Advanced Persistent Threat Detection using Anomaly Score Calibration and Multi-class Classification

by

Ornella Soh

A thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

Advanced Persistent Threat Detection using Anomaly Score Calibration and Multi-class Classification

by

Ornella Soh

Supervisory Committee

Dr. Issa Traore , Supervisor
(Department of Electrical and Computer Engineering)

Dr. Sherif Saad , Adjunct Prof.
(Department member)

Supervisory Committee

Dr. Issa Traore , Supervisor
(Department of Electrical and Computer Engineering)

Dr. Sherif Saad , Adjunct Prof.
(Department member)

ABSTRACT

Organisations worldwide continue to face a diverse range of attacks. Traditionally, these have been attacks of opportunity that quickly act upon weaker targets whenever possible. However, in the past decade, advanced persistent threats (APTs) have emerged that consist of targeted and long-term campaigns perpetrated by skilled and determined hackers who have clearly defined objectives and relentlessly work towards achieving their aims. APT breaches can go undetected for long periods because of the hackers' ability to adapt to and escape defensive methods. In this paper, we present a new approach to establishing whether a security event is part of an APT attack by predicting the corresponding kill chain stage. For monitored security activity and events, our approach derives a probabilistic anomaly score using an approach based on principal component analysis (PCA) and score calibration and classifying the event with a multi-class type of Bayesian Network (BN). We evaluate the proposed model using two different public APT datasets, which yielded very encouraging performance in accurately detecting APT event occurrences and stages.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my supervisor Dr. Issa Traore for providing the guidance, suggestions, support, encouragement and direction I needed to start and finish this document.

I am greatly indebted to Dr. Sherif Saad for his technical advice. I would like to thank the lecturers for giving me the background that was necessary to even attempt this. I would like to thank all classmates without whom I probably would not have been able to complete this program.

Finally, I would like to thank my family, without whose love and support, this would never have been started.

DEDICATION

I would like to dedicate my thesis work to my husband for its support, motivation and guidance.

Chapter 1

Introduction

1.1 Context

Advanced persistent threats (APTs) [2, 14, 17] are targeted, long-term hacking campaigns perpetrated by highly skilled and determined attackers with a very specific objective – namely sabotage or data exfiltration – and that adapt to defenders' efforts to resist it.

Some of the notorious authors of APTs are rogue states and criminals whose primary goal is financial gain. Other objectives of APT campaigns may include geopolitics and cyber-espionage. To operate, APT threat actors choose a variety of targets, including government and diplomatic agencies, financial sectors, or technology hardware sectors, just to name a few.

APT authors may work either alone or in groups using sophisticated malware and techniques to reach their goals. They are persistent as they try several approaches until reaching a successful outcome, e.g., mining and exfiltrating sensitive data without being detected. Successful APTs may result in serious consequences for the reputation of targeted organizations, such as loss of customers and partners, disclosure of confidential information, and financial loss. Attackers can move up an organization's hierarchy, and compromise staff members or delete entire databases to avoid detection.

1.2 Thesis Overview

A common characteristic of APT campaigns is their relatively long-term nature and the effort made by attackers to adapt to and escape any defense using various stealthy techniques. As a result, existing intrusion detection systems (IDS) and intrusion response systems (IRS) struggle to detect and mitigate APTs. Current IDSs rely mostly on homogeneous data sources. Although these data sources can be used for detecting more traditional attack methods, which tend to focus on time and target area, they are ineffective against APTs, which are more scattered in their time and delivery. Typical APTs unravel as a series or succession of events and stages which, when considered in isolation, may seem innocuous, but when linked together lead to damaging consequences over time. The scattered nature of APTs poses a formidable challenge in defending against such threats. In light of such challenge, the main research objective of this thesis is to develop an effective defense strategy by identifying and categorizing the stages of an APT campaign using machine learning.

One strategy for successfully defending against APTs is to map each of these events to the relevant cyber kill chain stage; the higher the cyber kill chain stage the more ominous the event, and therefore the more urgent is the need to address it. This allows categorizing and prioritizing the events making up the stages of an ongoing APT campaign and deploying an efficient and effective defense to counter and disrupt the campaign in a timely manner.

In this thesis, we propose a new approach to systematically detecting and mapping an APT event by generating a probabilistic score using *score calibration* and multi-class classification with a Bayesian network model. There are some advantages while considering Bayesian Network (BN). First, it performs an explicit treatment of uncertainty and provides a prediction interval. Second, BN avoid the problem of over-fitting faced by various machine learning models by addressing the regularization properties. Furthermore, we can interpret the results by explaining the reasons and process of how the model was built. The experimental evaluation of the proposed approach was conducted using two public APT datasets at two different levels of abstraction: the dataset for APT 2020 (DAPT2020) of raw events generated from targeted network appliances and a dataset of meta alerts partly generated by security information and event management (SIEM) system. The evaluation yielded very encouraging performance results.

1.3 Thesis contributions

In this thesis, two major contributions are made for APT detection. First, we detect APT attacks in their early stages, allowing us to disrupt such attacks before further damage is caused. This is done by building an hybrid model that combines seamlessly three different machine learning algorithms. These are inverse PCA, calibration with Platt's method, which is a relatively new approach to obtain posterior probabilities, and Bayesian Network, which is widely used nowadays because of its ability to describe interrelationships from a collection of random variables.
Second, we provide an effective way to predict the steps in the kill chain of an APT attack through multi-class classification. Accurate determination of ongoing APT stage allows selecting and deploying systematically the most adequate response against such attack while it is still unfolding. Furthermore, such information can help the post-attack forensic investigation through more accurate reconstruction of the attack scenario and better understanding of the corresponding tactics, techniques, and procedures (TTPs).

1.4 Thesis outline

The remainder of this thesis is structured as follows.

- An overview of APT attacks and related work on detecting them is presented in chapter 2.
- Chapter 3 presents our proposed detection approach, including its design and models.
- Chapter 4 presents the experimental evaluation of the proposed approach and the discussion of our results.
- chapter 5 gives concluding remarks and outlines future work to address the limitations of our model.

Chapter 2

Related Work

2.1 Background on APTs

The term APT was coined by the U.S. Air Force around 2006 as a general, unclassified, moniker used to describe state-sponsored organizations capable of perpetrating attacks against military, governmental or other sensitive and strategic assets [2]. In [2], Bejtlich described the words comprising the term APT as follows:

- **Advanced** means the adversary is highly skilled and can adapt to counter a target's defences by crafting custom exploits and utilizing zero-day vulnerabilities when necessary.
- **Persistent** means the adversary is not an opportunistic intruder, but instead has a specific objective and is therefore willing to engage in a long-term campaign and will work to maintain presence as long as required to accomplish its objective.
- **Threat** means the adversary is organized, well-funded and motivated, making them a real threat to even well-defended assets.

Over time the term has evolved from being a proper noun used for specific, known, actors to one generically describing an entire class of attacks that have the same characteristics. It also acquired a broader scope as it began to be used in the private domain, most commonly for intellectual property theft [4].

Some authors distinguish between the broader "targeted attacks" and the nation-state specific APT [29], while others use terms like advanced persistent attack (ATA)

[20] but only as an equivalent of APT. In practice, the APT moniker is applied regardless of the target and the actors responsible for the attack, as long as, according to the National Institute of Standards and Technology (NIST) definition, the attacker: "(i) pursues its objectives repeatedly over an extended period of time; (ii) adapts to defenders' efforts to resist it; and (iii) is determined to maintain the level of interaction needed to execute its objectives".

Despite being highly variable and adaptive, APTs follow a similar life cycle [12, 9, 21, 29, 4, 34]. Different authors have proposed various numbers of phases, but they mostly converge in a general life cycle that includes the following phases:

- *Reconnaissance*: This is the starting point of any attack. In this stage the attackers gather the maximum amount of intelligence possible about the target, its resources, personnel and relationships so they can identify possible points of weakness that can be leveraged.

- *Weaponization*: After identifying possible attack vectors, the attackers begin preparing and testing tools for the attack. Depending on the case, the attackers might use publicly available malware, purchase zero-day vulnerabilities from the dark web, or in the most advanced cases, build their own tools. The attacker then tests the tools for detection using replicas of the target environment and assesses their risk of detection and retaliation.

- *Delivery*: In this phase, the attackers deliver their exploits to the target. The two most common delivery methods are spear-phishing and watering-hole attacks [10]. For web-based attacks to be successful, they need to exploit a vulnerability in an application, usually a PDF reader, browser or Microsoft Office suite program. For air-gapped networks, USB sticks and other forms of physical media may be infected and used as delivery. When possible, bribed or disgruntled employees or double-agents may be used as well.

- *Foothold Establishment*: Once the malware is delivered and installed, it tries to take control of the infected system and initiate command and control (C&C) communication using remote access trojans (RATs), rootkits and other means. In [12], Hutchins et al. split this phase into *exploitation* and *installation*.

- *Progression*: Once inside, the attackers start working towards expanding their control over the network. Having access to a single computer is a precarious

position because that machine might be moved, shut-down or wiped clean, and besides, the infected machine might not be the final target of the attack. With that in mind, the intruders try to move laterally to other machines so they can open extra doors inside and gain access to more important resources. This phase works in a cycle of internal reconnaissance, lateral movement and access escalation, with some authors, such as [21, 4], similarly splitting this phase.

- *Mission Realization*: Once the attackers reach their target, they can start fulfilling their mission. That typically involves data exfiltration through their open control channels, but it might also involve sabotage, as in the case of Stuxnet.

- *Cover tracks*: Finally, to avoid detection and attribution, the attackers try to cover all their tracks by deleting logs and any unnecessary file and malware they have installed. In practice, this phase is continuously happening throughout the whole campaign so that the attackers can remain undetected for as long as necessary.

2.1.1 Example of Real-World APT Scenario: the ToddyCat APT Incident

A new APT group called ToddyCat has been targeting Microsoft Exchange servers in Europe and Asia with a focus on government and diplomatic entities. The group has been in action since December 2020. Exploitation of vulnerable servers has led to the creation of a web shell called China Chopper that is used to initiate a multi-step infection. According to Kapersky, the group uses two of the most popular cyber espionage tools, including a new trojan malware called Ninja and a previously unknown passive backdoor called Samurai. Both of these malware allow the attacker to take control of infected systems and move around the victim's networks. Figure 2.1 summarizes the steps involved in a typical ToddyCat APT attack.

Started in February 2021, the new APT attack began exploiting the proxy logon vulnerability on Microsoft Exchange to gain initial access. This followed with deploying the Samurai backdoor on the vulnerable Exchange servers. Samurai was used in the second wave of attacks against Microsoft Exchange servers. During this period, ToddyCat expanded to include desktop systems in addition to Microsoft Exchange

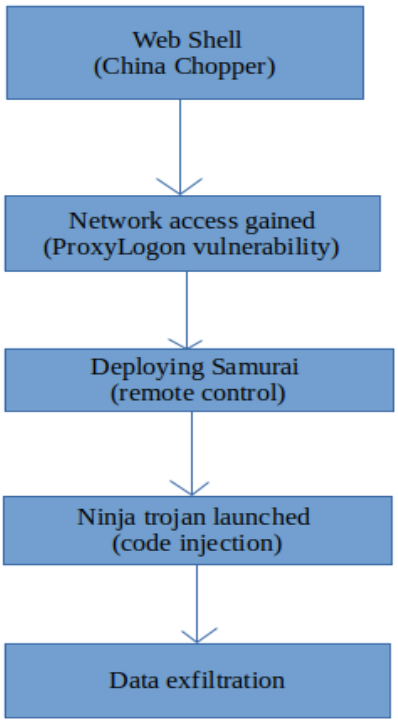

Figure 2.1: ToddyCat APT anatomy

servers. While using obfuscation to avoid detection, the backdoor communicates on ports 80 as well as 443 and allows code execution. Then, the Ninja Trojan was launched, allowing several attackers to simultaneously take control of the same machine and communicate over multiple protocols. Ninja is a collaborative Trojan tool allowing network penetration via a set of commands. In order to avoid behavior-based detection, the threat uses a specific time slot to operate.
ToddyCat is a persistent and sophisticated APT that remains stealthy to compromise many organizations. The APT's target are high-profile organizations and it has not been linked or attributed to any known actor. Unlike ransomware groups, ToddyCat APT campaigns are not out to make money as their primary goal is to collect data, which can be used for political purposes.

2.2 Related Work on APT detection

The characteristics of APTs make identifying them a complex task. Many countermeasures exist but are not always enforced or effective [31].

Based on the APT life stages, Hutchins et al. [12] proposed the *intrusion kill chain* model to guide defenders in developing mitigation against APTs. It works by creating a "courses of action" matrix with incrementally stronger mitigation designed to detect, deny, disrupt, degrade and deceive the intruders at each chain stage. The premise behind the model is that adversaries must progress through each stage of the chain (life cycle) before they can reach their objectives. Therefore, blocking a single link will thwart the attack in course and force the attackers to evolve. Then, armed with the knowledge gained from the thwarted attacks, the defenders can learn about the attackers and how they operated to also evolve.

The intrusion kill chain model's strength has been demonstrated by several authors, including Ioannou et al. [13] and Bhatt et al. [3], who used it as a conceptual framework for APT modelling and detection.

Another modelling framework based on the same life cycle principle is the *attack pyramid* proposed by Giura and Wang [9], which extends the attack tree model [28] by adding a third dimension, which allows for the modelling of a multi-stage attack towards a common goal at the top of the pyramid.

Luh et al. [20] conducted an extensive review of works in APT that provided several other interesting findings. For instance, it shows an almost even split between malware detection, host intrusions and network intrusions detection. That is explained by the multi-modal nature of APT attacks, which require a varied set of equally important tools, each covering different intrusion stages. Consider for instance the delivery stage, which requires host-based tools to prevent or at least detect infection, and the operations stage, which requires tools capable of identifying unauthorized network movement and C&C communication.

The paper also shows that the majority of the studies have focused on a single area of detection, be it host-based or network-based, with only a few utilizing multiple sources of data. That demonstrates a gap in the literature given what was discussed in the previous paragraph and the importance of linking data from multiple sources to identify hidden patterns of compromise among them. Nevertheless, many novel APT detection and prevention techniques have been proposed or developed in the past few years.

Giura and Wang [9] proposed a new anomaly-based detection framework, conceptualized on the attack pyramid model created by the same authors. The system uses data from several different network and application logs to categorize intrusion events into the pyramid's planes (life cycle phases) and correlate events across the planes to

detect APTs.

A similar proposition, but conceptually based on the intrusion kill chain model, was demonstrated in Ioannou et al. [13] and Bhatt et al. [3]. However, Ioannou et al. [13] used Markov belief models to manage conflict within the data and then try to predict the intruder's next steps, while the Bhatt et al. [3] model used a rule-based approach and had no predictive capabilities.

Focused on detecting the initial intrusion, Moon et al. [23] used decision tree based behavioural analysis to detect host infections. Their model employed mostly host-based features, but included some network flow features as well.

Levchuk et al. [16] introduced a model that utilized NetFlow data to generate graphs of host network behaviour and then employed graphical pattern matching techniques to match them against attack fingerprints and identify deviations from known behaviour.

Han and Weijie [11] proposed an APT detection model that utilizes system call information and ontological knowledge. The ontology model provided the knowledge base capturing APT malicious behaviours. The classification of APT malware was based on the dynamic API call sequence which is used to extract a feature vector. The correlation between the API sequence and the ontology knowledge base was used to evaluate the effectiveness of the model. The model achieved an accuracy between 98% and 99.3% on the VirusShare dataset.

Xuan and Dao [7] proposed an approach for APT detection combining different deep learning models, including multi-layer perceptron (MLP), convolution neural network (CNN), and long short-term memory (LSTM). The proposed method was based on three majors steps. First, network flows were extracted. Next, the combined models of CNN with LSTM and CNN with MLP performed IP feature extraction from the extracted network flows. In the last step, IPs were classified as either legitimate or malicious. The data used for the experiment were collected from 29 network traffic files involving 1,065 IPs, of which 139 were related to APT attacks and 926 were legitimate IPs. The model used 70%, 10% and 20% of the data to train, validate and test the model, respectively. The combined models achieved accuracy values between 93% and 98%, which was much higher than the results obtained by the individual deep learning models.

Neuschmied and Helmut [25] proposed a model for APT detection combining One-class support vector machine (OCSVM) and several auto encoders, including standard auto encoder (AE), AE-CNN, variational AE, variational AE using reconstruction

probability (VAE-prob), and AE with labelled data (AEC). The dataset used for the experiment was acquired by combining two publicly available benchmark datasets, namely, Contagio and CICIDS2017. The former contains only APT data while the later includes both attack and benign data. The evaluation of the model on the dataset achieved an accuracy value between 58% and 66%.

Dehghan and Motahareh [5] proposed an approach for APT detection based on deep reinforcement learning models. The approach involved use the Q-learning and LSTM algorithms to approximate the best action of each stage, where an action represents the next step of attacks. They defined the problem of APT prediction similarly to a Markov decision process that selects the best action using the ϵ - greedy policy. The proposed method is able to employ datasets or interact with the environment. The evaluation of the model using the DAPT2020 data set achieved an accuracy and a precision of 97.36% and 93.52%, respectively.

Park and Na-Eun [27] proposed an approach for fast and efficient APT detection using open source tools, including Google Rapid Response (GRR) and auditbeat, which improve the endpoint detection and response capability. GRR is based on a client-server model and collects forensic data while Auditbeat is a log collection tool. The approach uses a threshold model for attack detection and a ruleset, which can be composed of a single step or a combination of several attack steps. The optimal ruleset was derived based on the highest detection rate. The experimental evaluation of the model was conducted using a custom dataset that was collected in an environment where GRR and Auditbeat were deployed, and achieved a detection rate of about 80% based on the assumption that there were no false alarms (i.e. FPR=0%). passport, min visitor 6mois Kumar and Ayush [15] proposed an approach for APT detection in industrial IoT called RAPTOR. The approach involved identifying and correlating several APT attack stages to generate a high-level APT campaign graph. The proposed system used data from different sources such as network traffic traces, audit logs, and HIDS/NIDS alerts. The performance of the proposed model was evaluated for detecting the discovery and fieldbus scanning stages, and achieved precision and recall values around 99%.

Milajerdi et al. [22] proposed an approach to APT detection called HOLMES by identifying correlations of suspicious information flows. The approach involved generating a graph that summarizes an ongoing attack across many hosts and steps. The HOLMES model used audit data to perform signal detection. The first stage of the approach consisted of data collection and representation, where data were represented

as a provenance graph. The next stage was the specification of the tactics, techniques and procedures, followed by the construction of a high-level scenario graph. Finally, signal correlation and detection were performed. The evaluation was conducted using a dataset based on red team versus blue team simulations involving nine APT scenarios generated by the DARPA Transparent Computing Program. With threshold variations, the precision, recall and F-score values were relatively high, meaning that HOLMES successfully detected malicious behaviours [22].

Ghafir et al. [8] proposed an APT detection system that combined threat detection, correlation and prediction. They developed separate modules for detecting individual threats that trigger alerts related to specific APT stages, and the generated alerts were correlated by linking them to specific APT attack scenario. Based on historical traces of the correlation model output, a machine learning model was trained and used to predict APT attacks. Different classification techniques were studied including decision tree, support vector machine (SVM), k-nearest neighbor (k-NN) and ensemble learning. Prediction accuracy ranging from 68% and 84.8% were obtained, with SVM yielding the best accuracy result of 84.8%. The evaluation of the proposed approach focused on APT scenarios as a whole (i.e. all steps together) rather than identification of a specific stage. In contrast, our work was evaluated by determining its effectiveness at identifying separate stages of the attack.

Ahmed et al. [36] proposed a machine learning model for APT detection based on the cyber kill chain stages. They relabelled the dataset used by Ghafir et al. [8] by clarifying the attack stage information and used it to design and evaluate their model. The model extracts three types of features from an APT dataset based on information gain, gain ratio and oneR feature selection techniques. They explored different classifiers, including naive Bayes, SVM, random forest, k-NN and decision trees. Although their aim was to detect individual stages, they computed global performance measures that do not consider the accuracy in detecting individual stages. For instance, the detection rate was computed as the ratio between the number of actual stages detected and the total number of stages detected in the dataset. This means a detected stage was counted as a true positive regardless of whether it was an exact match or not. In contrast, our work provided finer grained performance measures about the accuracy of predictions for individual stages.

Xuan [6] proposed an approach for detecting APT by extracting suspicious domains and IPs from network traffic and applying machine learning classification using the random forest algorithm. The authors identified five new features from the ab-

normal domain behaviours, namely, the number of domains sharing the same IP, the IP addresses in the same class B, the daily similarity, the same query numbers in the same time window and the very low-frequency query. They also extracted five new features from the abnormal IP behaviours, including protocols and ports, TCP connections, data fluctuation abnormalities, the heartbeat traffic and discrepancies between the transferred and received data. The correlation between IP and domain was exploited to increase the efficiency of the proposed model. The dataset used for the experiment consists of 164,077 domains, of which 79,910 were benign samples and 84,167 is the number of malicious domains. In addition, 985,595 DNS queries were used to detect malicious IPs. The evaluation yielded accuracy rates of 88.3% for domain detection and 80.9% for IP detection.

Niu et al. [26] proposed an approach to the detection of APT malware C&C domains by extracting various features from DNS logs and classifying them using the global abnormal forest (GAF) model. The evaluation of the proposed model yielded an accuracy rate of 98.3%, a false positive rate of 0.013%, and a false negative rate of 0.004%, with a detection speed of 18.7 seconds. The authors also implemented traditional models for comparative studies, including k-NN, isolation forest (iForest) and local outlier factor (LOF). GAF was the best model overall in terms of accuracy and calculation efficiency. Zhao et al. [37] proposed a novel method to efficiently and accurately detect APT malware using traffic analysis and malicious DNS, based on the fact that DNS is one of the best tools for detecting C&C servers. The model used 14 features extracted from suspicious DNS grouped in five categories: domain name based, DNS answer based, time value based, TTL value based, and probing features [37]. The extracted features were processed by three detectors, including a detector for suspicious DNS using the J48 decision tree algorithm, a signature-based detector and an anomaly detector for APT malware behaviours. The outputs of the detectors were fed into a statistics-based reputation model, which computed a reputation score that was used to ultimately decide whether there was an APT malware infection. The experimental evaluation conducted using a dataset containing about 400 million DNS queries yielded a detection rate of about 96% and a false positive rate of 1.5%.

2.3 Summary

In this chapter, we reviewed the APT life cycle and examined existing research in detecting APT activities. We discussed the strength and limitations of existing approaches and contrasted them with our proposed method. In the next chapter, we present our approach in more detail.

Chapter 3

Models and Approach

3.1 Approach Overview

In this chapter, we summarize our proposed approach and revisit the different steps in more details.

Figure 3.1 depicts the main steps involved in our proposed approach. The approach starts by computing an anomaly score for an incoming event using *inverse principal component analysis (PCA)*. Next, the score is converted into a probability value using a new score calibration technique introduced in [35]. The probability value is inputted to a trained BN classifier and used to infer whether the corresponding event is benign or matches one of the APT kill chain stages.

3.2 Principal Component Analysis

While the primary use of PCA is reducing the dimensions of a feature space, in this study, we used the inverse model to compute a score or measure of anomalousness of a security event based on the corresponding reconstruction error. The idea underlying the use of the reconstruction error as an anomaly score is that a data model reconstructed from low dimension representations captures the essence of the data without extraneous features and noise. As such, the deviation (or error) between the original data and the reconstructed data corresponds to spurious data or anomalies compounded by the reconstruction error. A key difference between our work and previous studies using PCA-based reconstruction errors for anomaly detection [30] is that we convert the score into a probability measure, which relies on a more rigorous,

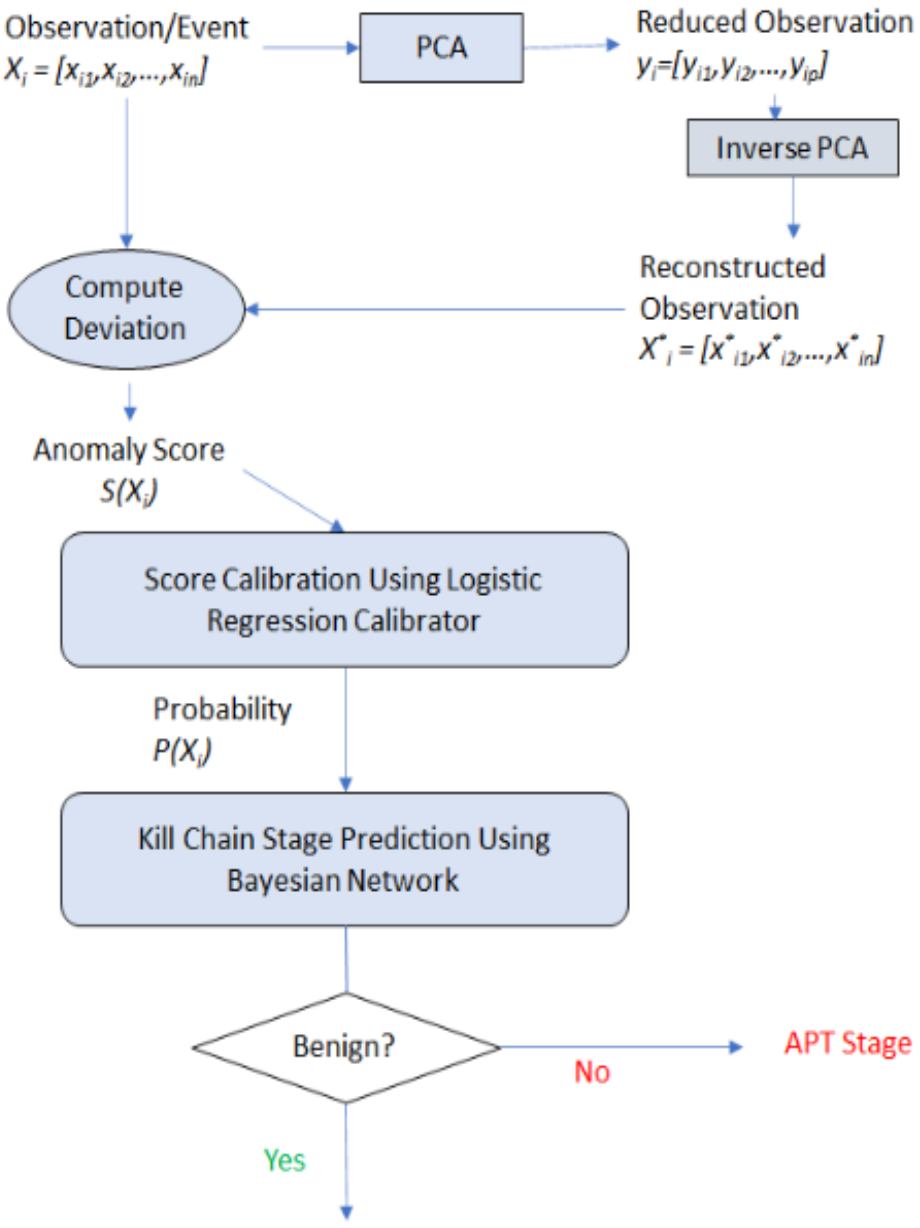

Figure 3.1: APT Detection and Stage Mapping Process

objective and sound theoretical foundation.

To compute the score, we apply PCA to the sample data and reconstruct the data using PCA inverse transformation [19]. The score is the reconstructed data's deviation from the original data.

Consider an observation or event x_i represented in an n-dimensional feature space as $x_i = [x_{ij}]_{1\leq j\leq n}$, where x_{ij} denotes the j^{th} feature of x_i. Given a training set consisting of d observations, let A denote the corresponding data matrix: $A = [x_{ij}]_{1\leq i\leq d, 1\leq j\leq n} \in \mathbb{R}^{d\times n}$.

We apply PCA to the data as follows: data standardization, computation of the covariance matrix, singular value decomposition (SVD), and projection in a lower dimensional space.

Given the data matrix A, we obtain the normalized data matrix $\tilde{A}$ with

$$\tilde{A}[i,j] = \frac{A[i,j] - \mu_i}{\sigma_i},$$

where $1 \leq i \leq d$ and $1 \leq j \leq n$. The parameter μ_i is the mean over x_i defined by

$$\mu_i = \frac{1}{n}\sum_{1\leq j\leq n} x_{ij},$$

and σ_i is the standard deviation given by:

$$\sigma_i = \frac{1}{n}\sqrt{\sum_{1\leq j\leq n}(x_{ij}^2 - \mu_i^2)}.$$

The main advantage of working with the normalised data $\tilde{A}$ rather than A is the fact that it removes bias by projecting each $\tilde{x}_{ij} = \tilde{A}[i,j]$ within 0 and 1. Having the normalized data matrix $\tilde{A}$, we compute the covariance matrix $\Sigma \in \mathbb{R}^{n\times n}$

$$\Sigma = \tilde{A}^T\tilde{A}, \tag{3.1}$$

which is then used to estimate the p first eigen values $\{\lambda_j\}_{1\leq j\leq p}$ with the associated eigenvectors $V = \big(V_j\big)_{1\leq j\leq p} \in \mathbb{R}^{n\times p}$ and $U = \big(U_j\big)_{1\leq j\leq p} \in \mathbb{R}^{n\times p}$. This gives

$$\Sigma = VDU^T, \tag{3.2}$$

where $D \in \mathbb{R}^{p\times p}$ is a diagonal matrix with $\big(\lambda_j\big)_{1\leq j\leq p}$ entries.

Given an observation (or data vector) $x_i = [x_{ij}]_{1 \leq j \leq n}$, the application of PCA will result in a corresponding representation in the reduced space $y_i = [y_{ij}]_{1 \leq j \leq p}$, where p is the size of the reduced space, and $p \leq n$. The new representation is calculated as $y_i = U^T x_i$.

Applying the inverse PCA to y_i allows for determining an approximation of the corresponding high-dimensional measurement computed as $x_i^* = U y_i$.

3.3 Anomaly Score Calculation

The anomaly scores are computed by taking the sum of the square difference between the original and the reconstructed data. The more significant the difference, the higher the probability of being an anomaly. This means that a significant deviation or sudden change in behavior implies that the activity is suspicious. Then, this sum of differences is scaled using a min-max scaler so that the value of the score is between 0 and 1.

The anomaly score for observation x_i is defined as follows:

$$score(x_i) = \sum_{j=1}^{n} (x_{ij} - x_{ij}^*)^2 \tag{3.3}$$

Equation 3.4 is used to scale those scores:

$$\tilde{score}(x_i) = \frac{score(x_i) - score_{min}}{score_{max} - score_{min}} \tag{3.4}$$

where $score_{max}$ and $score_{min}$ are the maximum and minimum values of the scores obtained from the training data. For more rigorous decision-making, we convert the obtained scores into probability scores using calibration.

3.4 Calibration technique

Calibration is the process of converting a classifier score into a posterior probability (PP)[35]. Depending on the task, either single-score or multi-score calibration can be performed. The difference between the two approaches is that the latter uses several classifiers to output a score for a single observation, while the former does not. There are several calibration methods, including logistic regression, isotonic regression, binning and PROPROC methods, where the modelling approach includes

the posterior probability, the likelihood, and the likelihood ratio. Logistic regression is suitable for multi-score calibration because it targets discrete variables and does not care about the size of the feature vector. In addition, it assess what variables are useful to make the classification.

The PP for an observation x_i can be calculated as follows:

$$P(class|\tilde{score}(x_i)) = Likelihood(\tilde{score}(x_i)|class) \times prior(class) \tag{3.5}$$

where, given a particular score, PP is a conditional probability for finding the estimated *class*. The likelihood and prior are similarly defined. The prior distribution is the uncertainty and may come from various distributions, including the normal distribution or the Bernoulli distribution. Equation 3.5 can be solved using a convex optimization problem.

$$\hat{class} = \arg\max_{class} P(class|\tilde{score}(x_i)) \tag{3.6}$$

$$= \arg\max_{class} \log P(class|\tilde{score}(x_i)) \tag{3.7}$$

where, given a score, the estimate of the correct class, $\hat{class}$ has the maximum posterior probability. Because the logarithm function is an increasing function that will avoid underflow and increase speed, solving equation 3.6 is equivalent to solving equation 3.7.

The performance of the calibrator was assessed with two measures, one of which is the mean square error (MSE) that is the square difference between the actual posterior probability and its calibrated value. The second measure is the Brier score, which is the square difference between the actual class label of a sample and its calibrated score.

3.5 Bayesian Network

Bayesian classifiers are statistical classifiers that build a probabilistic model and use it to predict the class of any new sample. There are several variants of Bayesian classifiers, one of the most prominent being naïve Bayes, which is based on a supervised learning model. Naïve Bayes is a form of Bayesian Network where the attributes of the sample are discrete in nature and conditionally independent. It is a generative model widely applied in various fields, including spam filtering , document classification, sen-

timent recognition, and image classification. The classifier uses Bayes theorem while assuming mutual independence of the features. That is, the covariance matrices are diagonal with class-specific matrices. The difference between various naive Bayes classifiers is related to the assumptions about the distribution of $P(s\tilde{c}ore(x_i)|class)$. Bayesian network is a probabilistic graphical model where nodes are discrete or continuous variable and links represent relationship between variables. It is used for model inputs having high sensitivity and uncertainty and aims to update uncertain parameters to fit the observed data. Therefore, Bayesian Network builds models by considering the relationships between variables. Bayesian network is represented by a direct acyclic graph (DAG), which models uncertainty based on the conditional or joint probability distribution of each random variable. In our work, we used the Gaussian naive Bayes classifier, which assumes that the likelihood of the features are normally distributed according to the following formula:

$$P(s\tilde{c}ore(x_i)|class) = \frac{1}{\sqrt{2\pi\sigma^2_{s\tilde{c}ore(x_i)}}} exp(-\frac{(class - \mu_{s\tilde{c}ore(x_i)})^2}{2\sigma^2_{s\tilde{c}ore(x_i)}}) \tag{3.8}$$

where the mean $\mu_{s\tilde{c}ore(x_i)}$ and standard deviation $\sigma_{s\tilde{c}ore(x_i)}$ are estimated using the maximum likelihood.

Figure 3.2 presents an example of a Bayesian Network. In this graph, six events can determine the label of an observation: the prior assumption made on classes, the activity that influences the stage of the record, the score which depends on the mean and variance of the data. For example, the activity 'Normal' associated with the Benign stage will be assigned a label of 0 while the activity 'Directory Bruteforce' associated with the Establish Foothold stage will be assigned a label of 4. We assume that the priors are adjusted according to the data. Depending on the values of the mean, variance and score, which are real numbers, we predict the label of the corresponding observation.

3.6 Attack Classification and Stage Prediction

Given an observation, we find the class with multi-class Bayesian network classification. Bayesian network is a probabilistic graphical model where nodes are discrete or continuous variables and links represent relationships between variables. It is used for model inputs that have high sensitivity and uncertainty and aims to update un-

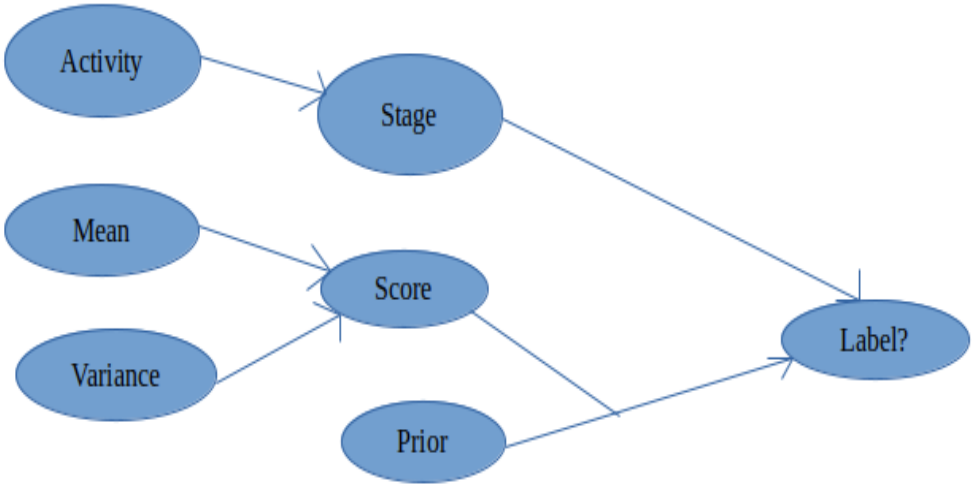

Figure 3.2: Example of Bayesian Network graph

certain parameters to fit the observed data. Bayesian network is represented by a direct acyclic graph (DAG), which models uncertainty based on the conditional probability of each random variable. Using Bayes's rule to transform equation 3.7 into probabilities with useful properties, we defined the conditional probability as follows:

$$p(class|s\tilde{c}ore(x_i)) = \frac{P(s\tilde{c}ore(x_i)|class) * P(class)}{P(s\tilde{c}ore(x_i))} \tag{3.9}$$

Because $P(s\tilde{c}ore(x_i))$ is not dependent on classes, it can be treated as a constant, and equation 3.9 can be rewritten as follows:

$$P(class|s\tilde{c}ore(x_i)) \propto P(s\tilde{c}ore(x_i)|class) \times P(class) \tag{3.10}$$

meaning that Eq. 3.10 is proportional to Eq. 3.9; $\propto$ denotes proportionality between the terms on both sides.
By substituting Eq. 3.10 into 3.7, we obtain Eq. 3.12:

$$\hat{class} = \underset{class}{arg\max} \log P(class|s\tilde{c}ore(x_i)) \tag{3.11}$$

$$= \underset{class}{arg\max} \log P(s\tilde{c}ore(x_i)|class) \times p(class) \tag{3.12}$$

where $P(class)$ is the prior and $p(s\tilde{c}ore(x_i))$ is the marginal probability of score.

We used the library *bnlearn*, which stands for Bayesian network learning, to train the model and for predictions. The library includes structure learning, which de-

scribes the relationship in the data, and parameter learning, which learns about the conditional probability. The package involves learning the structure through a DAG search algorithm, including chickening and k2 algorithms, while parameters are learned though the maximum likelihood estimator.

Given an observation, the classifier is used for basic attack classification when applicable data are available (i.e. benign vs. attack samples) and also for attack stage prediction.

Let $L = \{L_r \mid 0 \leq r \leq m\}$ denote a set of stages, where the first stage L_0 corresponds to benign or legitimate activities and the remaining stages ($1 \leq r \leq m$) are the kill-chain attack stages. Intuitively, the higher the cyber kill chain stage, the more dangerous the activity, and thereby, the greater the anomaly score. Based on this intuition, each stage L_r can be defined in terms of a probability interval (i.e. calibrated score interval) $[p(L_r)_{min}, p(L_r)_{max}]$. This corresponds to a confidence interval where an observation x_i is likely at stage L_r if $p(L_r)_{min} \leq \tilde{score}(x_i) \leq p(L_r)_{max}$. By treating each stage as a separate class, we predict the stage for a given observation x_i using multi-class classification by finding the class with the largest probability (i.e. calibrated score), as follows:

$$\hat{class} = \underset{L_r \in L}{arg\max} \log P(L_r | \tilde{score}(x_i)) \tag{3.13}$$

3.7 Summary

This chapter presented the general approach for our detection model. We described the process for classification and stage prediction of the attack. In the next chapter, we present the experimental evaluation and discuss the obtained results.

Chapter 4

Experiments and Evaluation

In this chapter, we present in details the experimental evaluation of our proposed approach. We outline the experiment procedure, the evaluation metrics and, present and discuss the evaluation results.

4.1 Experiment outline

To evaluate our model, we ran experiments using two public benchmark datasets for APT: the DAPT2020 dataset proposed by Myneni et al. [24] and the dataset of meta-alerts for APT detection released by Ladani et al. [1]. The first dataset allows for assessing our model's effectiveness at APT attack detection and stage prediction. The second dataset was used only for assessing stage prediction because it contains only attack data represented by the meta-alerts. We present and discuss the evaluation results for each of these datasets in the following.

4.2 Performance metrics

To evaluate our approach, for each of the datasets, we generated a confusion matrix and used it for computing different performance metrics, including precision, recall, F1 score, and accuracy.

A low number of false positives is crucial to avoid (false) alarm fatigue, whereby the high number of false alarms can lead to security officers ignoring most alerts, whether genuine or not. The lower the number of false positives, the higher the precision. High recall is desirable because this means a greater number of attacks are

detected.

Accuracy provides a combined measure of the ability of the system to correctly categorize observations, both benign and attack samples. Accuracy is a good indicator of performance when false positives and false negatives have the same cost or with a balanced dataset. In contrast, when the cost of false positives and false negatives are very different, F1 score is a better indicator. F1 score provides a balance between precision and recall when dealing with imbalanced datasets, which is the case for most intrusion detection datasets. Additionally, for an imbalanced dataset, balanced accuracy can be computed because it is insensitive to imbalanced class distribution.

We computed the metrics per class by determining from the confusion matrix the number of false positives and false negatives for each class. Given class $Class_i$, the number of true positives TP_i are the elements in the diagonal cell (i, i). Assuming that the rows and columns of the matrix correspond to the actual and predicted classifications, respectively, the numbers of false positives FP_i and false negatives FN_i correspond to the incorrectly classified samples in the class's column and row, respectively. The number of true negatives TN_i is the sum of all the remaining cells of the matrix.

The per-class metrics are computed as:

$$Precision_i = \frac{TP_i}{TP_i + FP_i} \tag{4.1}$$

$$Recall_i = \frac{TP_i}{TP_i + FN_i} \tag{4.2}$$

$$F1 - score_i = \frac{2 \times Precision_i \times Recall_i}{Precision_i + Recall_i} \tag{4.3}$$

$$Accuracy = \frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i} \tag{4.4}$$

The macro average of each of the metrics is obtained by computing the arithmetic mean of all the per-class values of the corresponding metrics. Macro averages are suited for datasets involving imbalanced class distributions, where classes are often equally important. Furthermore, to account for the contribution or weight of individual classes, weighted averages are provided by computing the mean of all per-class values, while considering the number of actual occurrences of each class in the dataset. Depending on the size of the data, there are several ways to perform data splitting. Less training data influence the estimated parameters with high variance and no sig-

nificant cross-validation results. It is recommended to split the data so that neither the variance nor the bias is too high. With a large amount of data, there is not much difference between the 80:20, 70:30 or 90:10 split. But the right starting point is 80:20, unless the method is computationally intensive, which requires less training data [32]. We trained three classifiers with different amounts of training data using the above split and recorded the performance of each classifier on the validation data. We noticed that the classifier having the best performance was the one built with the 80:20 split. We therefore decided to present the experiment focusing on the model that uses 80% of the data to train the classifier and the remaining 20% for testing.

4.3 Evaluation using the DAPT Dataset

4.3.1 Dataset overview

DAPT 2020 [24] consists of simulated APT behavior in a cloud network. It involves public network traffic data collected over five days, where day 1 corresponds to legitimate activities, and the remaining days (day 2 to day 5) correspond to APT attack vectors, including data exfiltration, network scan, account discovery, directory brute force, web vulnerability scan, account brute force, SQL injection, CSRF, malware download, backdoor and command injection. Each day of the week covers data for private and public networks compiled in 10 CSV and pcap files. The dataset consists of 85 features with the columns activity and stage used for labelling. It contains benign data and data involving four main APT stages including reconnaissance, foothold establishment, data exfiltration, and lateral movement. It contains 17,339 samples spread unevenly between the classes, so it is highly imbalanced.

The DAPT2020 data consists of 85 features; a subset of these features is presented in table 4.1.

4.3.2 Evaluation results

We started by concatenating the ten CSV files to obtain an entire dataset. In the pre-processing stage, we found no missing or duplicated values. We divided the data into training and test sets, with 80% of the data used to train the classifier and the remaining 20% used to test the model. We used single-score calibration. The advantage of this calibration technique is that the original and the estimated performance

DAPT Features	**Description**
Src Port	Source Port
Dst Port	Destination Port
Protocol	Communication Protocol
Flow Duration	Flow duration
Total Fwd Packet	Total number of packets in the forward direction
Total Bwd Packet	Total number of packets in the backward direction
Total Length of Fwd Packet	Total size of packets in forward direction
Total Length of Bwd Packet	Total size of packets in backward direction
Fwd Packet Length Max	Maximum size of packets in forward direction
Fwd Packet Length Min	Minimum size of packets in forward direction
Fwd Packet Length Mean	Average size of packets in forward direction
Fwd Packet Length Std	Standard deviation of packet size in forward direction
Fwd Act Data Pkts	Number of packets with at least 1 byte of TCP data payload in the forward direction
Fwd Seg Size Min	Minimum segment size observed in the forward direction
Active Mean	Mean time a flow was active before becoming idle
Active Std	Standard deviation of time a flow was active before becoming idle
Active Max	Maximum time a flow was active before becoming idle
Active Min	Minimum time a flow was active before becoming idle
Idle Mean	Mean time a flow was idle before becoming active
Idle Std	Standard deviation of time a flow was idle before becoming active
Idle Max	Maximum time a flow was idle before becoming active
Idle Min	Minimum time a flow was idle before becoming active

Table 4.1: Examples of features from the DAPT2020 dataset.

remain the same after calibration. Using two different calibrators, the logistic regression calibrator with and without Platt's method, we generated calibrated scores. As expected, the version invented by John Platt in the SVM context outperformed logistic regression due to its ability to reduce bias and obtain the probability distribution over classes by converting the outputs of a non-probabilistic classification model. Figure 4.1 shows sample observations from the dataset, with scores before and after calibration using Platt's model.

	Uncalibrated_score	Calibrated_score
0	1.859845e-06	0.990155
1	4.373830e-07	0.016706
2	1.695222e-06	0.973548
3	1.181179e-06	0.614608
4	2.678891e-07	0.005999
...	...	...
17334	2.475337e-07	0.005302
17335	2.545811e-06	0.999849
17336	3.609755e-07	0.010543
17337	2.425718e-07	0.005144
17338	2.815024e-07	0.006516

Figure 4.1: Scores before and after calibration with Platt's method for sample observations from the DAPT2020 dataset

Figure 4.2 shows sample observations with the calibrated scores and the corresponding stages from the dataset. As expected, as we moved upward in stages, the probability increased as well.

To predict the stages, we trained a multi-class classifier by considering the *Stage* feature as classes. The dataset contains five stages — including benign, reconnaissance, establish foothold, lateral movement and data exfiltration — which are labeled 0 through 4.

By training and testing the classifier, we obtained the confusion matrix shown in Figure 4.3.2.

Table 4.3 provides a breakdown per class of the performance metrics computed from the matrix. Table 4.4 provides global performance measures obtained by com-

	Stage	Uncalibrated Score	Calibrated score
0	Establish Foothold	5.358544e-07	0.212021
1	Establish Foothold	4.301933e-07	0.153976
2	Reconnaissance	5.269342e-07	0.206559
3	Establish Foothold	4.845407e-07	0.182030
4	Benign	3.582100e-07	0.122377
...	...	...	...
17334	Benign	3.560850e-07	0.121535
17335	BENIGN	5.875370e-07	0.245724
17336	Benign	4.308048e-07	0.154271
17337	Benign	3.562960e-07	0.121619
17338	Benign	3.577216e-07	0.122183

Figure 4.2: Stages associated with calibrated scores for sample observations from DAPT2020 dataset

puting the macro and weighted averages over the different classes.

4.3.3 Discussion

Our approach achieved high accuracy on the DAPT2020 dataset for all stages, ranging from slightly above 81% for stage 0 (benign) to above 97% for stage 2. However, the recall and F1 scores were uneven for the different stages. While high recall and F1 scores were obtained for stages 0, 1, and 2, the values for stages 3 and 4 were low. One of the main reasons for the low values is the small number of samples in these classes; there are only three samples in class 4, which is insufficient for proper training of the classifier. However, as shown in Table 4.4, the weighted averages, which account for the imbalance, yielded relatively high values, with accuracy, precision, recall and F1 scores of 84.59%, 73.39%, 95.27% and 81.79%, respectively.

Actual Classes \ Predicted Classes	Class0	Class1	Class2	Class3	Class4
Class0	9474	3	0	13	0
Class1	0	1363	12	114	0
Class2	0	150	1522	0	0
Class3	2233	346	193	363	1
Class4	1017	508	19	6	2

Table 4.2: Multi-class confusion matrix based on the DAPT2020 dataset

4.4 Evaluation using the Meta-alerts Dataset

4.4.1 Dataset overview

The semi-real dataset of meta-alerts [1] was synthesized by randomly injecting a set of APT attack meta-alerts into a set from an APT-attack-free meta-alert dataset generated over a 3-day period by a SIEM deployed in a company with 250 hosts. The APT attack meta-alerts were based on traces from real-world APT attack scenarios executed during a 7-day period and involved a variety of attack vectors, such as reflec-

MetricsClass	**0**	**1**	**2**	**3**	**4**
Total samples and ratios	12,724 (73.38%)	2,370 (13.67%)	1,746 (10.07%)	496 (2.86%)	3 (0.02%)
Precision (%)	74.46	57.51	87.17	73.19	66.67
Recall (DR) (%)	99.83	91.54	91.03	11.58	0.13
F1 score (%)	85.30	70.64	89.06	19.99	0.26
Accuracy (%)	81.16	93.47	97.84	83.24	91.05

Table 4.3: Class-based distribution of samples and performance results based on DAPT2020 dataset; the metrics were computed by categorizing the observations from each class

	Accuracy (%)	**Precision** (%)	**Recall** (%)	**F1 score** (%)
Macro average	89.35	71.80	58.82	53.05
Weighted average	84.59	73.39	95.27	81.79

Table 4.4: Average classification results for stage prediction based on the DAPT2020 dataset

Features	Data type
Alert creation time	Time
Alert Impact Severity	String
Source Address	IP address
Source Port	Integer
Target Address	IP address
Target Port	Integer
Intrusion Kill Chain (IKC) step	Integer

Table 4.5: List of the features from the Meta alert data set

tive loading, in-memory module loading and web-shell functionalities. Several APT activities were involved, including drive-by download, backdoor deployment, privilege escalation, reconnaissance, data exfiltration and concealment of attack footprints.

The dataset includes 118,465 samples spread almost evenly (i.e. about 25%) across the stages. It contains 250 text files representing the company's hosts. Each file is a meta-alert record for a host and contains seven features, including the alert creation time, the alert impact severity, the source address, the source port, the target address, the target port and the intrusion kill chain step. The meta-alerts were grouped in four main APT kill chain steps, as follows: reconnaissance and delivery, exploitation, operation and data collection with exfiltration.

The list of meta-alert features along with their data type is presented in table 4.5.

4.4.2 Evaluation results

In addition to the pre-processing step, which was performed in the same manner as the previous dataset, we found text files containing eight features, including the added feature of NaN values. We deleted this column and performed the 80% to 20% data split (for training and testing, respectively) on each file before concatenation. Then, we computed the scores and performed the calibration before applying the Bayesian model.

Figure 4.3 shows the calibration results for sample observations, with scores before and after calibration.

As previously mentioned, because this dataset includes only attack data (i.e. alerts), we used it only to evaluate the stage prediction model using multi-class classification. We trained and tested the multi-class classifier for stage prediction

	Uncalibrated_score	Calibrated_score
0	0.902274	1.000000e+00
1	0.002842	4.377480e-20
2	0.949989	1.000000e+00
3	0.360608	9.798502e-01
4	0.944757	1.000000e+00
...	...	...
52433	0.264691	1.107923e-04
52434	0.001256	3.530984e-20
52435	0.943337	1.000000e+00
52436	0.952207	1.000000e+00
52437	0.269667	2.173649e-04

52438 rows × 2 columns

Figure 4.3: Scores using Platt's method with the meta-alerts dataset

by considering the IKC step feature as classes. There are four values of IKC steps, labeled 1 through 4, where 1 is reconnaissance and delivery, 2 is exploitation, 3 is operation, and 4 is data collection and exfiltration. Because there is no benign data, there is no stage 0.

Table 4.4.2 shows the obtained confusion matrix. We can see that the model misclassified only five samples. Table 4.7 presents the performance metrics computed per class. Table 4.8 shows the global performance measures obtained based on macro averages over all the classes; the weighted averages are omitted because the classes are equally distributed. The approach performed very well across all metrics, with values above 99% in all cases.

4.4.3 Discussion

Compared to the DAPT2020 dataset, our approach achieved much better performance on the meta-alerts dataset across all metrics and for all stages. It achieves high accuracy, recall, precision and F1 scores for all four APT stages. In contrast to DAPT2020, meta-alerts is a balanced dataset, with enough samples in each class to properly train a classifier. This is a possible explanation for the difference in performances between the two datasets. Another possible reason is the fact that the

Actual Classes \ Predicted Classes	Class1	Class2	Class3	Class4
Class1	29773	0	0	0
Class2	0	29608	3	2
Class3	0	0	29593	0
Class4	0	0	0	29486

Table 4.6: Multi-class confusion matrix for stage prediction based on the meta-alert dataset.

two datasets are at different levels of abstraction: raw data versus alerts. This could be an indication that our approach may be better suited for use as an extension for SIEM or in environments where there is a layer of detection systems that generate and feed alerts to it for further classification.

Table 4.9 compares the results obtained in our work with the performance of previous works on APT detection. We compare our method with related works using different datasets because, to our knowledge, only the network traffic data and meta alerts data meet the requirement to inlude stage information. We also performed a comparative study, presented in table 4.10, by applying the K Nearest Neighbours (KNN) and MultiLayer Perception (MLP) algorithms to DAPT2020 and meta alerts data. While our performance with DAPT2020 is at par with most existing works, our results with the meta-alert dataset outperforms them.

MetricsClass	**1**	**2**	**3**	**4**
Total samples and ratios	29,773 (25.13%)	29,608 (24.99%)	29,596 (24.98%)	29,488 (24.90%)
Precision (%)	100	100	99.98	99.99
Recall (DR) (%)	100	99.98	100	100
F1 score (%)	100	99.99	99.99	99.99
Accuracy (%)	100	99.99	99.99	99.99

Table 4.7: Class-based distribution of samples and performance results based on the meta-alerts dataset, the metrics were computed by categorizing the observations from each class.

	Accuracy (%)	**Precision** (%)	**Recall** (%)	**F1 score** (%)
Macro average	99.99	99.99	99.99	99.99

Table 4.8: Average classification results for stage prediction based on the meta-alerts dataset

Reference	**Data set**	**Classifier**	**Accuracy** (%)	**Precision** (%)	**Recall** (%)	**F1-score** (%)
[6]	CTU Botnet and *Mila data*	Random *Forest*	94.62	80.69	85.45	83
[25]	Contagio and CICIDS2017	AE + VAE-Prob	86.9	60.0	88.0	71.3
[7]	CTU-13 data	CNN-LSTM	96.5	97	97	97
[33]	Network Simulator 3	Belief Rule Base	91.14	87.25	93.84	89.87
[18]	Real IOT data:APT29	SMOTE-RF	81	96.8	83.4	89.6
Our work	**DAPT 2020**	**Bayesian Network**	**84.59**	**73.39**	**95.27**	**81.79**
Our work	**Meta-alert Data**	**Bayesian Network**	**99.99**	**99.99**	**99.99**	**99.99**

Table 4.9: Performance comparison with related works.

Data set	**Classifier**	**Accuracy** (%)	**Precision** (%)	**Recall** (%)	**F1-score** (%)
DAPT2020	KNN	98.3	79.3	93.4	85.7
DAPT2020	MLP	96.1	61.5	69.8	65.4
DAPT2020	**Bayesian Network**	**84.59**	**73.39**	**95.27**	**81.79**
Meta alert	KNN	98.8	97.72	97.73	97.7
Meta alert	MLP	98.3	96.55	96.6	96.5
Meta alert	**Bayesian Network**	**99.99**	**99.99**	**99.99**	**99.99**

Table 4.10: Performance comparison with different models on the same dataset.

In table 4.10, we realized that alternative algorithms, including KNN and MLP work well on DAPT2020 data compare to Bayesian Network.

4.5 Summary

In this chapter, we evaluated several metrics against two public datasets. First is the accuracy which is the ratio of correct predictions. This metric describes how the model performs across all classes. Second is the precision, which is the quality of a positive prediction made by the model. Third is the recall that measures the model's ability to detect positive samples. We achieved high accuracy across all datasets, and from above average to excellent for the remaining metrics dependent on the type of dataset.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Due to their long-term nature and stealthiness, APTs can have far-reaching damaging consequences as the attack unravels over time. The further progress achieved by the attack in the kill chain stages, the more serious the corresponding incidents. Therefore, it is crucial to detect not only APT events, but also their corresponding kill chain stages. The goals of APTs fall into four general categories, including Cyber Espionage, theft of intellectual property or state secrets, eCrime for financial gain, Hacktivism and Destruction.

This research aimed to predict the stage of the attack enabling systematic determination and deployment of the most effective defense mechanisms against APTs facing many organizations. To achieve such goal, we have proposed a new approach to detecting APT occurrences and systematically mapping corresponding events to their kill chain stages using calibrated anomaly scoring and machine learning based on a Bayesian network model. The experimental evaluation was conducted on a public network traffic APT dataset and a meta-alert dataset. The evaluation results showed that our proposed model achieved high accuracy across all datasets. However, it performed better on detection accuracy when it was fed alerts rather than raw data. This means that while the proposed approach can be linked to raw intrusion data sources, its best uses is through integration with high alert aggregators such as security information and event management (SIEM) systems.

5.2 Future Work

For future work, a number of improvements within the system could be made. First is to evaluate our approach on other APT datasets. A requirement for our model to be testable is for the datasets to include stage information. To our knowledge, only the two datasets considered in this thesis meet this requirement, but we will be watching for applicable future datasets and consider using them for such further evaluations of our approach.

Furthermore, considering that many intrusion detection datasets are imbalanced, we will develop in our future work a new approach to handle imbalanced dataset using one-class machine learning models. This may involve adding a new layer where one-class machine learning models would be used to handle the imbalanced distribution before applying the proposed multi-class classification approach to detecting the APT events and corresponding stages.

The limitations of using Bayesian network models are various. The first is the assumption of prior knowledge, which plays an important role in evaluating the performance of the model. The prior knowledge must be reliable enough for the model to be of good quality. In other words, the choice of prior knowledge has an impact on the results. Second, it is difficult to represent continuous variables and always get the exact calculation of the posterior probabilities. To improve the abilities of approach on raw data, we will explore other shallow and deep learning classifiers, as alternative to Bayesian network.

Bibliography

[1] Ladai Behrouz Tork and Mehran Khosravi. A semi real dataset of meta-alerts for apt attack detection, 2020.

[2] Richard Bejtlich. What APT is (and what it isn't). *Information Security*, 12(6):20–24, 2010.

[3] Parth Bhatt, Edgar Toshiro Yano, and Per M. Gustavsson. Towards a framework to detect multi-stage advanced persistent threats attacks. In *2014 IEEE 8th International Symposium on Service Oriented System Engineering*, pages 390–395, April 2014.

[4] Ping Chen, Lieven Desmet, and Christophe Huygens. A study on advanced persistent threats. In Bart De Decker and André Zúquete, editors, *Communications and Multimedia Security*, pages 63–72, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[5] Motahareh Dehghan, Babak Sadeghiyan, Erfan Khosravian, Alireza Sedighi Moghaddam, and Farshid Nooshi. Proapt: Projection of apt threats with deep reinforcement learning. *arXiv preprint arXiv:2209.07215*, 2022.

[6] Cho Do Xuan. Detecting apt attacks based on network traffic using machine learning. *Journal of Web Engineering*, pages 171–190, 2021.

[7] Cho Do Xuan and Mai Hoang Dao. A novel approach for apt attack detection based on combined deep learning model. *Neural Computing and Applications*, 33(20):13251–13264, 2021.

[8] Ibrahim Ghafir, M. Hammoudeh, V. Prenosil, L. Han, R. Hegarty, K. Rabie, and F.J. Aparicio-Navarro. Detection of advanced persistent threat using machine-learning correlation analysis. *Future Generation Computer Systems*, 89:349–359, 2018.

[9] Paul Giura and Wei Wang. A context-based detection framework for advanced persistent threats. In *2012 International Conference on Cyber Security*, pages 69–74, December 2012.

[10] Will Gragido. Lions at the Watering Hole – The "VOHO" Affair Speaking of Security – The RSA Blog and Podcast. https://web.archive.org/web/20121212050809/https://blogs.rsa.com/lions-at-the-watering-hole-the-voho-affair, December 2012.

[11] Weijie Han, Jingfeng Xue, Yong Wang, Fuquan Zhang, and Xianwei Gao. Aptmalinsight: Identify and cognize apt malware based on system call information and ontology knowledge framework. *Information Sciences*, 546:633–664, 2021.

[12] Eric M Hutchins, Michael J Cloppert, and Rohan M Amin. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research*, 1(1):80, 2011.

[13] Georgios Ioannou, Panos Louvieris, Natalie Clewley, and Gavin Powell. A markov multi-phase transferable belief model: An application for predicting data exfiltration APTs. In *Proceedings of the 16th International Conference on Information Fusion*, pages 842–849, July 2013.

[14] Yong-Ho Kim and Won Hyung Park. A study on cyber threat prediction based on intrusion detection event for apt attack detection. *Multimedia tools and applications*, 71(2):685–698, 2014.

[15] Ayush Kumar and Vrizlynn LL Thing. Raptor: Advanced persistent threat detection in industrial iot via attack stage correlation. *arXiv preprint arXiv:2301.11524*, 2023.

[16] Georgiy Levchuk, John Colonna-Romano, and Mohammed Eslami. Application of graph-based semi-supervised learning for development of cyber COP and network intrusion detection. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 10206 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, page 102060D, May 2017.

[17] Meicong Li, Wei Huang, Yongbin Wang, Wenqing Fan, and Jianfang Li. The study of apt attack stage model. In *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, pages 1–5. IEEE, 2016.

[18] Shudong Li, Qianqing Zhang, Xiaobo Wu, Weihong Han, and Zhihong Tian. Attribution classification method of apt malware in iot using machine learning techniques. *Security and Communication Networks*, 2021, 2021.

[19] Wei Lu. Detecting malicious attacks using principal component analysis in medical cyber-physical systems. In Issa Traoré, Isaac Woungang, and Sherif Saad, editors, *Artificial Intelligence for Cyber-Physical Systems Hardening*, chapter 9. Springer Nature Switzerland AG, Gewerbestrasse 11, 6330 Cham, Switzerland, 2022.

[20] Robert Luh, Stefan Marschalek, Manfred Kaiser, Helge Janicke, and Sebastian Schrittwieser. Semantics-aware detection of targeted attacks: a survey. *Journal of Computer Virology and Hacking Techniques*, 13(1):47–85, February 2017.

[21] Mandiant. APT1 – exposing one of China's cyber espionage units. https://www.fireeye.com/content/dam/fireeye-www/services/pdfs/mandiant-apt1-report.pdf, 2013.

[22] Sadegh M Milajerdi, Rigel Gjomemo, Birhanu Eshete, Ramachandran Sekar, and VN Venkatakrishnan. Holmes: real-time apt detection through correlation of suspicious information flows. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1137–1152. IEEE, 2019.

[23] Daesung Moon, Hyungjin Im, Ikkyun Kim, and Jong Hyuk Park. DTB-IDS: an intrusion detection system based on decision tree using behavior analysis for preventing apt attacks. *The Journal of Supercomputing*, 73(7):2881–2895, July 2017.

[24] Sowmya Myneni, Ankur Chowdhary, Abdulhakim Sabur, Sailik Sengupta, Garima Agrawal, Dijiang Huang, and Myong Kang. Dapt 2020 - constructing a benchmark dataset for advanced persistent threats. In Gang Wang, Arridhana Ciptadi, and Ali Ahmadzadeh, editors, *Deployable Machine Learning for Security Defense*, pages 138–163, Cham, 2020. Springer International Publishing.

[25] Helmut Neuschmied, Martin Winter, Branka Stojanović, Katharina Hofer-Schmitz, Josip Božić, and Ulrike Kleb. Apt-attack detection based on multi-stage autoencoders. *Applied Sciences*, 12(13):6816, 2022.

[26] Weina Niu, Xiaosong Zhang, GuoWu Yang, Jianan Zhu, and Zhongwei Ren. Identifying apt malware domain based on mobile dns logging. *Mathematical Problems in Engineering*, 2017, 2017.

[27] Na-Eun Park, Yu-Rim Lee, Soyoung Joo, So-Yeon Kim, So-Hui Kim, Ju-Young Park, Seo-Yi Kim, and Il-Gu Lee. Performance evaluation of a fast and efficient intrusion detection framework for advanced persistent threat-based cyberattacks. *Computers and Electrical Engineering*, 105:108548, 2023.

[28] Bruce Schneier. Attack trees. *Dr. Dobb's Journal*, 24(12):21–29, 1999.

[29] Aditya K. Sood and Richard J. Enbody. Targeted cyberattacks: A superset of advanced persistent threats. *IEEE Security Privacy*, 11(1):54–61, January 2013.

[30] Chandola Varun, Banerjee Arindam, and Kumar Vipin. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41:1–58, July 2009.

[31] Nikos Virvilis, Dimitris Gritzalis, and Theodoros Apostolopoulos. Trusted computing vs. advanced persistent threats: Can a defender win this game? In *2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing*, pages 396–403, December 2013.

[32] Borislava Vrigazova. The proportion for splitting data into training and test set for the bootstrap in classification problems. *Business Systems Research: International Journal of the Society for Advancing Innovation and Research in Economy*, 12(1):228–242, 2021.

[33] Guozhu Wang, Yiwen Cui, Jie Wang, Lihua Wu, and Guanyu Hu. A novel method for detecting advanced persistent threat attack based on belief rule base. *Applied Sciences*, 11(21):9899, 2021.

[34] Tyler Wrightson. *Advanced Persistent Threat Hacking: The Art and Science of Hacking Any Organization.* McGraw-Hill Education Group, 1st edition, 2014.

[35] Waleed A. Yousef, Issa Traore, and William Briguglio. Classifier calibration: with application to threat scores in cybersecurity. *IEEE Transactions on Dependable and Secure Computing*, 2022.

[36] Ahmed1 Yussuf, Asyhari A. Taufiq, and Rahman Md Arafatu. A cyber kill chain approach for detecting advanced persistent threats. *Computers,Materials & Continua*, 67:2497–2513, 2021.

[37] Guodong Zhao, Ke Xu, Lei Xu, and Bo Wu. Detecting apt malware infections based on malicious dns and traffic analysis. *IEEE access*, 3:1132–1142, 2015.