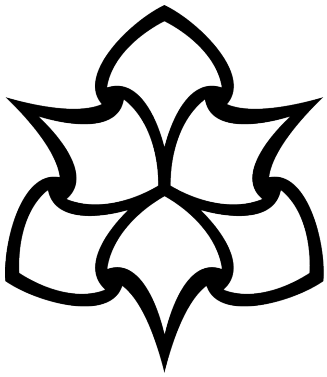

Detection of Advanced Persistent Threat Using Machine-Learning Correlation Analysis

Ibrahim Ghafir[a,b], Mohammad Hammoudeh[c], Vaclav Prenosil[b], Liangxiu Han[c], Robert Hegarty[c], Khaled Rabie[c], Francisco J. Aparicio-Navarro[d]

[a] *Department of Computer Science, Durham University, Durham, UK*
[b] *Faculty of Informatics, Masaryk University, Brno, Czech Republic*
[c] *Faculty of Science and Engineering, Manchester Metropolitan University, Manchester, UK*
[d] *School of Engineering, Newcastle University, Newcastle upon Tyne, UK*

Abstract

As one of the most serious types of cyber attack, Advanced Persistent Threats (APT) have caused major concerns on a global scale. APT refers to a persistent, multi-stage attack with the intention to compromise the system and gain information from the targeted system, which has the potential to cause significant damage and substantial financial loss. The accurate detection and prediction of APT is an ongoing challenge. This work proposes a novel machine learning-based system entitled MLAPT, which can accurately and rapidly detect and predict APT attacks in a systematic way. The MLAPT runs through three main phases: (1) Threat detection, in which eight methods have been developed to detect different techniques used during the various APT steps. The implementation and validation of these methods with real traffic is a significant contribution to the current body of research; (2) Alert correlation, in which a correlation framework is designed to link the outputs of the detection methods, aims to identify alerts that could be related and belong to a single APT scenario; and (3) Attack prediction, in which a machine learning-based prediction module is proposed based on the correlation framework output, to be used by the network security team to determine the probability of the early alerts to develop a complete APT attack. MLAPT is experimentally evaluated and the presented system is able to predict APT in its early steps with a prediction accuracy of 84.8%.

Keywords:
Cyber attacks, advanced persistent threat, malware, intrusion detection system, alert correlation, machine learning.

1. Introduction

The volume, complexity and variety of Cyber attacks are continually increasing. This trend is currently being driven by cyber warfare and the emergence of the Internet of Things [1–3]. The annual cost of cyber attacks was $3 trillion in 2015 and it is expected to increase to more than $6 trillion per annum by 2021 [4]. This high cost has brought much interest in research and investment towards developing new cyber attacks defence methods and techniques [5–8]. Although virus scanners, firewalls and intrusion detection and prevention systems (IDPSs) have been able to detect and prevent many of cyber attacks, cyber-criminals in turn have developed more advanced methods and techniques to intrude into the target's network and exploit their resources, targeting both wired and wireless communications [9, 10]. In addition, many of the defence approaches against cyber attacks consider those attacks are targeting random networks, so they assume that if the company's network is well protected, the attacker can surrender and move onto an easier target. Nonetheless, according to a technical report by Trend Micro [11], this assumption is no longer valid with the rise of targeted attacks, Advanced Persistent Threats (APTs), in which both cyber-criminals and hackers are targeting selected organizations and persisting until they achieve their goals.

The APT attack is a persistent, targeted attack on a specific organisation and is performed through several steps [12]. The main aim of APT is espionage and then data exfiltration. Therefore, APT is considered as a new and more complex version of multi-step attack. These APTs present a challenge for current detection methods as they use advanced techniques and make use of unknown vulnerabilities. Moreover, the economic damage due to a successful APT attack significant. The potential cost of attacks is the major motivation

for the investments in intrusion detection and prevention systems [13]. APTs are currently one of the most serious threats to companies and governments [14].

Most of the research in the area of APT detection, has focused on analysing already identified APTs [15–21], or detecting a particular APT that uses a specific piece of malware [22]. Some works have attempted to detect novel APT attacks. However, they face serious shortcomings in achieving real time detection [23], detecting all APT attack steps [23], balance between false positive and false negative rates [22], and correlating of events spanning over a long period of time [24, 25]. The existing work is encouraging. However, the accurate and timely detection of APT remains a challenge.

In this work, we have developed a novel machine learning-based system, called MLAPT, which can accurately, and quickly detect and predict APT attacks in a holistic way, making a significant contribution to the field of intrusion detection systems (IDS). MLAPT runs through three main phases: threat detection, alert correlation and attack prediction, the major contributions of this work include:

- Threat detection: the aim of this first phase is to detect threats during the multi-step APT attack. We have developed eight methods/modules to detect various attacks used in one of the APT attack steps. These include disguised exe file detection (DeFD), malicious file hash detection (MFHD), malicious domain name detection (MDND), malicious IP address detection (MIPD), malicious SSL certificate detection (MSSLD), domain flux detection (DFD), scan detection (SD), and Tor connection detection (TorCD). The output of this phase is alerts, also known as events, triggered by the individual modules. All the methods have been evaluated using real network traffic.

- Alert correlation: this second phase of the alert correlation intends to correlate the alerts produced in the first phase with one APT attack scenario. The main objective of using the correlation framework is to reduce the false positive rate of the MLAPT detection system. The process in

this phase undergoes three main steps: alerts filter (AF), to identify redundant or repeated alerts; clustering of alerts (AC), which most likely belong to the same APT attack scenario; and correlation indexing (CI), to evaluate the degree of correlation between alerts of each cluster.

- Attack prediction: in the final phase, a machine-learning-based prediction module (PM) is designed and implemented based on a historical record of the monitored network. This module can be used by the network security team to determine the probability of the early alerts to develop a complete APT attack.

- The proposed MLAPT system is able to process and analyse the network traffic in real time without needing to store data, and make possible the early prediction of APT attacks so that an appropriate and timely response can take place before the attack completes its life cycle.

The remainder of this paper is organized as follows. Section 2 presents the related work to APT detection. The proposed APT detection system and its architecture are described in Section 3. Section 4 explains the implementation of the proposed approach. The evaluation results and the performance comparison with the existing APT detection system are shown in Section 5 and Section 6 respectively. Section 7 concludes the paper.

2. Related Work

The APT detection has been a challenge for the current Intrusion Detection Systems (IDSs), and much research has been conducted to address this type of multi-stage attack. Table 1 describes current APT detection systems and mesmerises their limitations.

TerminAPTor, an APT detector, is described in [26]. This detector uses information flow tracking to find the links between the elementary attacks, which are triggered within the APT life cycle. TerminAPTor depends on an agent, which can be a standard intrusion detection system, to detect those elementary

Table 1: Current APT detection systems: description and limitations

APT Detector	Description	Limitations
TerminAPTor [26]	Uses information flow tracking to find the links between the elementary alerts	High false positives
C&C-based [27]	Considers the access to the C&C domains independent while the access to the legal domain is correlated	Can be easily evaded when the infected hosts connect to the c&C domains while users are surfing the Internet
Spear-phishing-based [28]	Uses "Tokens" and utilises mathematical and computational analysis to filter spam emails	The spear phishing email may not contain any of the Tokens - Detects only one step of APT life cycle
Statistical APT detector [29]	The generated events in each APT step are correlated in a statistical manner	Requires significant expert knowledge to set up and maintain
Active-learning-based [30]	Detects malicious PDFs based on white lists and their compatibility as viable PDF files	Detects only one step of APT life cycle
Data Leakage Prevention [31]	Utilises DLP algorithm to detect the step of data exfiltration	Detects only one step of APT life cycle - Cannot achieve real time detection
SPuNge [23]	Gathers the data on the hosts' side	Detects only one step of APT life cycle - Cannot achieve real time detection
Context-based [32]	Models APT as a pyramid in which the top of the pyramids represent the attack goal, and the lateral planes indicates the environments involved in the APT life cycle	Requires significant expert knowledge to set up and maintain

attacks. The authors evaluated TerminAPTor by simulating only two APT scenarios and demonstrated that the APT detector needs to be improved by filtering the false positives.

An APT detection system based on C&C domains detection is introduced in [27]. This work analyses the C&C communication and states a new feature that the access to C&C domains is independent while the access to legal domains is correlated. Despite the fact that the detection system achieved significant results when validated on a public dataset, the authors mentioned that the detection can be easily evaded when the infected hosts connect to the C&C domains while users are surfing the Internet. Moreover, missing the detection of C&C domains leads to failure in APT detection since this system depends on

detecting only one step of the APT life cycle.

An approach for APT detection based on spear phishing detection is explored in [28]. This approach depends on mathematical and computational analysis to filter spam emails. Tokens, which are considered as a group of words and characters such as (click here, free, Viagra, replica), should be defined for the detection algorithm to separate legitimate and spam emails. However, the spear phishing email might not include any of the tokens which are necessary for the algorithm process. Additionally, depending on one step for APT detection leads the system to fail when missing that step.

A statistical APT detector, similar to TerminAPTor detector, is developed in [29]. This system considers that APT undergoes five states which are delivery, exploit, installation, C&C and actions; and several activities are taken in each state. The generated events in each state are correlated in a statistical manner. This system requires significant expert knowledge to set up and maintain.

An active-learning-based framework for malicious PDFs detection is suggested in [30]. These malicious PDFs can be used in the early steps of APT to get the point of entry. The system collects all PDFs transferred over the network, then all known benign and malicious files are filtered by the "known files module" which depends on white lists, reputation systems and antivirus signature repository. Following this, the remaining files "unknown files" are checked for their compatibility as viable PDF files. This approach detects only one step of the APT life cycle.

An approach based on Data Leakage Prevention (DLP) is proposed in [31]. This approach focuses on detecting the last step of APT which is the data exfiltration. A DLP algorithm is used to process the data traffic to detect data leaks and generate "fingerprints" according to the features of the leak. The proposed system utilises external cyber counterintelligence (CCI) sensors in order to track the location or path of the leaked data. This approach is limited to detect only one step of APT which is the data exfiltration. In addition, it cannot achieve the real time detection as the CCI analysis unit should wait for the information from the sensors. Moreover, it is not guaranteed that the

CCI sensors can provide the required information regarding the leaked data fingerprints. This approach also introduces privacy issues, whereby actors in the CCI have access to the data stored and transfered by all users of the sytems.

A working prototype, SPuNge, is presented in [23]. The proposed approach depends on the gathered data on the hosts' side and aims to detect possible APT attacks. SPuNge undergoes two main phases, in the first one, the detected malicious URLs are analysed. Those URLs can be connected by the hosts' computers over HTTP(S) with an Internet browser or by malware installed on the infected machines. The computers which show a similar activity are then determined. This system depends on detecting one activity of the APT attack, which is malicious URL connection, and does not consider the other activities of APT. Meaning, if the detection system misses the malicious URL connection, the whole APT scenario will not be detected. Additionally, the system cannot achieve real time detection.

A context-based framework for APT detection is explained in [32]. This framework is based on modelling APT as an attack pyramid in which the top of the pyramid represents the attack goal, and the lateral planes indicates the environments involved in the APT life cycle. This detection framework requires significant expert knowledge to set up and maintain.

Finally, the existing APT detection systems face serious shortcomings in achieving real time detection, balance between false positive and false negative rates and correlating of events spanning over a long period of time. To address those weaknesses, this paper presents a new approach for APT detection and prediction.

3. A Correlation-based System for Real-time APT Detection and Prediction

3.1. Design Rationale

APTs are multi-step attacks, therefore effective detection should go through the detection of the techniques used within each stage of the APT life cycle.

Detection modules should be developed to detect the most common techniques used in the APT attack steps.

However, detecting a single stage of an APT technique itself does not mean detecting an APT attack. Even though an individual module alert indicates a technique which can possibly be used in an APT attack, this technique can be used for other types of attacks or it can be even a benign one. For example, domain flux, port scanning and malicious C&C communications, used in the APT attack, can be also used for botnet attacks [33]. Moreover, Tor network connection, used for data exfiltration in the APT attack, can also be used legally to protect the confidentiality of a user traffic [34]. Thus, individually these detection modules are ineffective and their information should be fused to build a complete picture regarding an APT attack. For this reason, a correlation framework should be developed to link the outputs of the detection modules and reduce the false positive rate of the detection system.

Predicting the APT attack in its early steps would minimise the damage and prevent the attacker from achieving the goal of data exfiltration. With a historical record of the correlation framework output, machine learning algorithms can be used to train a prediction model. As the purpose of the prediction model is to classify the early alerts of the correlation framework, classification algorithms should be selected to train the model.

3.2. MLAPT Architecture

Based on the design rationale, the architecture of the proposed system (MLAPT) is shown in Figure 1. The MLAPT runs through three main phases: threat detection, alert correlation and attack prediction.

Initially, the network traffic is scanned and processed to detect possible techniques used in the APT life cycle. To this end, eight detection modules have been developed; each module implements a method to detect one technique used in one of APT attack steps, and it is independent from the other modules. MLAPT implemented eight modules, presented later in Section 3.3 on page 10, to detect the most commonly used techniques in the APT life cycle. The output

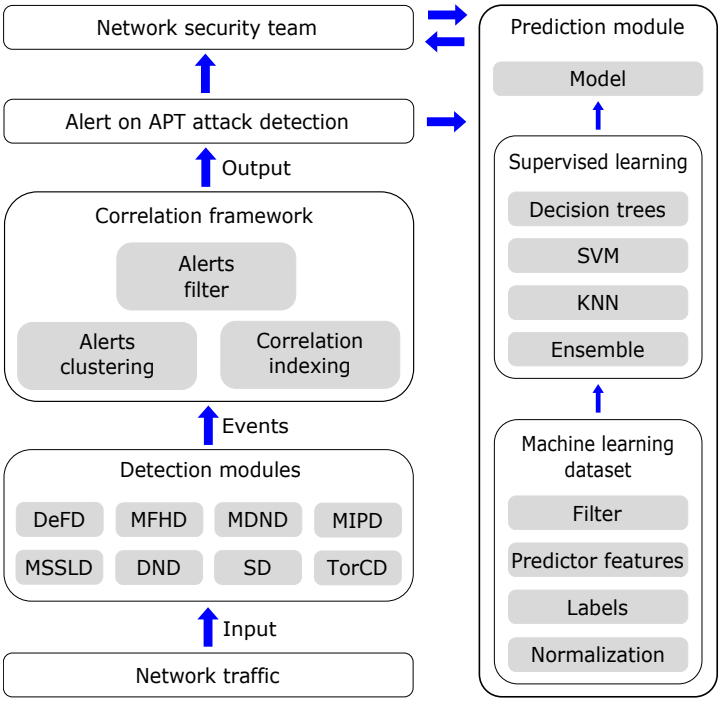

Figure 1: The Architecture of MLAPT.

of this phase are alerts, also known as events, triggered by individual modules.

The alerts raised by individual detection modules are then fed to the correlation framework. The aim of the correlation framework is to find alerts could be related and belong to one APT attack scenario. The process in this phase undergoes three main steps: alerts filter to identify redundant or repeated alerts; clustering of alerts which most likely belong to the same APT attack scenario; and correlation indexing to evaluate the degree of correlation between alerts of each cluster.

In the final phase, a machine-learning-based prediction module is used by the network security team to determine the probability of the early alerts to develop a complete APT attack. This allows the network security team to predict the APT attack in its early steps and apply the required procedure to stop it before completion and minimize the damage. The detection of APT is

different from the prediction. The detection can be when two or more steps of APT are correlated. However, the prediction can be achieved after the first two steps of APT are linked.

The detection modules have been presented in [35–42], this paper focuses on the correlation framework and prediction module.

3.3. MLAPT Detection Modules

Taking into consideration the APT steps, Table 2 shows the MLAPT detection modules for each APT step. These modules are:

Table 2: The MLAPT detection modules for each APT step.

APT step	**Detection modules**
Step 1 Intelligence gathering	This initial step includes a passive process which cannot be detected through network traffic monitoring, so there are no detection modules.
Step 2 Point of entry	Disguised exe file detection Malicious file hash detection Malicious domain name detection
Step 3 C&C communication	Malicious IP address detection Malicious SSL certificate detection Domain flux detection
Step 4 Lateral movement	This is internal traffic within the target's network. MLAPT monitors the inbound and outbound traffic, so there are no detection modules.
Step 5 Asset/Data discovery	Scanning detection
Step 6 Data exfiltration	Tor connection detection

Disguised exe File Detection (DeFD): This module detects disguised exe files over the connections. In other words, it detects if the content of the file is exe while the extension is not exe. The network traffic is processed, all connections are analysed and all exe files identified when transferring over the connections are filtered. This filtering is based on the file content. Following this, the file name extension should be checked to decide about raising an alert on disguised exe file detection [35].

Malicious File Hash Detection (MFHD): This module detects any malicious file downloaded by one of the network hosts. It is based on a blacklist of malicious file hashes. The network traffic is processed, all connections are analysed and MD5, SHA1 and SHA256 hashes are calculated for each new file identified when transferring over a connection. The calculated hashes are then matched with the blacklist [36].

Malicious Domain Name Detection (MDND): This module is used to detect any connection to a malicious domain name. It is based on a blacklist of malicious domain names. DNS traffic is filtered, all DNS requests are analysed and the queries are matched with the blacklist [37].

Malicious IP Address Detection (MIPD): This module detects any connection between an infected host and a C&C server. The detection is based on a blacklist of malicious IPs of C&C servers. MIPD processes the network traffic to search for a match in the source and destination IP addresses for each connection with the IP blacklist [38].

Malicious SSL Certificate Detection (MSSLD): This module aims at detecting C&C communications based on a blacklist of malicious SSL certificates. This blacklist consists of two forms of SSL certificates, the *SHA1 fingerprints* and the *serial & subject*, which are associated with malware and malicious activities. The network traffic is processed and all secure connections are filtered. The SSL certificate of each secure connection is then matched with the SSL certificate blacklist [39].

Domain Flux Detection (DFD): This module detects algorithmically generated domain flux, where the infected host queries for the existence of a large

number of domains, whilst the owner has to register only one. This leads to the failure of many of DNS queries. DFD utilizes DNS query failures to detect domain flux attacks. The network traffic is processed, particularly DNS traffic. All DNS query failures are analysed and a threshold for DNS query failures from the same IP address is imposed to detect domain flux attacks and identify infected hosts [40].

Scan Detection (SD): The SD module detects port scanning attacks which aims to identify the noteworthy servers and services for future data exploitation. SD is based on tracking all failed connection attempts, and a threshold for those failed attempts is imposed over a specific time interval to detect scanning attacks and identify infected hosts [41].

Tor Connection Detection (TorCD): This module detects any connection to a Tor network. It is based on a list of Tor servers which is publicly published. The network traffic is processed and the source and destination IP addresses for each connection are matched with Tor servers list [42].

3.4. FCI Correlation Framework

This phase of MLAPT takes the output of each of the detection modules (the generated alerts) as an input, and aims to find alerts could be correlated and belong to a single APT attack scenario. FCI (Filter, Cluster, and Index) runs through three main steps: (1) Alerts filter, which filters redundant or repeated alerts; (2) Alerts clustering, which clusters alerts which potentially belong to the same APT attack scenario; and (3) Correlation indexing, which evaluates the correlations between alerts of each cluster.

In Section 3.3, eight attack detection modules are presented, each module detects one possible technique used in one of the APT steps. The output of each module is an alert which is generated when an attack is detected. Each alert has seven attributes *(alert_type, timestamp, src_ip, src_port, dest_ip, dest_port, infected_host)*. Table 3 summarizes the steps of the APT attack that can be detected by MLAPT and the alerts which can be generated for each step.

All alerts generated by the detection modules are fed to the correlation

Table 3: The APT attack detectable steps and alerts.

APT step	Alerts
(A) Step 2 Point of entry	(a1) disguised_exe_alert (a2) hash_alert (a3) domain_alert
(B) Step 3 C&C communication	(b1) ip_alert (b2) ssl_alert (b3) domain_flux_alert
(C) Step 5 Asset/Data discovery	(c1) scan_alert
(D) Step 6 Data exfiltration	(d1) tor_alert

framework. However, those alerts are not the only ones detected by the the modules. When an APT technique is detected, and before an alert is generated, the module checks whether the same alert has been generated during the previous day, if so, the alert is ignored. This alerts suppression reduces the computational cost of the FCI correlation framework. The FCI process steps will be explained in this section. As an output of the FCI correlation framework, two main alerts can be generated:

- *apt_full_scenario_alert:* This alert is generated when FCI detects a *full* APT attack scenario during a specific time window, called the correlation time. This is the period in which APT is expected to complete its life cycle. A full attack scenario is one in which all possible detectable steps of an APT are detected by FCI. In other words, FCI detects four correlated steps of an APT, i.e. four different alerts each one is from a different step. Based on Table 3, and taking into consideration the APT life cycle, FCI is able to detect nine possible full scenarios of APT (APT-full). These

possible full APT scenarios can be expressed as:

$$APT_{full} = A \wedge B \wedge C \wedge D \tag{1}$$

where $A = [a_1 \vee a_2 \vee a_3]$, $B = [b_1 \vee b_2 \vee b_3]$, $C = [c_1]$ and $D = [d_1]$.

- *apt_sub_scenario_alert:* This alert is generated when FCI detects two or three, rather than all, correlated steps of an APT attack during a specific time window. In this partial attack detection scenario, alerts from one or two steps were not generated. Thus, FCI can generate two types of this alert: *apt_sub_scenario_two_steps_alert*; and *apt_sub_scenario_three_steps_alert*. FCI is able to detect forty six possible APT sub-scenarios which can be expressed as:

$$\begin{aligned} APT_{sub} = &[A \wedge (B \vee C \vee D)] \vee [B \wedge (C \vee D)] \vee [C \wedge D] \vee \\ &[(A \vee B) \wedge (C \vee D)] \vee [A \wedge B \wedge C] \vee [A \wedge C \wedge D] \vee [B \wedge C \wedge D] \end{aligned} \tag{2}$$

3.4.1. Alerts Filter (AF)

The first module of the FCI correlation framework filters redundant or repeated alerts. The AF module takes all alerts generated by the various detection modules as an input. For each new generated alert, the alerts filter checks if the alert has been generated during the correlation time window. If the new alert is the same type and has the same attributes of a recorded one, then the new alert is ignored. This filtering module reduces computational cost of the FCI correlation framework.

3.4.2. Alerts Clustering (AC)

This module clusters alerts which most likely belong to the same APT attack scenario. One cluster can represent a possible APT full or sub-scenario, i.e. it can contain one, two, three or four different alerts; each alert for a different APT step. The AC module takes the AF output, all alerts generated by the detection modules after repeated ones are filtered, as an input. All incoming alerts are

stored by AC for a correlation time. For each new alert, the AC module checks all stored alerts for the clustering possibility. The clustering algorithm in this module is scenario-based, which utilizes three main rules:

- Alert step: Alerts for the same APT attack step cannot be in one cluster.
- Alert type: Alerts of the same type cannot be in one cluster.
- Alert time: Cluster's alerts should be all triggered within the correlation time, and alerts order should be corresponded with the APT life cycle. Meaning, if *t(d), t(c), t(b)* and *t(a)* are the times when the alerts from the APT steps *six, five, three* and *two*, respectively, have been triggered, the clustering algorithm can classify those alerts into one cluster only if they meet the following two conditions:

```
t(d) > t(c) > t(b) > t(a)
t(d) - t(a) <= Correlation_time
```

The AC module has four processing engines, explained later in Section 4.2.2 on page 20, each engine processes all alerts which belong to one of the APT detectable steps. Based on the incoming alert step, a corresponded engine runs. As a result of AC process, the new incoming alert can be classified into an existing APT cluster, a new APT cluster can be created, or the new alert is ignored as it does not meet the rules and cannot be clustered at all. The output of AC is APT clusters. Each cluster contains a maximum of four alerts, which potentially belong to one APT full or sub-scenario. The produced cluster alerts are evaluated using the correlation index algorithm, presented in the following Section 3.4.3 on page 15, to decide if they are correlated or not.

3.4.3. Correlation Indexing (CI)

The third processing module evaluates the correlations between alerts in each cluster to determine if they belong to a full or sub APT attack scenario. This module has two major functions. The first function is to evaluate the correlations between alerts when building the cluster. The goal of this correlation

process is to filters clusters having uncorrelated alerts. The second function calculates the correlation index of each cluster by the end of the correlation window. The latter function is essential to build a historical record of the monitored network to be used in the next module of the FCI correlation framework, namely the prediction module.

The correlation indexing (CI) algorithm makes use of the attributes of each alert in the cluster to calculate the cluster's correlation index $Corr_{id}$. To find the $Corr_{id}$ for each cluster, the CI algorithm calculates the correlation between each two alerts (steps) in the cluster. Therefore, three values are calculated within each cluster: $Corr_{ab}$, the correlation between the second step ($alert_1$) and the third step ($alert_2$) of APT; $Corr_{bc}$, the correlation between the third step ($alert_2$) and the fifth step ($alert_3$) of APT; and $Corr_{cd}$, the correlation between the fifth step ($alert_3$) and the sixth step ($alert_4$) of APT.

The clustering algorithm is based on *alert_type* and *timestamp* attributes of each alert. However, the correlation indexing algorithm is based on *infected_host* and *scanned_host* attributes. To calculate $Corr_{ab}$, $Corr_{bc}$ and $Corr_{cd}$, taking into consideration the APT attack life cycle and the attributes of each alert in the cluster, the CI algorithm utilizes the following rules:

$$Corr_{ab} = \begin{cases} 1, & \text{if } [alert_2, infected\_host_2] = [alert_1, infected\_host_1] \\ 0, & otherwise \end{cases}$$

$$Corr_{bc} = \begin{cases} 1, & \text{if } [alert_3, infected\_host_3] = [alert_2, infected\_host_2] \\ & \text{or } [alert_3, infected\_host_3] = [alert_1, infected\_host_1] \\ 0, & otherwise \end{cases}$$

$$Corr_{cd} = \begin{cases} 1, & \text{if } [alert_4, infected\_host_4] = [alert_3, scanned\_host] \\ & \text{or } [alert_4, infected\_host_4] = [alert_3, infected\_host_3] \\ & \text{or } [alert_4, infected\_host_4] = [alert_2, infected\_host_2] \\ & \text{or } [alert_4, infected\_host_4] = [alert_1, infected\_host_1] \\ 0, & otherwise \end{cases}$$

When $Corr_{ab}$ equals to 1, this means there is a correlation between the second step and the third step of APT and the corresponding alerts can be in one cluster. When $Corr_{ab}$ equals to 0, there is no correlation and the two alerts cannot be in one cluster. And so on for $Corr_{bc}$ and $Corr_{cd}$.

The CI algorithm calculates the cluster's correlation index $Corr_{id}$ using the following equation:

$$Corr_{id} = Corr_{ab} + Corr_{bc} + Corr_{cd} \tag{3}$$

Since $Corr_{ab}$, $Corr_{bc}$ and $Corr_{cd}$ values can be only 1 or 0, the cluster's correlation index $Corr_{id}$ is always positive and can take one of the following values:

- 0; there is no correlation between any of the cluster's alerts, and the cluster's alerts cannot belong to one APT attack scenario.
- 1; there is a correlation between two different steps of an APT attack, and the cluster's alerts belong to one APT sub-scenario "*apt_sub_scenario_two_steps*".
- 2; there is a correlation between three different steps of an APT attack, and the cluster's alerts belong to one APT sub-scenario "*apt_sub_scenario_three_steps*".
- 3; there is a correlation between four different steps (all detectable steps) of an APT attack, and the cluster's alerts belong to one APT full scenario "*apt_full_scenario*".

All the clusters and their correlation index values are recorded into a specific dataset, the *correlation_ dataset*, to be used in the Prediction module.

3.5. Prediction Module (PM)

This module is used by the network security team to estimate the probability of an *apt_ sub_ scenario_ two_ steps_ alert*, generated by the FCI correlation framework, to develop a complete APT attack. In practical terms, it predicts if FCI will generate an *apt_ full_ scenario_ alert* in the future based on the attributes of the current *apt_ sub_ scenario_ alert*. This prediction gives the network security team a sign to perform more forensics on the corresponding two suspicious connections and deny the attacker to complete the APT life cycle. The prediction module uses a historical record of the monitored network and applies machine learning techniques to achieve its functionality.

PM takes the correlation dataset, built by FCI over six months or more, as an input. The required period of time to build the correlation dataset depends on the number of correlated clusters generated by FCI. This number affects the number of samples used to train the prediction model. The correlation dataset contains the correlated clusters, both full and sub APT scenarios, and the correlation index $Corr_{id}$ for each cluster. The process in this module undergoes three main steps: (1) Preparing the dataset, to be available to be consumed by machine learning algorithms; (2) Training the prediction model, different machine learning algorithms are applied and the best model which has the highest accuracy is chosen; and (3) Using the model for prediction, the security team apply the model on FCI real time alerts. The output of this module is a prediction model used by the network security team for live traffic monitor and APT prediction.

4. MLAPT Implementation

In this section, the implementation of MLAPT are introduced and the used frameworks, tools and programming languages are mentioned. As MLAPT con-

sists of three main phases: threat detection, alert correlation and attack prediction; the implementation algorithms of each phase are presented separately.

4.1. Implementation of the Detection Modules

All detection modules are implemented on top of *Bro* [43]. The implementation and evaluation of the detection modules have been published in [35–42]. Therefore, this paper presents only the implementation and evaluation of the correlation framework and prediction module.

As an output of each detection module, in case of an APT technique is detected, a corresponding event (alert) is generated. This event is to be used in the FCI correlation framework as explained later in Section 4.2 on page 19. Additionally, an alert email is sent to RT (Request tracker) [44] where the network security team can perform additional forensics and respond to the triggered alert. Along with generating a new alert, information regarding the alert and the malicious connection (*alert_type, timestamp, src_ip, src_port, dest_ip, dest_port, infected_host, malicious_item*) is written into a specific log (individual log for each APT technique detection) to keep a historical record of the monitored network.

In case of cryptographically embedded payloads for APTs paradigms, even the connections are encrypted, the detection modules (except DeFD and MFHD) are still effective as they depend on investigating the packets' headers and not the payload.

4.2. Implementation of the FCI Correlation Framework

The FCI framework is implemented in two versions. The first one is implemented on top of Bro to be used on live traffic for real time detection; it can be also used offline on PCAP (Packet Capture) files. The second version is implemented in Python to be used offline on saved alerts' logs. Using FCI offline-version is useful when having a PCAP file for a network which is not monitored by Bro.

4.2.1. Implementation of the Alerts Filter (AF) Module

When generating a new alert by one of the detection modules, the AF algorithm checks *t_detection_modules_alerts* table to determine if the same alert has been generated within the last *correlation_time*. *t_detection_modules_alerts* table contains all alerts which have been generated by the detection modules and sent to AC within the last *correlation_time*. Thus, AF either (1) ignores the new alert, if it is a repeated one; or (1) sends the new alert to AC, to be processed and clustered, and (2) writes the new alert into *t_detection_modules_alerts* table where it is saved for the next *correlation_time*. The AF algorithm pseudo-code is provided in the supplementary material of this paper.

4.2.2. Implementation of the Alerts Clustering (AC) Module

All produced APT clusters are recorded into a specific dataset, the *clustered_dataset*, to be consumed by the next module, namely the correlation indexing module. The clustering algorithm dataset "*clustered_dataset*" consists of clusters. Each cluster contains a maximum of four alerts and each alert represents one of the APT detectable steps:

1. $alert\_1 \in \{disguised\_exe\_alert, hash\_alert, domain\_alert\}$.
2. $alert\_2 \in \{ip\_alert, ssl\_alert, domain\_flux\_alert\}$.
3. $alert\_3 \in \{scan\_alert\}$.
4. $alert\_4 \in \{tor\_alert\}$.

Alert clustering can affect the performance of the correlation indexing and the prediction module as well. For this reason, the first function of CI, evaluating the correlations between the cluster's alerts, mentioned in Section 3.4.3 on page 15, is also implemented within the AC algorithm. Implementing the first function of CI within AC reduces the computational cost of the FCI correlation framework, since AC does not classify any new alert into a cluster unless it is correlated with the cluster alerts, as explained later in this section. The AC algorithm pseudo-code is provided in the supplementary material of this paper.

First, the AC module determines to which one of the APT steps the *new alert*, coming from the AF module, belongs. MLAPT can detect four steps of the APT life cycle, mentioned in Section 3.4, Table 3 on page 13. Based on the *new alert* step, AC has four processing engines, each engine processes all alerts which belong to one APT step.

For *alert_ 1 processing engine*, the second step of APT is the first detectable step, therefore, as soon as an alert of the second APT step is triggered, AC starts a new cluster and writes the new alert into *alert_ 1*.

For *alert_ 2 processing engine*, when a new alert for the third step of APT is triggered, the AC module checks all the clusters in the *clustered_ dataset*. The cluster of interest is the one that has *alert_ 1* and the other alerts (*alert_ 2, alert_ 3, alert_ 4*) are still missed. For that cluster of interest, the algorithm checks time attributes: *time*, the time when the current processed alert is triggered; and *time_ 1*, the time when the *alert_ 1* is triggered. For the new alert to be considered, those time attributes should meet two conditions: $time > time\_1$ and $time - time\_1 <= TW$; whereas TW stands for the time window "correlation time". Following this, the first function of the CI module checks the *infected_ host* attributes: *infected*, the infected host of the current processed alert; and *infected_ 1*, the infected host of alert_1. If both infected host attributes are matched, the current processed alert is added into the current cluster of interest as *alert_ 2*. In addition, an event *apt_ sub_ scenario_ two_ steps_ alert* is generated and an alert email is sent to RT informing the network security team regarding this APT sub scenario detection. When one of the previous checks fails, AC checks if the current cluster is the last one in the *clustered_ dataset*: if true, a new cluster is started and the current processed alerts is added as *alert_ 2*; if false, the process is to be repeated again for the next cluster.

For *alert_ 3 processing engine*, when a new alert for the fifth step of APT is triggered, AC checks all the clusters in the *clustered_ dataset*. There are three cases for the cluster of interest: (1) when the cluster has *alert_ 1* and *alert_ 2* and the other alerts "*alert_ 3* and *alert_ 4*" are missed; (2) when the cluster has *alert_ 1* and the other alerts "*alert_ 2*, *alert_ 3*, *alert_ 4*" are missed; (3) and

when the cluster has *alert_ 2* and the other alerts "*alert_ 1*, *alert_ 3*, *alert_ 4*" are missed.

For the first case of cluster of interest, AC checks all time attributes which should meet two conditions: *time* > *time_ 2* and *time* - *time_ 1* <= *TW*. Following this, CI checks all infected host attributes that should meet the condition *infected* == *infected_ 2*, as *alert_ 1* and *alert_ 2* are already in the cluster so it is guaranteed that *infected_ 1* == *infected_ 2* and there is no need for the first function of CI to check it. The current processed alert is then added into the current cluster of interest as *alert_ 3*, an event *apt_ sub_ scenario_ three_ steps_ alert* is generated, and an alert email is sent to RT informing the network security team regarding this APT sub-scenario detection. If one of the previous checks is failed, it is checked if the current cluster is the last one in *clustered_ dataset*: if true, a new cluster is started and the current processed alerts is added as *alert_ 3*; if false, the process is to be repeated again for the next cluster.

For the second and third case of cluster of interest, the process is similar to the first case, taking into consideration the corresponded time and infected host attributes.

For *alert_ 4 processing engine*, the first step is to find the cluster of interest in the *clustered_ dataset*. When a new alert for the sixth step of APT is triggered, AC checks all the clusters in the *clustered_ dataset*. There are seven cases for the cluster of interest: (1) when the cluster has *alert_ 1*, *alert_ 2*, and *alert_ 3*, and the last alert "*alert_ 4*" is missed; (2) when the cluster has *alert_ 1* and *alert_ 2* and the other alerts "*alert_ 3* and *alert_ 4*" are missed; (3) when the cluster has *alert_ 1* and *alert_ 3* and the other alerts "*alert_ 2* and *alert_ 4*" are missed; (4) when the cluster has *alert_ 2* and *alert_ 3* and the other alerts "*alert_ 1* and *alert_ 4*" are missed; (5) when the cluster has *alert_ 1* and the other alerts "*alert_ 2*, *alert_ 3*, and *alert_ 4*" are missed; (6) when the cluster has *alert_ 2* and the other alerts "*alert_ 1*, *alert_ 3*, and *alert_ 4*" are missed; (7) and when the cluster has *alert_ 3* and the other alerts "*alert_ 1*, *alert_ 2*, and *alert_ 4*" are missed.

The process of all cases of cluster of interest in *alert_ 4* processing engine

is similar to the process in *alert_3* processing engine explained above. The AC algorithm checks all time attributes of the cluster; after that, the CI algorithm checks all infected host and scanned host attributes; to decide whether the current processed alert is to be added into the current cluster of interest as *alert_4*. Based on the cluster of interest, three events can be generated as an output of *alert_4* processing engine: *apt_full_scenario_alert* for case 1; *apt_sub_scenario_three_steps_alert* for cases 2, 3, and 4; and *apt_sub_scenario_two_steps_alert* for cases 5, 6, and 7. In addition, an alert email is sent to RT informing the network security team regarding this APT full or sub-scenario detection. If one of the algorithms' conditions fails, the process moves to the next cluster in *clustered_dataset* or it is ended if the current cluster is the last one.

4.2.3. Implementation of the Correlation Indexing (CI) Module

The first function of CI, evaluation the correlations between the cluster's alerts, is implemented within AC algorithm, as explained in the previous Section 4.2.2 on page 20. For the second function of CI, to calculate $Corr_{id}$ for each cluster, the CI algorithm makes use of the attributes of each alert in the cluster, applies the correlation rules mentioned in Section 3.4.3 on page 15, and calculates the correlation index $Corr_{id}$ based on the equation 3 mentioned also in Section 3.4.3 on page 17. The CI algorithm pseudo-code is provided in the supplementary material of this paper.

4.3. Implementation of the Prediction Module (PM)

The PM module uses machine learning techniques to achieve its functionality. The process in this module undergoes three main steps: (1) Preparing the dataset, implemented in Python; (2) Training the prediction model, implemented in Matlab ; and (3) Using the model for prediction, in Python and Matlab.

4.3.1. Preparing the Machine Learning Dataset

Building the *machine_ learning_ dataset* is based on the *correlation_ dataset*, which is the output of the FCI correlation framework over a period of six months or more. The *correlation_ dataset* contains the correlated clusters, both full and sub APT scenarios, and the correlation index $Corr_{id}$ for each cluster. To prepare the *machine_ learning_ dataset*, PM makes the following modifications on the *correlation_ dataset*:

- The prediction of *apt_ sub_ scenario_ two_ steps_ alert* to complete the APT life cycle is based on the first two detectable steps of APT, therefore, only the clusters containing at least alerts for the first two detectable steps, i.e. *alert_ 1* and *alert_ 2*, are kept; the other clusters are filtered out of the *correlation_ dataset*.

- Based on the $Corr_{id}$ value, the *correlation_ dataset* clusters can be classified into four classes: class 3, for APT full scenario and the cluster has four correlated alerts; class 2, for APT sub-scenario and the cluster has three correlated alerts; class 1, for APT sub-scenario and the cluster has two correlated alerts; and class 0, the cluster has only one alert. The *machine_ learning_ dataset* contains only two classes: class 1 for APT full scenario; and class 0, for uncompleted APT scenario. Thus, the PM module considers: (1) class 3, in the *correlation_ dataset*, as class 1, for the *machine_ learning_ dataset*; and (2) classes 2, 1, and 0, in the *correlation_ dataset*, as class 0, for the *machine_ learning_ dataset*.

- The class prediction is based on the first two detectable steps of APT, therefore, all the columns related to the third and fourth detectable alerts, i.e. *alert_ 3* and *alert_ 4* attributes, are filtered out of the *correlation_ dataset*.

- Since the chosen machine learning classifiers work with numeric values, columns which are not numeric in the *correlation_ dataset* are represented in a numerical format for the *machine_ learning_ dataset*. The *alert_ type*

values are mapped to numbers from 1 to 6, and the columns which contain IPs values (*src_ip_1, dest_ip_1, infected_host_1, src_ip_2, dest_ip_2, infected_host_2*) are mapped to numeric values using *socket* [45] and *struct.unpack* [46] functions built in Python.

4.3.2. Training the Prediction Model

As the task is to predict classes, classification methods are chosen and different machine learning algorithms are applied on *machine_learning_dataset* to train the model. The model is trained using four machine learning approaches, commonly used for classification problems, which are: decision tree learning, support vector machine, k-nearest neighbours and ensemble learning. The prediction accuracy of each trained model is calculated and the best model, which has the higher prediction accuracy, is chosen. The best model is saved to be used by the network security team.

4.3.3. Using the Model for Prediction

When a new *apt_sub_scenario_two_steps_alert* is generated by the correlation framework, the new data, i.e. the cluster attributes, is prepared as explained above in Section 4.3.1 on page 24, then the prediction model, which has been trained and chosen in the previous step, is applied.

As a result, the network security team can determine the probability of the current alert to complete the APT life cycle, and apply the required procedure to stop the attack before completion and achieving the final aim of data exfiltraition.

5. Experimental Evaluation of MLAPT

In this section, the evaluation of MLAPT is introduced and the achieved results are presented. As MLAPT consists of three main phases: threat detection, alert correlation and attack prediction; the evaluation of MLAPT undergoes the evaluation of the three phases respectively. Additionally, a comparison between the developed system MLAPT and other existing systems is provided.

5.1. Evaluation of the Detection Modules

Two main methods were used to evaluate the detection modules. In the first one, the detection modules were applied on pcap files which contain malicious traffic. Each pcap file was provided by a different third party, pcap file size and data source are mentioned in the evaluation section of each detection module. In the second evaluation method, Bro was installed on an experimental server (2x 4-core Intel Xeon CPU E5530 @ 2.40 GHz, 12 GB RAM) with passive access to part of the university campus live traffic (200 Mbps, 200 users, 550 nodes) via an optical TAP (Test Access Port). The detection modules were run on the experimental server and the network was monitored for one month.

5.2. Evaluation of the FCI Correlation Framework

In the absence of any publicly available data which contains APT attack traffic, which can be used in the evaluation of the FCI framework. We had to build a new dataset which contains APT attack traffic. Using the campus network to gather attack data does not guarantee capturing any APT attack traffic against the monitored network.

The aim of the correlation framework is to identify different alerts raised by the various detection modules, which could be correlated and belong to one APT attack scenario. To effectively evaluate the FCI correlation framework, a dataset containing many of the detection modules alerts, in which some of those alerts belong to APT attack scenarios, has been built. The data is generated to appear as APT attack scenarios were simulated on the campus network, the techniques used in the APT life cycle were identified by the detection modules, and all generated alerts were written into the *simulation dataset*. That dataset also contains many of the generated alerts which do not belong to APT attack scenarios. All the detection modules have been evaluated on pcap files and on the real live traffic as well. The aim of this experiment is to test if the FCI correlation framework is able to detect those APT scenarios in the *simulation dataset*.

5.2.1. Data Generation

A script is written, using Python. This script generates two types of alerts: (1) Random alerts which do not relate or belong to one APT attack scenario; and (2) Related alerts which belong to a full or sub-APT attack. Each alert has seven attributes: *alert_type, timestamp, src_ip, src_port, dest_ip, dest_port* and the *infected_host*; only the *scan_alert* has the extra *scanned_host* attribute.

To generate a random alert, the *alert_type* is selected randomly from the set of all 8 detectable alerts, i.e. *disguised_exe_alert, hash_alert, domain_alert, ip_alert, ssl_alert, domain_flux_alert, scan_alert* and *tor_alert*. The *timestamp* is assigned a random value between Fri, 01 Jan 2016 00:00:01 GMT and Thu, 30 Jun 2016 23:59:59 GMT. The *src_ip* is randomly assigned an IP address on the campus network. The *src_port* is selected randomly from the 49152, 65535 range of ports, which are usually assigned dynamically to client applications when initiating a connection. The *dest_ip* value is assigned based on the selected *alert_type*: If the *alert_type* is *disguised_exe_alert*, *hash_alert* or *ssl_alert*, then the *dest_ip* can be any valid IP address which is not on the campus network; if the *alert_type* is *domain_alert* or *domain_flux_alert*, then the *dest_ip* can use an IP address which is on the campus network; if the *alert_type* is assigned *ip_alert*, then the *dest_ip* can select a random IP address from the *ip_blacklist*; if *alert_type* is *scan_alert*, the *dest_ip* is selected randomly from campus network IP addresses; and if the *alert_type* is *tor_alert*, the *dest_ip* is selected randomly from *tor_server_list*. The *dest_port* is selected based on the selected *alert_type*: if the *alert_type* is *disguise_exe_alert* or *hash_alert*, the *dest_port* should be 80; if the *alert_type* is *domain_alert* or *domain_flux_alert*, the *dest_port* should be 53; if the *alert_type* is *ip_alert*, *ssl_alert* or *tor_alert*, the *dest_port* should be 443; and if the *alert_type* is *scan_alert*, the *dest_port* is selected randomly from the 1, 1024 range of ports. The *infected_host* should be the same *src_ip* of the connection. Finally, the *scanned_host* (only if *alert_type* is *scan_alert*) should be the same *dest_ip* of the connection.

To generate an APT full-scenario (consisting of 4 correlated alerts) or sub-scenario (consisting of 2 or 3 correlated alerts), the APT life cycle should be taken into consideration. Meaning, the generated alerts' attributes of each scenario are selected to appear as an APT attack which is simulated through the campus network.

5.2.2. Experimental Setup

To determine the number of random alerts to be generated for the *simulation_ dataset*, the experimental server, previously mentioned in Section 5.1 on page 26, was used to monitor part of the university campus network. All detection modules were run on the experimental server to analyse the network traffic; the monitoring period and the number of detected alerts were determined. According to the actual university network size and the actual *simulation_ dataset* monitoring period, the number of the generated random alerts was calculated. The number of the generated APT full- and sub-scenarios should be suitable to get enough samples for each class in the *machine_ learning_ dataset* previously explained in Section 4.3.1 on page 24.

The network monitoring was conducted via the experimental server for 2 weeks and 9 different alerts were detected by the detection modules. The size of the monitored network was 550 nodes, while the whole campus network is 23500 nodes. Meaning, if the whole campus network is monitored for 6 months, 4900 alerts are expected to be detected by the detection modules. Therefore, 4900 alerts were generated for the *simulation_ dataset*, of which 100 APT full attack (each scenario is 4 correlated alerts) and 50 APT sub-attack 3 steps (each scenario is 3 correlated alerts); 50 APT sub-scenarios 2 steps (each scenario is 2 correlated alerts); and 4250 random alerts (which do not relate or belong to APT attack scenarios). The APT life cycle period was configured to be for a maximum of one week.

5.2.3. Results and Discussion

The FCI correlation framework was applied on the *simulation_ dataset*. Table 4 shows the FCI correlation framework detection results. This table indicates the *True Positive Rate (TPR)* and the *False Positive Rate* (FPR) [47] for each studied APT attack, both full and partial attacks. Among all studied APT attacks, the best TPR results were for the APT sub-attack two steps scenario, followed by the APT sub-attack three steps scenario and APT full attack, respectively. The results show that the higher the number of related alerts, the lower the TPR and the higher FPR. This is due to the higher possibility of the random alerts to be incorrectly clustered when more alerts are to be correlated for APT. By manual analysis for the results, the incorrect alerts clustering was the main reason of the false alarms. Some APT attacks were not detected due to some of the random alerts which were incorrectly clustered and correlated. This can happen if those random alerts, by chance, meet the clustered and correlation rules, so one random alert can interfere with a running APT scenario (if the random alert is triggered for the missed scenario step, for the same infected host, and within the correlation time) and cause the false positive detection of APT and false negative detection of the random alert. Besides, a very rare case can cause the wrong detection is when two, three or four random alerts can meet the correlation rules, by chance, and are correlated incorrectly.

Table 4: Correlation framework detection results.

APT attack scenario	**Detection result**	**TP**	**FP**	**FN**	**TN**	**P**	**N**	**TPR**	**FPR**
APT full scenario (4 steps)	90*4	78*4	12*4	88	4452	400	4500	78%	1%
APT sub-scenario (3 steps)	65*3	42*3	23*3	24	4681	150	4750	84%	1.4%
APT sub-scenario (2 steps)	85*2	47*2	38*2	6	4724	100	4800	94%	1.6%
APT full and sub-scenario	725	532	193	118	4132	650	4250	81.8%	4.5%

5.3. Evaluation of the APT Prediction Module (PM)

To evaluate the PM module, three main steps were followed: (1) Preparing the *machine_ learning_ dataset*; (2) Training the prediction model; and (3) Saving the model for prediction.

Using the *correlation_ dataset*, which is the output of the FCI correlation framework over a period of six months, the *machine_ learning_ dataset* is prepared as explained in Section 4.3.1 on page 24.

As there is no machine learning algorithm which can be regarded as the best or the optimal one, various experiments should be performed on the *machine_ learning_ dataset* using several machine learning algorithms, and then a comparison between the trained models is made.

The Matlab's Classification Learner application is used to train models to classify the *machine_ learning_ dataset*. Automated training is performed to search for the best classification model type, including decision trees, support vector machines, nearest neighbours, and ensemble classification; the characteristics of each classifier type can be found in [48]. Cross-validation is used as a validation scheme to examine the prediction accuracy of each trained model.

Cross-validation is a model assessment technique used to evaluate a machine learning algorithm's performance in making predictions on new datasets which has not been trained on . This is done by partitioning a dataset and using a subset to train the algorithm and the remaining data for testing. Each round of cross-validation involves randomly partitioning the original dataset into a *training set* and a *testing set*. The training set is then used to train a supervised learning algorithm and the testing set is used to evaluate its performance. This process is repeated several times and the average accuracy is used as a performance indicator. Table 5 shows the prediction accuracy for all investigated classification algorithms used to train the classification models.

Experimental results show that the best classification algorithm is the *Linear SVM*, with a prediction accuracy of 84.8%. This trained model can be saved by the network security team to be applied on real time traffic when a new real

Table 5: Classification algorithms and the prediction accuracy of the trained models.

Classification algorithms		**Prediction accuracy**
Decision trees	Complex tree	83.0%
	Medium tree	83.0%
	Simple tree	84.4%
Support vector machines	Linear SVM	84.8%
	Quadratic SVM	81.6%
	Cubic SVM	76.9%
	Fine Gaussian SVM	69.4%
	Medium Gaussian SVM	80.3%
	Coarse Gaussian SVM	81.0%
Nearest neighbour classifiers	Fine KNN	76.2%
	Medium KNN	80.3%
	Coarse KNN	68.0%
	Cosine KNN	82.3%
	Cubic KNN	78.9%
	Weighted KNN	78.2%
Ensemble classifiers	Boosted trees	83.7%
	Bagged trees	82.3%
	Subspace discriminant	81.6%
	Subspace KNN	72.8%
	RUSBoosted trees	81.0%

time *apt_sub_scenario_two_steps_alert* is triggered, as previously explained in Section 4.3.3 on page 25.

6. A Performance Comparison Between the Proposed Approach and Existing APT Detection Systems

This section presents a performance analysis of four existing APT detection systems, and provides a comparison between the developed system MLAPT and these current systems, as shown in Table 6.

Table 6: A comparison between MLAPT and other existing systems.

APT detection system	Autonomy	APT steps	speed	TPR	FPR	Prediction accuracy
MLAPT	Autonomous	4	Real time	81.8%	4.5%	84.8%
TerminAPTor	Agent-based	4	Real time	100%	high	No
C&C-based	Autonomous	1	Off-line	83.3%	0%	No
Spear phishing based	Autonomous	1	Real time	97.2%	14.2%	No
Context-based	Agent-based	4	Real time	?	27.88%	No

The most effective system in terms of true positive rate is TerminAPTor [26] with a TPR of 100%, previously mentioned in Section 2 on page 4. However, the developers mentioned that TerminAPTor has a high rate of false positives (although they did not mention the figure of FPR) and needs to be improved by filtering the false positives. Moreover, this detector requires the alerts to be provided by other systems (agent-based) and cannot work autonomously. Despite having the lowest false positive rate of 0%, the C&C-based system [27], presented previously in Section 2 on page 5, does not achieve the real time detection. Furthermore, the authors stated that the detection can be easily evaded when the infected hosts connect to the C&C domains while users are surfing the Internet. Additionally, missing the detection of C&C domains leads to failure in APT detection since this system depends on detecting only one step of the APT life cycle. Whilst the spear phishing based system [28], explored earlier in Section 2 on page 6, has a TPR of 97.2%, the FPR of 14.2% is considerably high. In addition, depending on one step for APT detection leads the system to fail when missing the spear phishing email detection. This missing can happen when the spear phishing email does not include any of the tokens which are necessary for the algorithm process. The context-based system [32], already stated in Section 2 on page 7, has a significantly high FPR of 27.88% while the TPR

was not provided by the authors. Besides, this framework requires significant expert knowledge to set up and maintain; and similar to TerminAPTor, it is an agent-based system and cannot work autonomously.

Having a high rate of true positives is significant. Nevertheless, increasing the amount of true positives means that the false positive rate also increases. Thus, the balance between TPR and FPR is an essential requirement for any detection system. The developed system MLAPT has a suitable balance between the two values of TPR and FPR with 81.8% and 4.5% respectively. MLAPT can also work autonomously and generate the required events based on its own detection modules. The generated events covers four detectable steps of the APT life cycle which reduces the false positives and gives more possibility of APT detection in case one of the steps is missed. Furthermore, this system can achieve the real time detection, so it can be much easier to trace back to the attacker, minimise the damage and prevent further break-ins. Moreover, to the author's knowledge, MLAPT is the only system which can predict APT in its early steps with a prediction accuracy of 84.8%, which prevents the attacker from achieving the goal of data exfiltration.

7. Conclusion and Future Work

The volume, sophistication, and variety of cyber attacks including APT attacks are increasing exponentially on a global scale. There is an urgent need to develop an efficient system for fast and accurate detection of attacks for quick response and defense. This paper has developed a novel machine learning based system (MLAPT) to detect and predict APT attacks in a holistic approach. The MLAPT consists of three main phases: threat detection, alert correlation and attack prediction. The contributions of the MLPT are

- In the alert correlation, we have developed correlation framework which can link the alerts produced in the first phase with the APT attacks to ensure the reduction of false positive rate.

- In the final phase, a machine-learning-based prediction module (PM) is designed and implemented based on a historical record of the monitored network.

- The proposed system is capable of accurately capture attacks in a timely fashion.

MLAPT is experimentally evaluated and its performance is compared against four of its most prominently cited rivals according to recent literature. Evaluation results show that MLAPT balances the true positive rate and the false positive rate with 81.8% and 4.5% respectively.

Some of the developed detection modules (i.e. the blacklist-based modules) require a continuous update and may not work consistently. For future work, a number of improvements within the system could be made. First, it is suggested that more detection modules are added to detect other techniques used in the APT attack life cycle. Furthermore, if MLAPT were able to monitor the internal network traffic, other detection modules could be added to detect brute force and pass the hash attacks, increasing the detectable steps of the the system. Second, it is also recommended that more than one detection module for the same technique are developed. Third, it is advised that alerts from external IDSs deployed on the network are received and fed to MLAP, which can reduce the false positive rate of the system. Fourth, MLAP detection modules were evaluated on real traffic and pcap files contain real attacks. However, the FCI framework was validated on simulated data. Therefore, it would be beneficial to test MLAP on real APTs. Nevertheless, obtaining such data is not easy, and the lack of relevant publicly available data sources was the main reason for using the synthetic data when evaluating the correlation framework.

References

[1] M. Conti, A. Dehghantanha, K. Franke, S. Watson, Internet of things security and forensics: Challenges and opportunities (2018).

[2] A. MacDermott, T. Baker, Q. Shi, Iot forensics: Challenges for the ioa era, in: New Technologies, Mobility and Security (NTMS), 2018 9th IFIP International Conference on, IEEE, 2018, pp. 1–5.

[3] H. HaddadPajouh, A. Dehghantanha, R. Khayami, K.-K. R. Choo, A deep recurrent neural network based approach for internet of things malware threat hunting, Future Generation Computer Systems.

[4] S. Morgan, Hackerpocalypse: A cybercrime revelation, 2016 Cybercrime Report, Cybersecurity Ventures.

[5] G. Epiphaniou, P. Karadimas, D. K. B. Ismail, H. Al-Khateeb, A. Dehghantanha, K.-K. R. Choo, Non-reciprocity compensation combined with turbo codes for secret key generation in vehicular ad hoc social iot networks, IEEE Internet of Things Journal.

[6] S. Walker-Roberts, M. Hammoudeh, A. Dehghantanha, A systematic review of the availability and efficacy of countermeasures to internal threats in healthcare critical infrastructure, IEEE Access.

[7] F. J. Aparicio-Navarro, K. G. Kyriakopoulos, Y. Gong, D. J. Parish, J. A. Chambers, Using pattern-of-life as contextual information for anomaly-based intrusion detection systems, IEEE Access 5 (2017) 22177–22193.

[8] H. M. Al-Khateeb, G. Epiphaniou, Z. A. Alhaboby, J. Barnes, E. Short, Cyberstalking: Investigating formal intervention and the role of corporate social responsibility, Telematics and Informatics 34 (4) (2017) 339–349.

[9] A. Salem, K. A. Hamdi, K. M. Rabie, Physical layer security with rf energy harvesting in af multi-antenna relaying networks, IEEE Transactions on Communications 64 (7) (2016) 3025–3038.

[10] O. S. Badarneh, P. C. Sofotasios, S. Muhaidat, S. L. Cotton, K. Rabie, N. Al-Dhahir, On the secrecy capacity of fisher-snedecor f fading channels, arXiv preprint arXiv:1805.09260.

[11] T. M. technical report, Targeted attacks and how to defend against them, http://www.trendmicro.co.uk/media/misc/targeted-attacks-and-how-to-defend-against-them-en.pdf, accessed: 05-12-2017.

[12] I. Ghafir, V. Prenosil, Advanced persistent threat attack detection: An overview, International Journal of Advances in Computer Networks and Its Security (IJCNS) vol. 4 (Issue 4) (2014) 50–54.

[13] T. R. Rakes, J. K. Deane, L. Paul Rees, It security planning under uncertainty for high-impact events, Omega 40 (1) (2012) 79–88.

[14] P. Wood, M. Nisbet, G. Egan, N. Johnston, K. Haley, B. Krishnappa, T.-K. Tran, I. Asrar, O. Cox, S. Hittel, et al., Symantec internet security threat report trends for 2011, Volume XVII.

[15] M. I. Center, Apt1: Exposing one of china's cyber espionage units, Tech. rep., Mandiant, Tech. Rep (2013).

[16] K. L. ZAO, Red october diplomatic cyber attacks investigation, http://www.securelist.com/en/analysis/204792262/Red_October_Diplomatic_Cyber_Attacks_Investigation, accessed: 10-11-2017.

[17] C. Tankard, Advanced persistent threats and how to monitor and deter them, Network security 2011 (8) (2011) 16–19.

[18] R. Deibert, R. Rohozinski, Tracking ghostnet: Investigating a cyber espionage network, Information Warfare Monitor (2009) 6.

[19] S.-T. Liu, Y.-M. Chen, S.-J. Lin, A novel search engine to uncover potential victims for apt investigations, in: Network and Parallel Computing, Springer, 2013, pp. 405–416.

[20] O. Thonnard, L. Bilge, G. O'Gorman, S. Kiernan, M. Lee, Industrial espionage and targeted attacks: Understanding the characteristics of an escalating threat, in: Research in Attacks, Intrusions, and Defenses, Springer, 2012, pp. 64–85.

[21] M. Lee, D. Lewis, Clustering disparate attacks: Mapping the activities of the advanced persistent threat., in: Proceedings of the 21st Virus Bulletin International Conference.(October 2011) pp, pp. 122–127.

[22] B. Bencsáth, G. Pék, L. Buttyán, M. Félegyházi, Duqu: Analysis, detection, and lessons learned, in: ACM European Workshop on System Security (EuroSec), Vol. 2012, 2012.

[23] M. Balduzzi, V. Ciangaglini, R. McArdle, Targeted attacks detection with spunge.

[24] C. Moxey, M. Edwards, O. Etzion, M. Ibrahim, S. Iyer, H. Lalanne, M. Monze, M. Peters, Y. Rabinovich, G. Sharon, et al., A conceptual model for event processing systems, IBM Redguide publication.

[25] L. Brenna, A. Demers, J. Gehrke, M. Hong, J. Ossher, B. Panda, M. Riedewald, M. Thatte, W. White, Cayuga: a high-performance event processing engine, in: Proceedings of the 2007 ACM SIGMOD international conference on Management of data, ACM, 2007, pp. 1100–1102.

[26] G. Brogi, V. V. T. Tong, Terminaptor: Highlighting advanced persistent threats through information flow tracking, in: New Technologies, Mobility and Security (NTMS), 2016 8th IFIP International Conference on, IEEE, 2016, pp. 1–5.

[27] X. Wang, K. Zheng, X. Niu, B. Wu, C. Wu, Detection of command and control in advanced persistent threat based on independent access, in: Communications (ICC), 2016 IEEE International Conference on, IEEE, 2016, pp. 1–6.

[28] J. V. Chandra, N. Challa, S. K. Pasupuleti, A practical approach to e-mail spam filters to protect data from advanced persistent threat, in: Circuit, Power and Computing Technologies (ICCPCT), 2016 International Conference on, IEEE, 2016, pp. 1–5.

[29] J. Sexton, C. Storlie, J. Neil, Attack chain detection, Statistical Analysis and Data Mining: The ASA Data Science Journal 8 (5-6) (2015) 353–363.

[30] N. Nissim, A. Cohen, C. Glezer, Y. Elovici, Detection of malicious pdf files and directions for enhancements: a state-of-the art survey, Computers & Security 48 (2015) 246–266.

[31] J. Sigholm, M. Bang, Towards offensive cyber counterintelligence: Adopting a target-centric view on advanced persistent threats, in: Intelligence and Security Informatics Conference (EISIC), 2013 European, IEEE, 2013, pp. 166–171.

[32] P. Giura, W. Wang, A context-based detection framework for advanced persistent threats, in: Cyber Security (CyberSecurity), 2012 International Conference on, IEEE, 2012, pp. 69–74.

[33] E. Alomari, S. Manickam, B. Gupta, M. Anbar, R. M. Saad, S. Alsaleem, A survey of botnet-based ddos flooding attacks of application layer: Detection and mitigation approaches, in: Handbook of Research on Modern Cryptographic Solutions for Computer and Cyber Security, IGI Global, 2016, pp. 52–79.

[34] S. M. Milajerdi, M. Kharrazi, A composite-metric based path selection technique for the tor anonymity network, Journal of Systems and Software 103 (2015) 53–61.

[35] I. Ghafir, V. Prenosil, M. Hammoudeh, F. J. Aparicio-Navarro, K. Rabie, A. Jabban, Disguised executable files in spear-phishing emails: Detecting the point of entry in advanced persistent threat, in: Proceedings of International Conference on Future Networks and Distributed Systems, ACM Digital Library, 2018.

[36] I. Ghafir, V. Prenosil, Malicious file hash detection and drive-by download attacks, in: Proceedings of the Second International Conference on Computer and Communication Technologies, Springer, 2016, pp. 661–669.

[37] I. Ghafir, V. Prenosil, Dns traffic analysis for malicious domains detection, in: 2nd International Conference on Signal Processing and Integrated Networks (SPIN), IEEE Xplore Digital Library, 2015, pp. 613–918.

[38] I. Ghafir, V. Prenosil, Blacklist-based malicious ip traffic detection, in: Global Conference on Communication Technologies (GCCT), IEEE Xplore Digital Library, 2015, pp. 229–233.

[39] I. Ghafir, V. Prenosil, M. Hammoudeh, L. Han, U. Raza, Malicious ssl certificate detection: A step towards advanced persistent threat defence, in: Proceedings of International Conference on Future Networks and Distributed Systems, ACM Digital Library, Cambridge, UK, 2017.

[40] I. Ghafir, V. Prenosil, Dns query failure and algorithmically generated domain-flux detection, in: International Conference on Frontiers of Communications, Networks and Applications (ICFCNA), IEEE Xplore Digital Library, 2014, pp. 1–5.

[41] Bro-Project, TCP scan detection, `https://www.bro.org/sphinx/scripts/policy/misc/scan.bro.html`, accessed: 12-01-2018.

[42] I. Ghafir, J. Svoboda, V. Prenosil, Tor-based malware and tor connection detection, in: International Conference on Frontiers of Communications, Networks and Applications (ICFCNA), IEEE Xplore Digital Library, 2014, pp. 1–6.

[43] V. Paxson, Bro: a system for detecting network intruders in real-time, Computer networks 31 (23) (1999) 2435–2463.

[44] Best-Practical-Solutions, Rt: Request tracker, `https://www.bestpractical.com/rt/`, accessed: 15-02-2017.

[45] Python-Software-Foundation, socket, low-level networking interface, `https://docs.python.org/3/library/socket.html`, accessed: 15-03-2018.

[46] Python-Software-Foundation, struct, interpret strings as packed binary data, https://docs.python.org/2/library/struct.html, accessed: 15-03-2018.

[47] M. Elhamahmy, H. N. Elmahdy, I. A. Saroit, A new approach for evaluating intrusion detection system, CiiT International Journal of Artificial Intelligent Systems and Machine Learning 2 (11).

[48] The-MathWorks, Characteristics of classifier types, https://uk.mathworks.com/help/stats/choose-a-classifier.html#bunt0ky, accessed: 15-03-2018.