many positive examples are correctly predicted by the model out of total positive examples. The formula is given in **Eq. (20)**. TNR is calculated in a similar way for the negative class.

$$TPR = \frac{TP}{TP + FN} \tag{20}$$

**Table 9**
A comparison between Random Forest and XGBoost Classifiers.

| Classifiers | Acc | TPR | TNR | FNR | FPR | F1 | P |
|---|---|---|---|---|---|---|---|
| Random Forest | 98.5 | 98.6 | 98.5 | 1.4 | 1.4 | 95.8 | 97.1 |
| XGBoost | 99.6 | 98.6 | 1.0 | 1.4 | 0.0 | 99.2 | 100 |

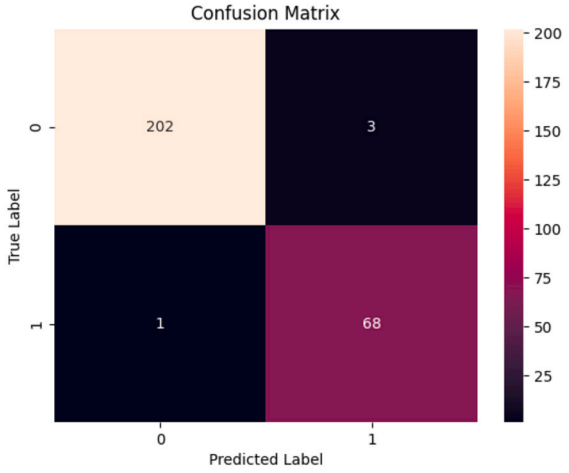


**Fig. 8.** Confusion Matrix.

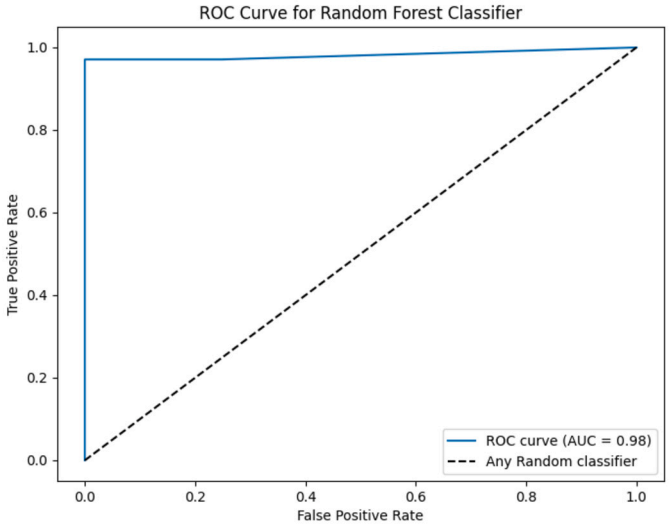


**Fig. 9.** ROC Curve.

Precision is the total correctly classified examples out of the total predicted positive examples. The formula is given in **Eq. (21)**.

$$Precision = \frac{TP}{TP + FP} \tag{21}$$

Precision tries to reduce recall and vice versa, but the goal is to improve both. F1 score is the harmonic mean of both recall and precision, which can be computed by the formula given in **Eq. (22)**.

$$F1 - Score = \frac{2 * Precision * TPR}{Precision + TPR} \tag{22}$$

The Random Forest and XGBoost classifier's prediction accuracy, TPR, TNR, FNR, FPR, F1 score, and Precision are given in **Table 9**. The confusion matrix and Receiver Operating Characteristic (ROC) curve for the Random Forest classifier are shown in **Fig. 8** and **Fig. 9** respectively.

Similarly, the confusion matrix and ROC curve for the XGBoost classifier are shown in **Fig. 10** and **Fig. 11** respectively.

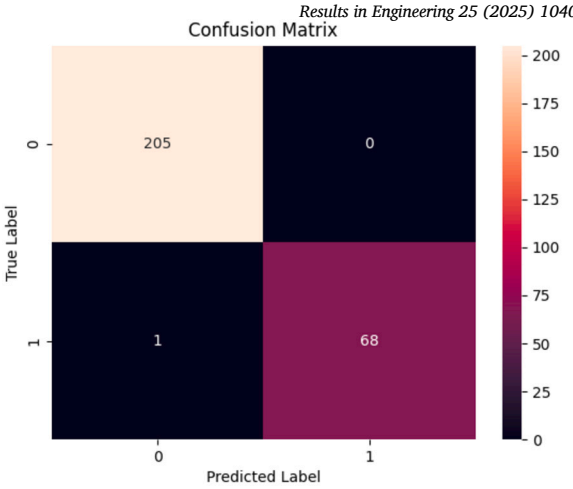


**Fig. 10.** Confusion Matrix.

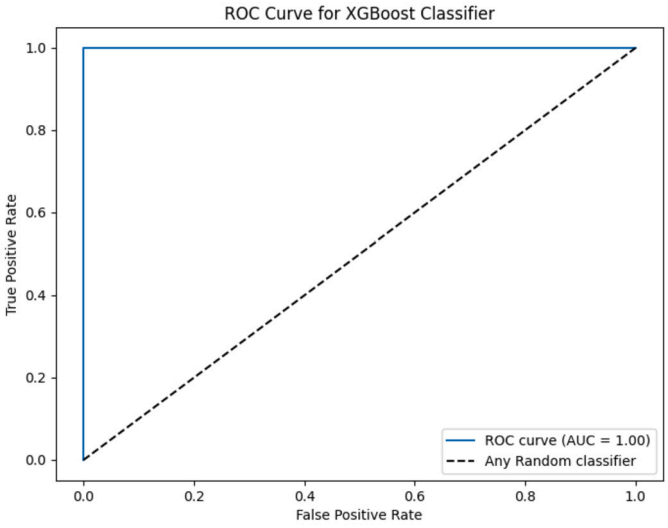


**Fig. 11.** ROC Curve.

The importance of each feature/attribute is computed in the prediction task as shown in **Fig. 12**. All the features are used in the prediction step, however, less number of features can give nearly the same results. Thus, making the predictor suitable for embedded systems or the Internet of Things.

The experimental results suggest that if the data is properly preprocessed and encoded, ML techniques can greatly enhance the performance of any SEIM tool.

### *4.5. Comparisons among proposed and state-of-the-art framework*

Both the existing and proposed models use machine learning techniques to detect the different stages of APTs. However, the proposed model outperformed existing models by detecting four APT stages with higher accuracy. The comparison among existing and proposed models is shown in **Table 10**. The results show that the proposed model has outperformed all SOTA techniques in terms of accuracy, TPR, TNR, FPR, and F1 score. Furthermore, the proposed model exhibits broader detection capability with a lower FPR.

## 5. Conclusion

This research paper covers the issues that the CSIRT faces while dealing with enormous alerts and time-consuming manual operations in the

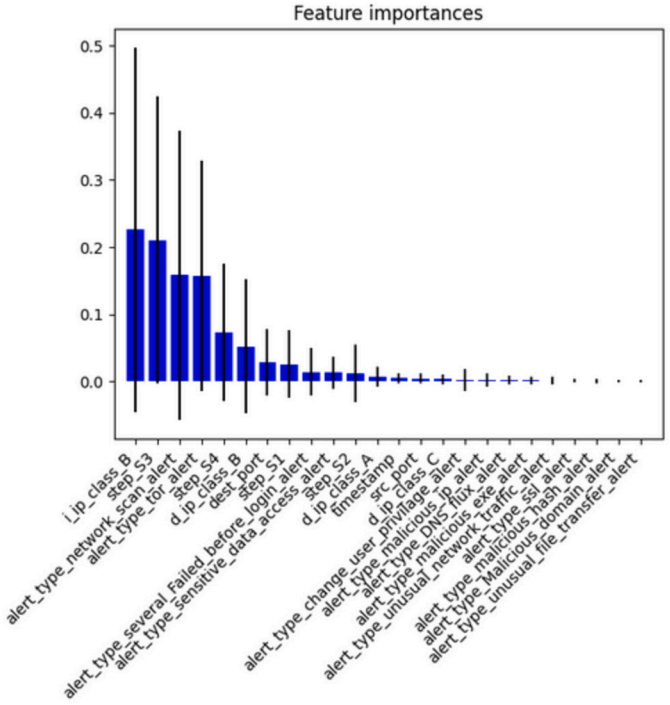


**Fig. 12.** Features Importance.

**Table 10**
A comparison among Existing and Proposed Models.

| Model | APT steps | Acc (%) | TPR (%) | TNR (%) | FNR (%) | FPR (%) | F1 | P |
|---|---|---|---|---|---|---|---|---|
| [12] | 4 | 84.8 | 81.8 | – | – | 4.5 | – | – |
| [13] | 4 | – | 100 | – | – | high | – | – |
| [14] | 4 | – | – | – | – | 27.8 | – | – |
| [15] | 1 | – | 97.2 | – | – | 14.2 | | |
| [16] | 1 | 97.1 | 99.0 | 94.7 | 0.96 | 5.3 | 97.4 | 95.9 |
| Proposed Model | 4 | 99.6 | 98.5 | 100 | 1.4 | 0.0 | 99.2 | 100 |

SOC. This study proposes a novel architecture to improve malicious behavior detection and reduce false positive alarms. We implemented six different malicious behavior detection modules. These modules can detect 14 malicious behaviors. These malicious behaviors are reflected in the APT dataset by adding the new alert types. Then, ML approaches, such as Random Forest and XGBoost, are trained and tested on the extended APT dataset. The proposed Random Forest-based model detects malicious behaviors with an accuracy of 98.54% and FPR of 1.46%. However, the XGBoost-based model generates a detection accuracy of 99.6% and an FPR of 0.0%.

### *5.1. Future work*

In this section, we present the potential future directions: 1) Integration of transfer learning model to enhance the detection capabilities of the proposed framework; 2) Identifying and adding emerging indicators to the IoC database; 3) Detecting APT at a more fine-grained level by incorporating more APT steps in real-world scenarios; 4) Identifying more malicious behaviors by adding more detection modules.

### *5.2. Limitations of the study*

In this section, we discuss the limitations of the proposed framework: 1) Requires continuous update of IoCs. 2) Reliance on Splunk tools.

## CRediT authorship contribution statement

**Gauhar Ali:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Formal analysis, Data curation, Conceptualization. **Sajid Shah:** Writing – review & editing, Visualization, Validation, Software, Data curation. **Mohammed ElAffendi:** Supervision.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgement

This paper is derived from a research grant funded by Prince Sultan University - Kingdom of Saudi Arabia - with grant number (SEED-CCIS-2024-168). Also, the authors would like to acknowledge the support of Prince Sultan University for paying the Article Processing Charges (APC) of this publication.

## Data availability

Data will be made available on request.

## References

[1] Cisco, 2017 annual cybersecurity report, https://learningnetwork.cisco.com/s/article/cisco-2017-annual-cybersecurity-report-pdf. (Accessed 10 March 2024).
[2] D. Schlette, M. Caselli, G. Pernul, A comparative study on cyber threat intelligence: the security incident response perspective, IEEE Commun. Surv. Tutor. 23 (4) (2021) 2525–2556.
[3] M. Bromiley, Sans 2019 incident response (ir) survey: It's time for a change, Technical Report, SANS Institute, 2019.
[4] P. Cichonski, T. Millar, T. Grance, K. Scarfone, et al., Computer security incident handling guide, NASA Spec. Publ. 800 (61) (2012) 1–147.
[5] N. Tendikov, L. Rzayeva, B. Saoud, I. Shayea, M.H. Azmi, A. Myrzatay, M. Alnakhli, Security information event management data acquisition and analysis methods with machine learning principles, Results Eng. 22 (2024) 102254.
[6] P. Sai Charan, T. Gireesh Kumar, P. Mohan Anand, Advance persistent threat detection using long short term memory (lstm) neural networks, in: Emerging Technologies in Computer Engineering: Microservices in Big Data Analytics: Second International Conference, ICETCE 2019, Jaipur, India, February 1–2, 2019, in: Revised Selected Papers, vol. 2, Springer, 2019, pp. 45–54.
[7] S. Abdelkader, J. Amissah, S. Kinga, G. Mugerwa, E. Emmanuel, D.-E.A. Mansour, M. Bajaj, V. Blazek, L. Prokop, Securing modern power systems: implementing comprehensive strategies to enhance resilience and reliability against cyber-attacks, Results Eng. (2024) 102647.
[8] C. Islam, M.A. Babar, R. Croft, H. Janicke, Smartvalidator: a framework for automatic identification and classification of cyber threat data, J. Netw. Comput. Appl. 202 (2022) 103370.
[9] Z. Jia, X. Wang, Y. Xiong, Y. Zhang, J. Zhao, Aisle: self-supervised representation learning for the investigation of advanced persistent threat, in: 2022 7th IEEE International Conference on Data Science in Cyberspace (DSC), IEEE, 2022, pp. 360–366.
[10] A. Masarweh, J. Al-Saraireh, Threat led advanced persistent threat penetration test, Int. J. Secur. Netw. 17 (3) (2022) 203–219.
[11] A. Alsanad, S. Altuwaijri, Advanced persistent threat attack detection using clustering algorithms, Int. J. Adv. Comput. Sci. Appl. 13 (9) (2022).
[12] I. Ghafir, M. Hammoudeh, V. Prenosil, L. Han, R. Hegarty, K. Rabie, F.J. Aparicio-Navarro, Detection of advanced persistent threat using machine-learning correlation analysis, Future Gener. Comput. Syst. 89 (2018) 349–359.
[13] G. Brogi, V.V.T. Tong, Terminaptor: highlighting advanced persistent threats through information flow tracking, in: 2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS), IEEE, 2016, pp. 1–5.
[14] P. Giura, W. Wang, A context-based detection framework for advanced persistent threats, in: 2012 International Conference on Cyber Security, IEEE, 2012, pp. 69–74.
[15] J.V. Chandra, N. Challa, S.K. Pasupuleti, A practical approach to e-mail spam filters to protect data from advanced persistent threat, in: 2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT), IEEE, 2016, pp. 1–5.
[16] N. Saini, V. Bhat Kasaragod, K. Prakasha, A.K. Das, A hybrid ensemble machine learning model for detecting apt attacks based on network behavior anomaly detection, Concurr. Comput. Pract. Exp. 35 (28) (2023) e7865.
[17] C. Do Xuan, Detecting apt attacks based on network traffic using machine learning, J. Web Eng. 20 (1) (2021) 171–190.

[18] A. Williams, M. Nicolett, Improve it security with vulnerability management, 2005, Gartner ID (G00127481).
[19] D.R. Miller, S. Harris, A. Harper, S. VanDyke, C. Blask, Security Information and Event Management (SIEM) Implementation, McGraw Hill Professional, 2010.
[20] M. Sheeraz, M.A. Paracha, M.U. Haque, M.H. Durad, S.M. Mohsin, S.S. Band, A. Mosavi, Effective security monitoring using efficient siem architecture, Hum.-Cent. Comput. Inf. Sci. 13 (2023) 1–18.
[21] Gartner magic quadrant for security information and event management, https://www.gartner.com/en/documents/4019750?ref=null. (Accessed 25 February 2024).
[22] A. Hwoij, A.h. Khamaiseh, M. Ababneh, Siem architecture for the Internet of things and smart city, in: International Conference on Data Science, E-learning and Information Systems 2021, 2021, pp. 147–152.
[23] M. Hristov, M. Nenova, G. Iliev, D. Avresky, Integration of splunk enterprise siem for ddos attack detection in iot, in: 2021 IEEE 20th International Symposium on Network Computing and Applications (NCA), IEEE, 2021, pp. 1–5.
[24] N. Balaji, B.K. Pai, B. Bhat, B. Praveen, Data Visualization in Splunk and Tableau: a Case Study Demonstration, Journal of Physics: Conference Series, vol. 1767, IOP Publishing, 2021, p. 012008.
[25] D. Dunsin, M.C. Ghanem, K. Ouazzane, V. Vassilev, A comprehensive analysis of the role of artificial intelligence and machine learning in modern digital forensics and incident response, Forensic Sci. Int., Digit. Investig. 48 (2024) 301675.
[26] O. ElSahly, A. Abdelfatah, An incident detection model using random forest classifier, Smart Cities 6 (4) (2023) 1786–1813.
[27] P. Pothumani, E.S. Reddy, Original research article network intrusion detection using ensemble weighted voting classifier based honeypot framework, J. Auton. Intell. 7 (3) (2024).
[28] W.N.H. Ibrahim, S. Anuar, A. Selamat, O. Krejcar, R.G. Crespo, E. Herrera-Viedma, H. Fujita, Multilayer framework for botnet detection using machine learning algorithms, IEEE Access 9 (2021) 48753–48768.
[29] H.J. Hadi, Y. Cao, S. Li, N. Ahmad, M.A. Alshara, Fcg-mfd: Benchmark function call graph-based dataset for malware family detection, J. Netw. Comput. Appl. 233 (2025) 104050.
[30] T.T. Nguyen, V.J. Reddi, Deep reinforcement learning for cyber security, IEEE Trans. Neural Netw. Learn. Syst. 34 (8) (2021) 3779–3795.
[31] A.S. Almasoud, Enhanced metaheuristics with machine learning enabled cyberattack detection model, Intell. Autom. Soft Comput. 37 (3) (2023).
[32] H. Sadia, S. Farhan, Y.U. Haq, R. Sana, T. Mahmood, S.A.O. Bahaj, A. Rehman, Intrusion detection system for wireless sensor networks: a machine learning based approach, IEEE Access (2024).
[33] A. Sopan, M. Berninger, M. Mulakaluri, R. Katakam, Building a machine learning model for the soc, by the input from the soc, and analyzing it for the soc, in: 2018 IEEE Symposium on Visualization for Cyber Security (VizSec), IEEE, 2018, pp. 1–8.
[34] H.M. Farooq, N.M. Otaibi, Optimal machine learning algorithms for cyber threat detection, in: 2018 UKSim-AMSS 20th International Conference on Computer Modelling and Simulation (UKSim), IEEE, 2018, pp. 32–37.
[35] K. Sethi, E. Sai Rupesh, R. Kumar, P. Bera, Y. Venu Madhav, A context-aware robust intrusion detection system: a reinforcement learning-based approach, Int. J. Inf. Secur. 19 (2020) 657–678.
[36] C. Nilă, I. Apostol, V. Patriciu, Machine learning approach to quick incident response, in: 2020 13th International Conference on Communications (COMM), IEEE, 2020, pp. 291–296.