Advanced Persistent Threat Stage Prediction

João Pedro Marinho Pires

Dissertation to obtain the Master of Science Degree in

Military Electrical Engineering Signals Specialty

Supervisor: Professor Doutor Miguel Nuno Dias Alves Pupo Correia
Co-Supervisor: Doutor Luís Filipe Xavier Mendonça Dias

Examination Committee:
Chairperson: Professor João Manuel de Freitas Xavier
Supervisor: Professor Miguel Nuno Dias Alves Pupo Correia
Members of the Committee: Professor João Nuno De Oliveira e Silva
Professor José Silvestre Serra da Silva
Coronel TM Henrique Martins dos Santos Cunha

Novembro 2023

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

First of all I want to express my most gratitude to my supervisors, Professor Doutor Miguel Correia and Doutor Luís Dias, for their generosity in having allowed me to pursue my will. For being a shining example of great professionals, for their precious advice and motivation. I have benefited enormously from their guidance since the earlier reunions, I would have made numerous errors and mistakes, and any that may still exist are solely my responsibility.

I am also indebted to Escola Secundária de Fafe, Instituto Superior Técnico (IST), and Academia Militar (AM) for their education and all the resources they made available to me. I will always appreciate the challenges that contributed to my academic and personal growth.

To my biggest supporters, my Father, my Mother and brother, who have made countless sacrifices along the way and for instilling in me the most important values in life. Thank you for you relentless and unconditional support.

I extend my heartfelt gratitude to Rita Gonçalves whose unwavering support and encouragement have been invaluable throughout the completion of this thesis. Her constant belief in my abilities, unwavering patience, and enduring love made this achievement possible.

To the FOXTROT class of 17/18 and all my friends, for all the adventures we had together, I will keep found memories for the rest of my life. This journey and work would not be possible without you.

Abstract

Advanced Persistent Threat (APT) have become one of the primary challenges in cyber defense. Characterized by sophisticated and prolonged attacks, these threats infiltrate networks aiming to steal sensitive data, often remaining undetected for extended periods. This evolution in attack tactics underscores the urgent need for improvements in defense strategies and threat detection.

Within the scope of this thesis, a framework named *Advanced Persistent Threat Stage Prediction (APTSP)* was developed. APTSP is capable of predicting, based on identified threats, the current stage of the attack, as well as the most likely subsequent stage. It also provides insights into the most probable perpetrating APT group, considering known APTs. To achieve this, APTSP takes network data classified by an Intrusion Detection System (IDS) and applies a Markov model to determine the probabilities for the APT stages. It also uses a machine learning model to identify the potential agent responsible for the attack.

APTSP was experimentally evaluated on a public dataset, comparing its results with different solutions. APTSP outperformed previous approaches in all the metrics used.

Keywords

Advanced Persistent Threat (APT); Markov model; stage of the attack; identify the potential agent; cyber defense.

Resumo

As Ameaças Persistentes Avançadas tornaram-se um dos principais desafios da ciberdefesa. Caraterizadas por ataques sofisticados e prolongados, estes ataques infiltram-se nas redes com o objetivo de roubar dados sensíveis, permanecendo frequentemente indetetadas por longos períodos. Esta evolução nas táticas de ataque sublinha a necessidade urgente de melhorias nas estratégias de defesa e na deteção de ameaças.

No âmbito desta tese desenvolveu-se uma ferramenta capaz de prever com base em ameaças identificadas, o estágio em que se encontra o ataque, assim como o estágio mais provável de acontecer no futuro, fornecendo ainda o grupo perpetuador do APT mais provável tendo em conta APTs já conhecidas. Para esse efeito APTSP recebe dados de rede classificados por um sistema de deteção de intrusões, ao qual aplica um modelo de Markov para obter as probabilidades para os estágios do APT, assim como aplica um modelo de aprendizagem automática para identificar o possível agente responsável pelo mesmo.

APTSP foi avaliado experimentalmente num dataset público e onde foi feita a comparação dos resultados com diferentes soluções. APTSP obteve melhores resultados em todas as métricas usadas em relação ás abordagens anteriores.

Palavras Chave

Ameaças Persistentes Avançadas; modelo de Markov; estágio do APT; identificar os agentes; ciberdefesa.

Contents

List of Figures

List of Tables

Acronyms

APT	Advanced Persistent Threat
APTSP	Advanced Persistent Threat Stage Prediction
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
ML	Machine Learning
NIST	National Institute of Standards and Technology
SIEM	Security Information and Event Management
TTP	Tactics, Techniques, and Procedures

1

Introduction

Contents

This Chapter presents the problem and motivation of this report. In this chapter, we also outline the objectives we aim to achieve with this work, while also presenting the structure of the document.

1.1 Problem and Motivation

In recent years cyberspace has become an integral part of everyday life and the foundation of most economic, commercial, cultural, social, and governmental interactions. The increasing threat of cyber attacks poses not only a financial risk, but also threatens the operations and survival of businesses, organizations, and government entities [7].

Cybersecurity refers to the technologies and practices that aim to protect computers, networks, programs, and data from attacks, unauthorized access, modification, or destruction. Cybersecurity systems consist of both network security systems and computer (host) security systems, which typically include a firewall, antivirus software, and an Intrusion Detection System (IDS) at a minimum [8]. Cyber threats have increasingly targeted private and corporate sectors, nation-states, and their connected entities [3]. Although the volume and sophistication of cyber attacks are increasing, the state of the art of cybersecurity solutions has yet to keep up [4].

In the face of growing cyber threats, improving cyber information sources and a standardised cybersecurity knowledge database is critical to identify and combat these emerging threats. There are already efforts underway to create globally accessible knowledge bases, such as the Common Vulnerabilities and Exposures list and the MITRE Adversarial Tactics, Techniques & Common Knowledge (ATT&CK) Enterprise Matrix developed by the MITRE Corporation [7].

Advanced Persistent Threat (APT) represents a sophisticated cybersecurity challenge, stemming from highly resourced entities aiming to extract vital data from their targets. The expression APT is distilled from three key descriptors. The term “Advanced” hilights how these adversaries demonstrate a high level of skill in their choice of tools, expertise, and assault methodologies. They meticulously tailor their attack strategies to their targets, often orchestrating them in several phases. Persistent stands for the culprits unwavering commitment to fulfilling their malicious intent. Their tactics are characterized by stealthy maneuvers designed to bypass the protective measures set in place by network guardians. APTs initially targeting nation-states and their associated entities have expanded the target zone to include the private and corporate sectors. These threats become widely known as APT, a class of cyber threats that have traditionally targeted nation-states and their associated entities. However, these threats have recently expanded to include the private and corporate sectors, causing concern among these organisations. APT attacks can also be sponsered by nations and these are known for their sophistication. APT attacks in the corporate sector may not be as advanced, but they can still be equally challenging for organisations to defend against. Protecting against APTs is a top priority for nations and established organisations [3].

Although enterprise IDS and Intrusion Prevention System (IPS) systems can detect and alert for sus-

picious events on a host, it is difficult to use these low-level alerts to gain a comprehensive understanding of an active APT campaign. Today, alert correlation is typically performed using Security Information and Event Management (SIEM) systems. A SIEM gathers log events and alerts from various sources and try to identify connections between them. These systems often rely on straightforward indicators, like timestamps, to perform this correlation. While these methods can be useful, they do not always have the ability to comprehend the intricate relationships between alerts and actual intrusions or to accurately reconstruct the sequence of events in an attack that occurs across multiple hosts over an extended period of time [2].

To effectively defend against APT, it is necessary to accurately evaluate the security of the targeted cyber networks. However, traditional security evaluation methods, which are designed to handle one-time or repeated cyber attacks, are not suitable for addressing APT, which are characterized by their sustained, continuous nature [9].

According to the National Institute of Standards and Technology (NIST), an APT group is characterized by their persistence, adaptability, and determination. They will repeatedly attempt to achieve their objectives over a long period of time and will adapt to efforts to resist them. These objectives may include the theft of information or the disruption of critical aspects of a mission or program through multiple attack vectors. The attacker will maintain the necessary level of interaction with the target to achieve their objectives [3].

1.2 Objectives

In this paper we propose a framework created for APT identification and stage prediction named Advanced Persistent Threat Stage Prediction (APTSP). Our framework should be integrated with an IDS or SIEM to establish a connection between detected threats, aiming to determine the presence of an APT. APTSP goal is to aid defenders understand what the agents responsible for the APT are targeting and so prevent and mitigate their malicious actions. In the event of an APT, this tool enables the calculation of the most likely stage of the kill chain based on the detected threat, as well as the calculation of the next most probable stage. Finally, our tool attempts to identify the APT actor using a database containing various malicious techniques employed by known APTs. APTSP extracts features from a processed network data by a IDS, than implements a Markov model to predict the most likely stages of an APT, present and also a Machine Learning (ML) model to try identify the agent responsible for the attack.

We implemented and compared APTSP with three semi-supervised models e Stacked Auto-Encoder (SAE), the LSTM Stacked Auto-Encoder (LSTM-SAE), and a single-class Support Vector Machine (SVM) [6]. We tested APTSP on the dataset DAPT 2020 [6] and compared with the results obtained by the previous models. Our evaluation shows that APTSP was able to predict the stages for multiple

threats presented in the dataset. APTSP also achieving high values in every metric. Moreover, APTSP improves previous works by having predictions for the stages Lateral Movement and Mission Complete.

1.3 Report Outline

This document consists of six chapters. Chapter 2 covers the concepts and components of the base framework. Chapter 3 discusses prior work in this area by other authors. Chapter 4 presents the proposed architecture of the system being developed. Chapter 5 presents the results produced by our framework. Finally, Chapter 6 displays the conclusions achieved and the future work.

2

Background

Contents

This chapter provides an academic literature review of the fundamental concepts that sustain the research presented in this work.

2.1 MITRE ATT&CK

MITRE was founded in 1958 by the U.S. Air Force as a not-for-profit company to provide an objective advice in systems engineering to both military and civilian government agencies [10]. As of the current MITRE, focus areas vary from AI & Machine Learning, Cybersecurity, Defence & Intelligence, and Homeland Security [11]. MITRE, as a company, exists to convene the government, industry, and academia to face significant societal challenges without competing with the industry.

MITRE also operates the CWE (Common Weakness Enumeration) List, a community-created list of software, firmware, hardware, or other service components' weaknesses and vulnerabilities [12]. CWE aims to mitigate software and hardware vulnerabilities in products before they are delivered by educating developers, architects, and designers on how to eliminate the most dangerous weaknesses. The system helps security practitioners and developers prevent hardware and software vulnerabilities before deployment, find weaknesses in existing software or hardware products, and evaluate coverage tools for these vulnerabilities [13].

According to the Cybersecurity and Infrastructure Security Agency (CISA), the first step in protecting networks and data is often to understand the adversary's behaviour. The MITRE ATT&CK framework is a worldwide available knowledge of adversary tactics and techniques derived from prior real-world observations [14]. ATT&CK provides detailed information on 100+ threat actor groups, such as reported Software used and attributed Campaigns they have been mapped using. ATT&CK can identify defensive gaps, assess security tool capabilities, organise detections, hunt for threats, engage in red team activities, or validate mitigation controls.

The MITRE ATT&CK framework is designed to be used by a wide range of cybersecurity professionals, including incident responders, threat hunters, and security analysts, to understand better, detect, and respond to cyber threats. Organisations also use it to assess their defences and identify gaps in their coverage. The framework is updated regularly to reflect the evolving threat landscape. It is widely used as a reference for cybersecurity best practices and as a benchmark for evaluating the effectiveness of security products and services [1].

2.1.1 AT&TCK Model

ATT&CK is a framework that categorises various techniques and sub-techniques that adversaries may use to achieve specific objectives. These techniques are grouped under tactic categories, providing a comprehensive understanding of the technical details of the actions taken and the reasons behind them. The framework balances providing sufficient technical detail and contextual information about the tactics used [1]. ATT&CK presents behaviours across the adversary life-cycle, commonly known as Tactics, Techniques, and Procedures (TTP)s. In ATT&CK, these behaviours belong to four increasingly granular levels [1]:

- *Tatics*: Denoting short-term, tactical adversary goals during an attack.
- *Techniques*: Describing how adversaries achieve tactical goals.
- *Sub-techniques*: Describing more specific means by which adversaries achieve tactical goals at a lower level than techniques.
- *Procedures*: Procedures in ATT&CK describe the ways in which adversaries employ techniques and sub-techniques. It's important to note that a single procedure may involve multiple techniques and sub-techniques.

Each component of the ATT&CK framework is connected in some way. The relationships described before can be seen in the following Figure 2.1:

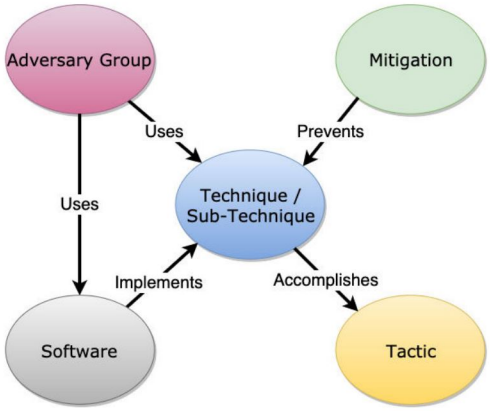


Figure 2.1: *ATT&CK Model Relationships [1]*

2.1.2 ATT&CK Matrix

The ATT&CK Matrix for Enterprise visually represents the relationships between tactics, techniques, and sub-techniques. Figure 2.2 illustrates this relationship.

Figure 2.2: *The ATT&CK for Enterprise Matrix* [1]

Additionally, some techniques can be decomposed into smaller, more specific sub-techniques that provide further insight into how to execute them effectively [1].

2.2 Advanced Persistent Threats

Today, attacks against companies, organisations and countries often involve sustained, targeted efforts known as APT. These types of attacks are designed to evade detection and continue for an extended period of time [15]. The term APT was originally introduced in 2006 by US Army Air Force specialists who were observing unidentified intrusion activities [16]. APT attacks are executed by well-funded groups who have a specific plan in place to access a target confidential information or data. These types of attacks involve multiple steps and are sustained over a long period of time, allowing the attackers to remain in the victim's system undetected for months [3].

APT differ from traditional cyberattacks in that they are highly targeted and specifically focus on organizations, government institutions, and commercial enterprises, rather than individual systems. In contrast, traditional attacks are often indiscriminate and do not have a specific target in mind [3]. There are also differences in the approach taken by traditional attacks and APT attacks. Traditional attacks tend to be noisier and are typically completed in a single attempt, while APT attacks involve repeated attempts, sustained persistence over a long period of time, and are designed to remain undetected by adapting to the victim's defences [15]. Traditional attacks can be easier to prevent because they are generic and predictable. In contrast, APT attacks require defenders to constantly adapt their detection systems and methods because they use techniques that may be unfamiliar or unexpected. This makes APT attacks more difficult to detect and defend against [15].

In 2013, cybersecurity firm Mandiant published an article discussing the findings of their investigation into APT attacks carried out by one of the largest APT organizations. According to the report, this group had been targeting a wide range of victims for several years, starting in 2006, by maintaining an extensive network of computers around the world [17].

The continued evolution and sophistication of attack methods and tools highlight the need for strong defence strategies to be implemented by organizations seeking to protect themselves and their data. These defence methods should be applied at every stage of an APT attack to be effective [3].

To effectively defend against APT attacks, it is important to understand the motivations and tactics of the attackers. This includes understanding how they operate and what drives them to carry out these types of attacks [2]. To gain a deeper understanding of APT attacks and how to defend against them, the Mandiant report created an APT lifecycle model shown in Figure 2.3, otherwise refered to as Kill-chain, consisting in a series of steps that allow one to understand how the APT actors are able to achieve their goals [17].

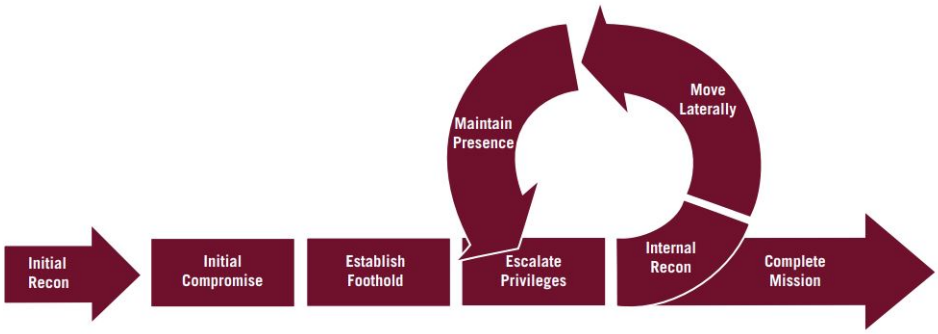


Figure 2.3: *Mandiant's Attack Lifecycle Model* [2]

IDS systems in an enterprise can generate alerts for suspicious activity on a host, but it can be difficult to use these low-level alerts to gain a complete understanding of an ongoing APT campaign [2].Alert correlation, which involves combining multiple alerts from different security systems to identify and understand a larger attack or threat, is typically done using SIEM systems like Splunk, LogRhythm, and IBM QRadar [2].

2.2.1 APT Defensive Methods

To effectively defend against an APT attack, a defence-in-depth approach that utilizes multiple defence mechanisms at different levels of the network must be implemented. These defence measures should be designed to detect and prevent different stages of an APT attack. By using a variety of defence mechanisms and correlating the events they generate, it is possible to create a robust defence that is less likely to be penetrated by attackers. A defence-in-depth approach also allows for a more flexible response to an APT attack, as it gives defenders time to assess the risk and develop a plan to mitigate the threat [3].

Yang et al. [9] conducted a security assessment of cyber networks subjected to APT. They modelled these networks under the threat of APTs launched by a strategic attacker and used the equilibrium of the cyber network as a security metric. They analyzed the effect of various attack and defence strategies on this equilibrium metric. The authors confirmed through their analysis that the security equilibrium of a cyber network decreases as the resources used for attacking a node increase, and increases as the resources used for preventing or recovering a node increase. They also studied the effect of three other factors on the security equilibrium: the addition of new edges to the network, the relationship between prevention and recovery resources, and the level of defence resources. They found that the security equilibrium decreases with the addition of new edges, is highest when prevention and recovery resources are balanced, and increases with the increase of defence resources. Based on these findings, the authors concluded that cyber networks with dense connections are more vulnerable to APT attacks than those with sparse connections, and recommend distributing defence resources equally between prevention and

recovery, as recovery is just as important as prevention in the context of APT attacks.

APT defence methods can be categorised in 3 major categories, Monitoring Methods, Mitigation Methods and Detection Methods [3]. Each category or class can be further divided into subcategories, as illustrated in Figure 2.4.

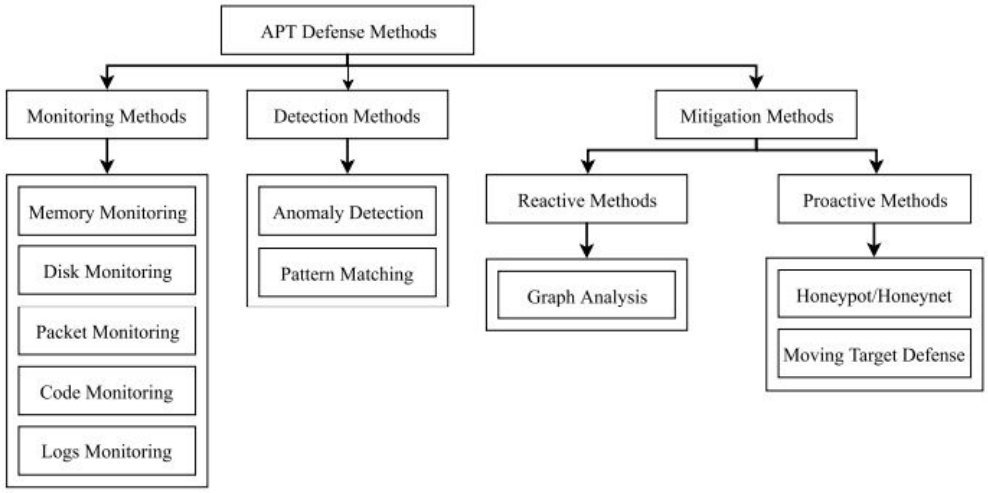


Figure 2.4: *APT defense method classifications* [3]

2.3 Machine Learning

ML has been increasingly used in cybersecurity to improve the ability to detect and prevent cyber attacks. One common use of ML in cybersecurity is in the detection of malware and malicious activity. By training ML algorithms on large data sets of known malware and benign software, it is possible to develop models that can accurately identify new, previously unseen malware [18]. ML can also be used to detect anomalies in network traffic, which may indicate a cyber attack. For example, ML algorithms can be trained on normal network traffic patterns and then used to flag any deviations from those patterns as potentially malicious [19]. In addition to detecting cyber attacks, ML can also be used to predict and prevent future attacks by analyzing past attack data and identifying patterns that can be used to improve security measures [20].

When designing a ML approach to solving a specific problem, the following steps are typically performed: identifying the class attributes and classes in the training data, selecting a subset of attributes that are necessary for classification, training the model using the training data, and using the trained model to classify unknown data. In theory, an ML approach has two phases: training and testing. However, in practice, it is often split into three phases: training, validation, and testing [8]. Machine learning techniques can generally be classified into three categories: unsupervised, semi-supervised, and supervised.

Authors in [8,21,22] presented a summary of anomaly detection systems that utilize machine learning techniques. They discussed the most commonly ML used approaches, including:

- **Association Rule**: An association rule is deemed robust when both the rule's support and confidence exceed predefined user-established thresholds for minimum support and minimum confidence. Association rules offer benefits in uncovering intriguing connections, including causality, between subsets of items (attributes) and class labels. Robust association rules can categorize prevalent patterns of attribute-value pairs into different class labels. Nonetheless, comprehending all noteworthy relationships through rules can introduce computational intricacies, even with moderately sized datasets. Restricting and trimming the rule space can accelerate the process of association rule mining.

- **Support Vector Machines (SVM)**: It functions as a classifier by identifying a discriminative hyperplane in the feature space that maximizes the distance between the hyperplane and the nearest data points from each class. Support Vector Machines (SVM) excel in accurately determining linear, nonlinear, and intricate classification boundaries, even when working with a limited training dataset.

  SVM can also be employed for outlier detection. In this scenario, it operates on the assumption that the origin of a kernel-based transformed representation belongs to the outlier category. Once the model has been trained, it is capable of assigning scores to any given data point. According to the modeling assumptions of the support-vector machine, a negative score value indicates that the data point is an outlier, while a positive score value signifies that it is a non-outlier.

  As per the findings in [22], this method exhibits high speed; however, its runtime increases fourfold when the sample data size doubles. Support Vector Machines are fundamentally rooted in binary classification, and the selection of appropriate kernel functions and associated parameter fine-tuning often involves a trial-and-error process.

- **Naive Bayes (NB)**: The Bayesian Network (BN), often referred to as the belief network, employs a factored representation of a joint probability distribution within a graphical model to make decisions regarding uncertain variables. The BN classifier operates on the principles of Bayes rule. Naive Bayes is a straightforward BN model that makes the simplifying assumption that all variables are independent. When applying the Bayes rule for NB classification, our goal is to identify the maximum likelihood hypothesis, which in turn assigns a class label to each testing data point, than the NB classifier is resolved by determining the Maximum A Posteriori (MAP) hypothesis for the given data. NB is known for its efficiency in performing inference tasks. Despite relying on a strong independence assumption among the variables, it is noteworthy that this method can yield favorable results even when the independence assumption is not strictly upheld.

- **K-Nearest Neighbor (KNN)**: In KNN, each data point is assigned the label with the highest confidence from among the k data points closest to the query point. The choice of the number of nearest neighbors (k) and the distance metric are crucial aspects of the KNN algorithm. Determining the appropriate k value should involve a cross-validation process across various k settings. Generally, a larger k mitigates the impact of data noise on classification but may also lead to a blending of class distinctions. A practical guideline is to set k to be less than the square root of the total number of training patterns. For two-class classification problems, it's advisable to select odd values of k to avoid tied votes. KNN offers the advantage of not requiring parameter training, making it straightforward to implement and interpret. However, KNN classification can be computationally time-consuming and demanding in terms of storage.

- **Artificial Neural Networks (ANN)** : ANN is a machine learning model that accomplishes the transformation of inputs into outputs, aligning them with predefined targets. This is achieved through non-linear information processing within a network of interconnected artificial neurons. The process begins with input data that activates the neurons in the initial layer of the network. The output from this layer serves as the input for the subsequent layer of neurons, and this sequence continues until the final layer produces the desired result.

  ANN methods exhibit strong performance in tasks involving the classification or prediction of latent variables that are challenging to directly measure. They are also effective at solving non-linear classification problems that are less affected by outlier data points. It's worth noting that ANN outputs may be more complex and less straightforward to interpret compared to results obtained from classification methods assuming functional relationships between data points, such as those relying on association rules.

  One important characteristic of ANN methods is their dependence on data. The performance of an ANN can significantly improve as the size of the sample data increases. To accommodate the computational demands of advanced ANN models, they are commonly implemented on graphics processing units (GPUs) due to their enhanced processing capabilities.

- **Hidden Markov Model**: HMM is proficient in addressing sequential supervised learning problems. It offers an elegant and robust approach to accurately classify or predict the hidden states of observed sequences, particularly when the data conform to the Markov property. Nonetheless, when the actual relationship between hidden sequential states diverges from the prescribed HMM structure, the performance of HMM in classification or prediction deteriorates.

  Moreover, HMM encounters challenges when handling extensive training datasets and intricate computations, especially in cases of lengthy sequences with numerous labels. The assumption of independence between past or future states and the current states also poses a hindrance to

achieving high classification or prediction accuracy using HMM.

- **k-Means**: The k-Means clustering technique partitions the given data points into k clusters. In this partitioning, each data point is assigned to a cluster whose centroid is closest to that data point when compared to the centroids of other clusters. The k-means clustering algorithm generally follows these steps:

  - Choose the initial cluster centroids.
  - Assign each instance x in the dataset to the cluster with the nearest centroid.
  - Recalculate the centroids for each cluster based on the data points it contains.
  - Repeat the first and second steps until convergence is achieved.

  Repeat the first and second step until convergence is achieved. Two critical considerations for the successful application of the k-means method are determining the number of clusters, "k," for partitioning, and selecting an appropriate distance metric. The Euclidean distance is a commonly used metric in k-means clustering. Without prior knowledge of the optimal cluster number "k," no evaluation method can guarantee that the selected value of"k" is optimal. Nevertheless, researchers have explored various metrics, including stability and accuracy, to assess the performance of clustering.

Nath and Mehtre [23] investigated the effectiveness of four machine learning methods for static analysis of malware and found that none of them was sufficient for defending against multi-stage attacks or attacks involving multiple files that are executed in parallel or sequentially, such as APT attacks. Table 1 presents a comparison of various anomaly-based APT attack defence methods, including their learning methods (supervised, unsupervised, and semi-supervised) and detection techniques (statistical and machine learning-based techniques, including rule-based and neural network approaches) presented on Table 2.1.

Table 2.1: Comparison between Anomaly Detection Based Solutions [3].

Learning Approach	Anomaly Detection Method	Source of Data	APT stages
Semi-Supervised	Machine Learning	Network/Host logs	Establish Foothold and Lateral Movement
Supervised	Machine Learning	Network Traffic	Establish Foothold
Semi-Supervised	Statistical	Host-based logs	Accomplishing Foothold
Semi-Supervised	Machine Learning	Network Traffic	Accomplishing Foothold, Lateral Movement, Exfiltration
Supervised	Machine Learning	Malware Detection	Accomplishing Foothold
Supervised	Machine Learning	Network Traffic	Lateral Movement, Exfiltration
Supervised	Machine Learning	Network Traffic	Internal Exfiltration
Supervised	Machine Learning	Emails	Reconnaissance, Establishing Foothold
Supervised, Unsupervised	Machine Learning	Host/Network events	Not specific to a stage, established behavior profiling
Unsupervised	Machine Learning	Active Directory domain service logs	Lateral Movement, Exfiltration
Supervised	Statistical	Network traffic, Access Information	Maintaining Access, Lateral Movement, Data Exfiltration

2.4 Intrusion Detection System

In numerous computer systems, various forms of data are gathered regarding actions within the operating system, network traffic, or other user behaviors. This data can exhibit atypical patterns due to potentially harmful actions. Identifying and flagging such actions is known as intrusion detection. [24] IDS are used to identify and respond to cyber attacks. In the past, IDS typically relied on predetermined patterns or "signatures" to identify attacks. However, as the number and variety of attacks have increased, machine learning techniques are increasingly being used to improve the effectiveness of IDS. These approaches allow the IDS to learn and adapt to new attacks, enhancing its ability to identify and respond to them. IDS divides into two major categories:

- **Signature-based Detection (SD)** can be used to identify cyber attacks by comparing patterns or strings of data to a database of known attacks, or "signatures." If the IDS detects a pattern or string that matches an entry in its signature database, it will report the presence of an attack. This methods excel at identifying established attack patterns while maintaining a manageable false alarm rate. However, they necessitate regular manual updates to their databases, incorporating new rules and signatures [8]. This approach relies on knowledge about real attacks and system vulnerabilities that have been compiled and stored in the signature database. The IDS compares incoming data to this database in order to identify and respond to attacks [25]. Networks safeguarded by misconfigured detection systems experience extended periods of susceptibility, stretching from the identification of a novel attack to the development of a new signature. Misuse detection stands as the most prevalent approach in IDS [26].

- **Anomaly-based Detection (AD)** focuses on identifying activities that deviate from known or expected behaviour, such as unusual network connections, host or user interactions, and endpoint behaviour. AD relies on machine learning algorithms to identify and detect unusual activity that may indicate an attack [25]. Anomaly detection typically relies on labeled data that describe typical behaviors, whereas labeled data for anomalous behaviors are often absent. Supervised machine-learning methods require training data that is free from attacks. Nevertheless, obtaining such training data is a challenging task in real-world network environments. This shortage of training data is responsible for the widely recognized issue of imbalanced data distribution in machine learning [22]. Anomaly detection techniques typically produce two primary types of outputs. The first type is a score, representing a value that combines factors such as distance or deviation from a set of profiles or signatures, the influence of the majority within its neighborhood, and the distinct dominance of the relevant subspace. The second type of output is a label, which assigns a value of normal or anomalous to each test [19].

In recent years, machine learning has gained significance in the context of anomaly detection, with

techniques varying from domain-specific to more generic applications [19]. Literature shows IDSs have tended to produce alerts that are too numerous and low-level for human operators. Techniques needed to be developed to summarize these low level alerts and greatly reduce their volume [2].

3

Related Work

Contents

This section presents papers on APT detection systems, as well as stage prediction and APT identification. They also dive in the relation between TTPs and the APT stages. The papers explore different approaches to the problem, from clustering to Markov models, presenting their architecture and going into more detail on their implementation.

3.1 Mapping TTPs

Milajerdi et al. [2] developed HOLMES, a real-time APT detection system that correlates TTPs that might be used to carry out each APT stage. HOLMES is an effective real-time intrusion detection tool that can detect APT campaigns with high precision and low false alarm rates. It does this by generating a high-level graph that summarizes the attacker's steps in real-time and evaluating it against nine real-world APT threats.

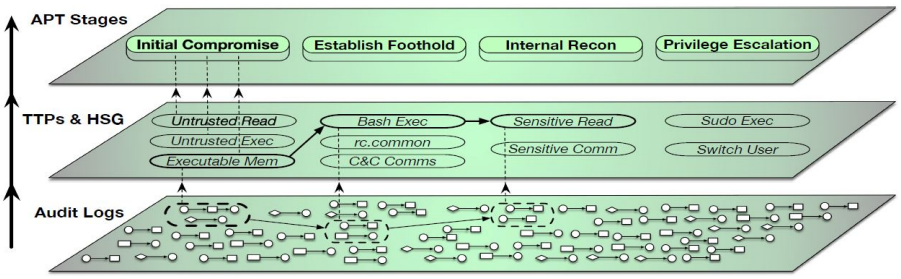


Figure 3.1: *HOLMES Approach: From Audit Records to High-Level APT Stages* [2]

To address the difference in understanding between the low-level system-call perspective and the high-level kill chain perspective, they created a middle layer as illustrated in Figure 3.1. This intermediate layer is designed using the MITRE ATT&CK framework, which outlines nearly 200 distinct behavioral patterns known as TTP that have been observed in real-world situations. The intermediate layer serves as a link between these two views. The TTPs are individual methods of achieving a specific high-level capability. For example, the capability of establishing persistence on a compromised Linux system can be achieved through the use of 11 distinct TTPs, each of which represents a possible series of lower-level actions in the ATT&CK framework, such as installing a rootkit or modifying boot scripts. These lower-level actions are more closely aligned with the level of detail found in audit logs, so it is possible to express TTPs in terms of nodes and edges in the provenance graph [2].

Milajerdi et al. [2] TTP specifications are an important aspect of our approach because they connect low-level audit events to high-level APT steps. In this subsection, we describe three key design choices for TTPs that enable efficient and accurate attack detection. In order to accurately capture the steps of a TTP, we also need to consider its prerequisites. These prerequisites help reduce false positives and provide insight into the TTP's role in the larger context of an APT campaign. In our TTP specifications,

prerequisites are represented as causal relationships and information flow between APT stages. An example of a TTP rule specification is shown in Table 3.1.

Table 3.1: TTPs Specification [2].

APT Stage	TTP	Event Family	Events	Severity	Prerequisites
$Initial\_Compromise(P)$	$Untrusted\_Read(S, P)$	READ	FileIoRead (Windows), read/pread/readv/preadv (Linux,BSD)	L	$S.ip \notin$ {Trusted_IP_Addresses}
	$Make\_Mem\_Exec(P, M)$	MPROTECT	VirtualAlloc (Windows), mprotect (Linux,BSD)	M	\$PROT_EXEC\$ $\in M.flags \wedge \exists\ Untrusted\_Read(?, P') : path\_factor(P', P) <= path\_thres$
$Establish\_Foothold(P)$	$Shell\_Exec(F, P)$	EXEC	ProcessStart (Windows), execve/fexecve (Linux,BSD)	M	$F.path \in$ {Command_Line_Utilities} $\wedge \exists\ Initial\_Compromise(P') : path\_factor(P', P) <= path\_thres$

In Table 3.1, the first column represents the APT stage, the second column lists the associated TTP name and the entities involved in the TTP, and the third column specifies the event family associated with the TTP. The fourth column includes specific events in this technique family. Lastly, the fifth column represents the danger lever of each TTP, being extremely important in the process of ranking the alarms of the system, providing various levels related to the severity of the attack. These severity levels are retrieved using MITRE ATT&CK framework [2]. The architecture of HOLMES is shown in Figure 3.2

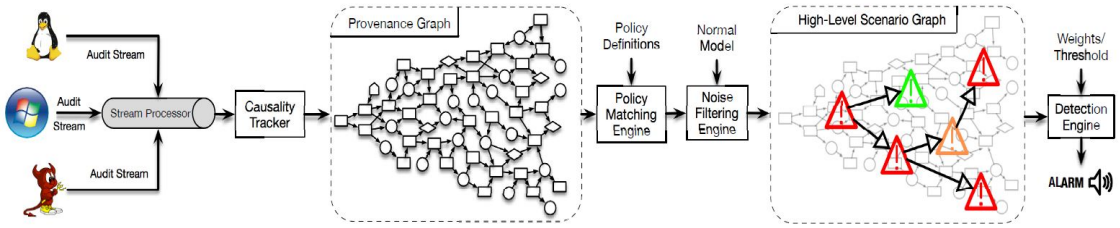
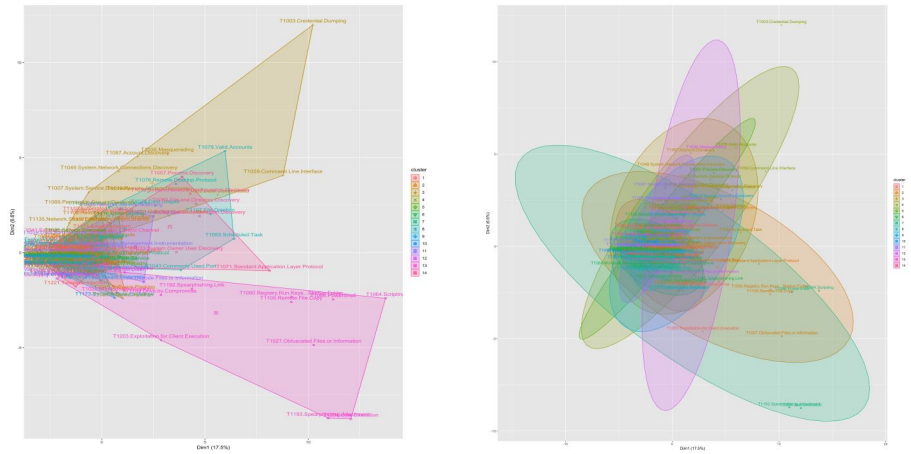


Figure 3.2: *HOLMES architeture* [2]

3.2 Clustering Techniques

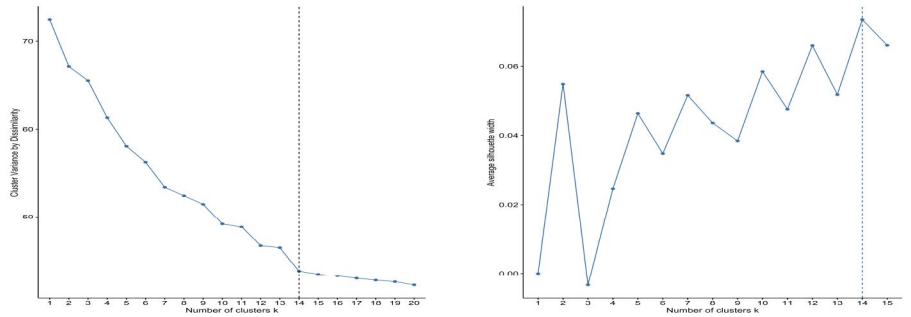
3.2.1 Partitioned Clustering

Al-Shaer et al. [4] used Partitioned clustering in order to find the optimal number of clusters to divide a dataset, being K the number of clusters. The methods used were Elbow and Silhouette methods. Despite both methods performed equally for a dataset of APT attacks, the results were not sufficient, shown in Figure 3.4. They went further and made a preliminary analysis with K-means, OAM, and Fuzzy clustering, realizing these methods did not provide a good partition of the dataset and resulted in the overlap of techniques, making it impossible to distinguish Technique association. Figure 3.3 illustrates the results obtained.



(a) PAM Clustering (b) Fuzzy Clustering

Figure 3.3: *Partitioned Clustering APT attacks* [4]



(a) Elbow method (b) Silhouette method

Figure 3.4: *K for APT attacks* [4]

3.2.2 Hierarchical Clustering

Al-Shaer et al. [4] explain Hierarchical clustering is a method of clustering data resulting in a tree-like structure, known as a dendrogram, which is constructed to represent the relationships between the data points. This method uses a distance matrix as the basis for the clustering. The dendrogram produced by hierarchical clustering shows the different levels of the hierarchy and allows the user to identify clusters by cutting the tree at a desired height. Unlike other clustering methods, the number of clusters does not need to be specified in advance. The clusters can be determined based on specific constraints after generating the tree. The results for the Hierarchical Clustering are portrayed in Figure 3.5, where is shown an example of a dendrogram and the different data points relations.

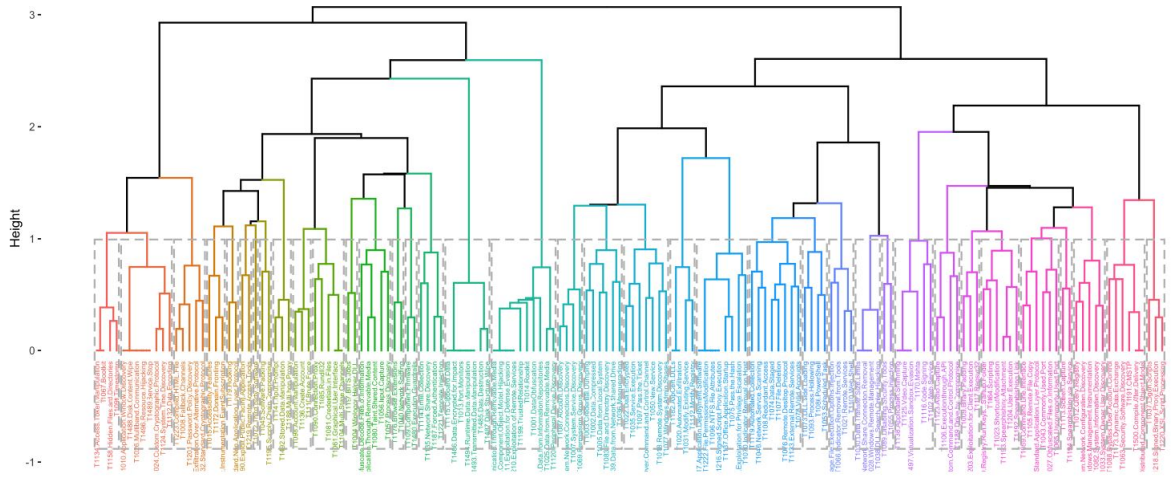


Figure 3.5: *Hierarchical Clustering of APT Attacks* [4]

After evaluating the learned attack technique associations using a theoretic approach, they computed the mutual information of the techniques in the fine-grain clusters, as well as the coarse-grain clusters. To compare the mutual information between different associations as an evaluation metric [27]. Their approach allowed them to identify 37 fine-grain technique associations in the APT attacks dataset and 61 in the Software attacks dataset. Evaluation of the learned fine-grain and coarse-grain associations using mutual information and found that 78% of the fine-grain and 75% of the coarse-grain technique associations had high predictability for APT attacks [4].

3.3 Markov Models

Shin et al. [28] developed a framework called advanced probabilistic approach for network-based IDS (APAN) accurately forecast and identify network intrusions. APAN goes beyond mere detection of attacks, it also offers an assessment of the risk level on a probabilistic scale. To determine the likelihood of a network attack, APAN constructs a Markov chain model with three primary stages. In the initial phase, it employs K-means clustering to establish network states and introduces the concept of an outlier factor. Following this, a Markov model is created, encompassing the state transition probability matrix and the initial probability distribution, all while considering practical assumptions. In the last phase, the model is utilized to dynamically assess the level of irregularity in incoming data in real-time. This contribution represents a novel endeavor to employ a Markov model in the analysis of network data with the aim of forecasting the duration and intensity of internet threats.

Allahdadi et al. [29] suggest investigating the network's temporal usage patterns using a range of Hidden Markov Models (HMMs). Unusual patterns are identified based on their likelihood. In this likelihood series, transitions that occur less frequently are more likely to be linked to anomalous instances, making

them candidates for being labeled as outliers. Moreover, HMM indicators scrutinize the characteristics of each data instance in relation to the model parameters, determining whether any signs of abnormality are associated with the data points and identifying potential sources of issues from the model parameters' perspective.

Ghafir et al. [5] proposed a system for detecting and predicting APT that consists of two phases, one for reconstructing the attack scenario and another for decoding the attack. The initial stage involves reconstructing the attack scenario through a correlation framework, while the second stage focuses on decoding the attack using HMM). Initially, the APT scenario is reconstructed by establishing connections between the alerts observed throughout the APT life cycle.

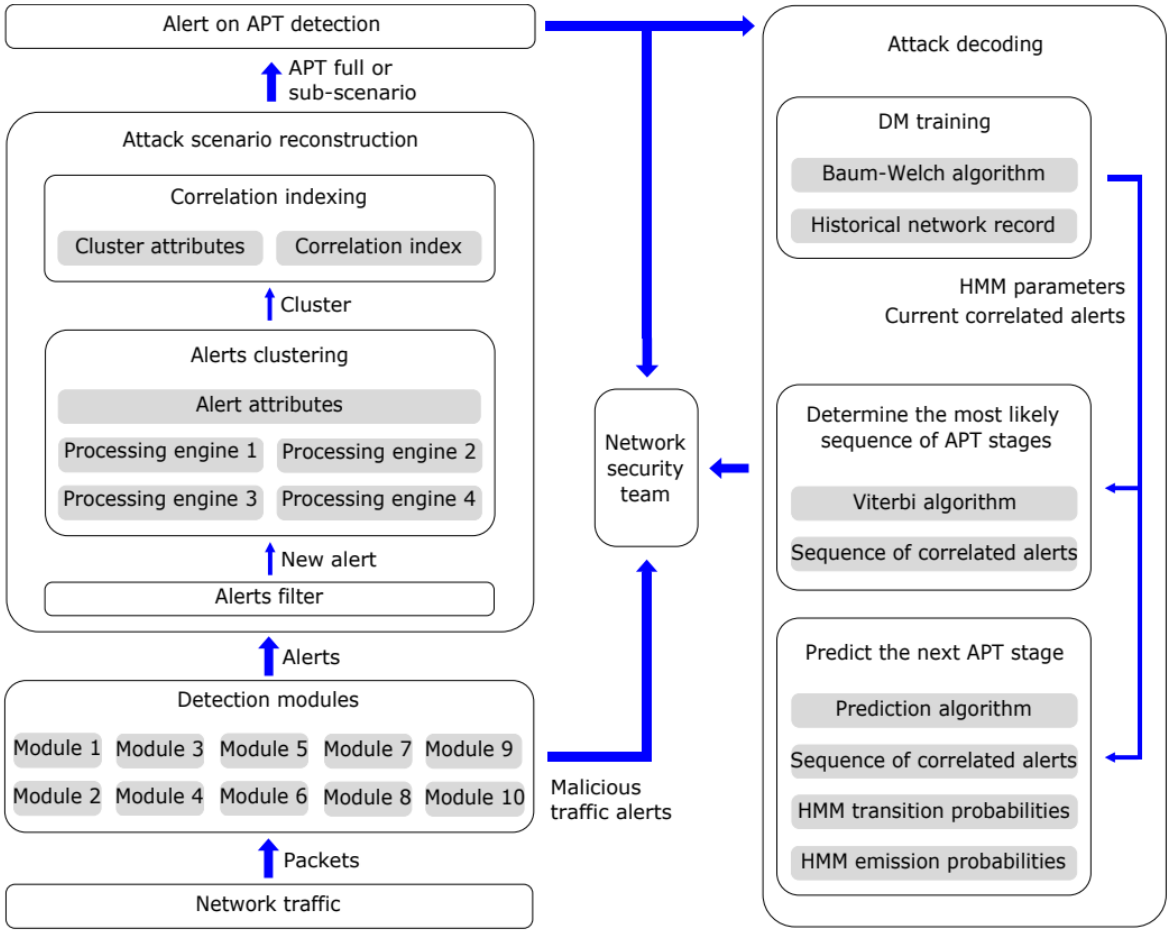


Figure 3.6: *Architecture of HMM model for APT detection and prediction* [5]

To start, the APT scenario is reconstructed by establishing connections between alerts observed throughout the APT life cycle. This correlation is determined by comparing attributes of generated alerts within a customizable time window. Next, the sequence of correlated alerts is input into a decoding module, which employs HMM to perform two specific functions. The initial function utilizes the Viterbi

algorithm to estimate the most probable sequence of APT stages for the correlated sequence of alerts during the attack reconstruction phase. The second function anticipates the next phase of the attack campaign. This prediction method employs HMM parameters to calculate the likelihood of each APT stage being the next potential step for the attacker. Subsequently, the stage with the highest probability is forecasted as the subsequent step. Figure 3.6 outlines the architecture of the model created, and it can be seen that the initial step is an IDS or a detection module triggering alerts.

Another approach using a HMM model by Brogi et al. [30] HMM to assess and prioritize the APT scenarios reconstructed by other alert correlation tools. In this context, the APT stages are regarded as the HMM states, and the correlated alerts serve as observations. The method leverages HMM parameters, including transition, emission, and initial probabilities, to calculate a probability score for each sequence of interconnected alerts. Consequently, the APT scenarios are organized in accordance with their highest probability. Consequently, APT scenarios with lower probabilities are interpreted as scenarios where alerts have been inaccurately correlated.

Kholidy et al. [31] employs a HMM to introduce an adaptable risk assessment strategy for forecasting Multistage Attacks in cloud systems. This strategy quantifies the potential effect of a malicious activity on assets by considering its probability of occurrence. Initially, the system collects diverse events from multiple detectors. These events are subsequently organized in accordance with the IDMEF protocol to facilitate event correlation. Correlation is established by comparing the events with a predefined set of attack rules, and events triggered by the same rule are regarded as correlated. Subsequently, the HMM is employed to assess the current state of the cloud system and gauge the level of threat. In this approach, four states are considered for the cloud system:

- Hale: Signifying the absence of any threat.
- Investigate: Indicating the presence of an attempted attack.
- Attack: Denoting that an attack is currently underway.
- Penetrate: Representing the completion of an attack.

The observations consist of events or alerts collected from various sensors, and they drive transitions between these states. These observations are associated with the system states and categorized into four threat levels: Low, Medium, High, and Very high. Consequently, HMM can estimate the system's status using the Viterbi algorithm and assess the risk based on the threat level of the observed alerts. This system defines the states of the HMM and estimates the threat level based on the current observation.

4

Proposed Solution

Contents

As of today enterprises have been improving their protections against intrusions. SIEM produce and monitor an enormous amount of data, although there is also the human side represented by analysts performing complex and difficult tasks. APTSP that we developed in this thesis, differentiates itself from the works presented before because the system which was implemented detects the stage of an APT attack, and provide insight in what will be the attacker's next steps. This will enhance the capability of mitigating the risks of an attack. APTSP should be integrated with an IDS/SIEM. After receiving network traffic processed by these systems, this tool extracts the necessary features for our models. We apply our constructed Markov model to predict if an APT is present, its most likely stage, and the next probable stage. Finally, we also apply our model with the Multinomial Naive Bayes classifier to determine the probability of a known APT being present. Our system will use the programming language Python in order to leverage Scikit-learn library, offering multiple ML algorithms.

4.1 Data Processing

For the empirical foundation of our research, we utilized the DAPT 2020 dataset [6]. This dataset is a rich compilation of network traffic, collected over a span of five days. Intriguingly, each of these five days is not just a mere 24-hour representation, but rather emulates approximately three months of activity in a real-world scenario. This design simulates the prolonged and stealthy nature of APT, replicating the temporal dynamics and complexities these threats manifest in live environments. By employing this dataset, our approach will be rooted in data that not only offers depth but also encapsulates the broader temporal patterns of cybersecurity threats, ensuring our framework's applicability and robustness in real-world conditions.

The initial step of our framework will involve a process of feature extraction from the dataset. The key features, which are fundamental to our analysis and subsequent implementation, are delineated in Table 4.1. In this phase, we'll also emphasize the removal of extraneous or redundant features. By streamlining our dataset in this manner, we aim to enhance the accuracy of our model while simultaneously reducing computational overhead and potential sources of noise.

We explain our approach by considering the DAPT 2020 dataset, which consists in network traffic collected from a 5 day period, where each day correspond to a approximately 3 months in a real-world scenario. First, we will extract key Features from our dataset represented in Table 4.1, eliminating unnecessary data.

Table 4.1: Features Description

Features	Description	Example
Timestamp	Time of an event	19/07/2019 04:53:29 PM
Activity	Attack methods employed in DAPT 2020	Network Scan
Stage	APT stage of each attack method	Reconnaissance

The DAPT2020 dataset offers an intricate examination of the multiple stages inherent to an APT, alongside an attack vector wielded by adversaries, as detailed in Table 4.2. This dataset systematically unfolds across pivotal phases: Reconnaissance, Establish Foothold, Lateral Movement, and Data Exfiltration. Events that are deemed routine or non-malicious are categorically classified under "Benign". A meticulous description of the extracted features will be provided to streamline the decision-making process. Importantly, the second column of Table 4.2 represents the various techniques present on the dataset and how they associate with a specific stage.

Table 4.2: APT Stages/Phases present in DAPT2020

APT Phases	**DAPT 2020**
Reconnaissance	Network Scan
	Application Scan
	Account Bruteforce
Establish Foothold	CSRF
	SQL Injection
	Malware Download
	Backdoor
	Reverse Shell
	Command Injection
Lateral Movement	Internal Scanning
	Account Discovery
	Password Dumping
	Credential Theft
	Creation of user accounts
	Privelege Escalation
Data Exfiltration	Data Theft

4.2 Defining APT Activities

To facilitate the prediction of the most probable APT, we meticulously curated dictionaries delineating the activities of each APT, drawing inspiration and guidance from the esteemed MITRE's ATT&CK framework. For each APT, we create this dictionaries with all the activities that compose them. Our research particularly emphasizes three notable APTs: APT41 [32, 33], Target Breach [34], and RSA SecureID [35]. The construction of these dictionaries was executed with due regard to the techniques distinctly manifested within the dataset.

- *APT41*: Researchers have classified the threat group APT41 as a state-sponsored Chinese espionage organization that also carries out financially motivated operations. APT41 has been observed targeting the healthcare, telecom, technology, and video game industries in 14 countries since at least 2012. APT41's coverage of organizations like BARIUM and Winnti Group overlaps, at least somewhat, with that of the general public [32,33].

- *Target Breach*: A possible attack scenario that researchers can use to test their theories. A sample

enterprise network architecture and a trace of an attack from initial compromise to data exfiltration make up the scenario [34].

- *RSA SecureID*: attack used a targeted phishing campaign with an Excel file that had a Flash object embedded in it. The attack's primary goal was to install back-door malware on computers connected to RSA's network, likely after conducting reconnaissance on social networking sites [35].

The dictionaries created for each of the previous APT's can be seen in Table 4.3, where different approaches of each attack can be seen. This enables us to map each group of activities into a specific type of APT.

Table 4.3: APT Dictionaries

APT Phases	DAPT 2020	APT41	Target Breach	RSA SecureID
Reconnaissance	Network Scan			
	Application Scan			
	Account Bruteforce	✓		
Establish Foothold	CSRF	✓		
	SQL Injection	✓	✓	
	Malware Download		✓	
	Backdoor	✓		✓
	Reverse Shell	✓	✓	
	Command Injection			
Lateral Movement	Internal Scanning	✓	✓	
	Account Discovery		✓	
	Password Dumping	✓	✓	✓
	Credential Theft	✓	✓	✓
	Creation of user accounts	✓		
	Privelege Escalation	✓		✓
Data Exfiltration	Data Theft	✓	✓	✓

4.3 Modeling for APT prediction

In this Section we delve extensively into the intricacies of our APT prediction model, breaking down each procedure in detail. Starting by we explaining how activities are categorized using binary indicators for distinct APT types. Subsequently, we discuss the mechanics of our machine learning model, detailing the creation of unique models for each APT type and the systematic transformation of data suitable for machine learning processes.

4.3.1 Adding Binary Columns

In this step, binary columns are created for each APT type to categorize activities. For example, if "IsAPT41" is a binary column, it is filled with 1 if the activity is associated with "APT41" and 0 if it is not. This process involves creating a binary indicator for each APT type to label whether a specific activity belongs to that APT type or not. The resulting DataFrame will have columns like "IsAPT41", "IsTargetBreach", "IsRSASecureID", and so on.

4.3.2 Input Features

The code prepares input features for the machine learning model. It primarily focuses on the "Activity" column, which contains textual descriptions of activities. This column is chosen as the feature that the model will use to make predictions. The "Activity" column represents the activities associated with various APT types, and the model will learn to classify these activities into their respective APT types.

4.3.3 Model Components

For each APT type defined in the dictionary (e.g., "APT41", "Target Breach", "RSA SecureID"), a separate machine learning model is created. This allows the system to classify activities into these specific APT types by checking if certain traffic patterns correspond to one of the models. Next we explain in detail how we created these models:

- *Pipeline*: The machine learning model is constructed as a pipeline. A pipeline is a way to streamline a sequence of data processing steps. In this case, it includes two main components.

- *One-Hot Encoding*: This step is used to convert categorical data (textual "Activity" descriptions) into a numerical format that the machine learning algorithm can work with. One-hot encoding represents each unique activity with a binary vector (0s and 1s), where 1 indicates the presence of an activity and 0 its absence.

- *Multinomial Naive Bayes Classifier*: The Multinomial Naive Bayes classifier is a probabilistic classification algorithm. It's suitable for text data and works well when WE have categorical features (like word counts in text) and discrete output labels. In this context, it's used to classify activities into APT types based on the encoded "Activity" descriptions.

The model is trained using the dataset DAPT2020. The "Activity" column serves as the feature, and the binary columns ("IsAPT41", "IsTargetBreach" etc.) are the labels. The model learns to make predictions based on the features (one-hot encoded activity descriptions) and the labels (binary columns indicating APT type associations).

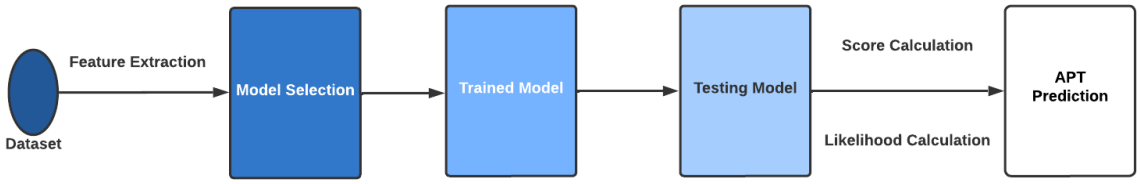


Figure 4.1: *Sequence of Steps*

4.4 Stage Prediction

This section elucidates the methodology our framework employs to predict the stages of an APT. First, we explain the process of converting stage data into numerical values, making it compatible with machine learning algorithms. We then discuss the creation and adjustment of the transition matrix, which captures the randomness and pattern of stage progressions. Based on this matrix, our forecasting function emerges as the core of our predictions. This function covers everything from starting variables to predicting stages.

4.4.1 Encoding Stages

To work with the Stage data effectively, we encoded the stage data into numerical values. This encoding is necessary for machine learning tasks, as our algorithm require a numerical input. For each unique stage we create a list called "unique stages", which is we named stage2id. In this list, we are able to map each stage name by pairing it with a unique integer. For instance, "Stage: Reconnaissance" might be mapped to the integer 0, "Stage: Lateral Movement" to 1, and so on. After this we implement a reverse mapping from integers to stage names named id2stage.This list allows us to retrieve the original stage name based on the integer values.

4.4.2 Initialize and Build the Transition Matrix

Markov Chains provide a straightforward way to model stochastic processes. They allow for the efficient computation of transition probabilities between states, making them suitable for real-time applications. By constructing a transition matrix, we can capture the likelihood of moving from one stage to another. This matrix becomes the core of the predictive model, offering insights into the flow of stages or activities. The transition matrix is a critical component of Markov Chain modeling, and it captures the probabilities of transitioning from one stage to another. The transition matrix is initialized as a square matrix using numpy. Its dimensions are determined by the number of unique stages in the dataset. The code iterates through the training data to count transitions between stages. For each consecutive pair of stages in the training data, the corresponding entry in the transition matrix is incremented. This step involves

incrementing counts and lays the groundwork for calculating transition probabilities.

4.4.3 Normalize the Transition Matrix

The transition matrix, in its current state, contains transition counts. To convert these counts into probabilities, normalization is necessary. Here's how it's done: The code divides each row of the transition matrix by the sum of its values. This normalization ensures that the transition probabilities for each stage sum to 1. It's a crucial step because it transforms counts into probabilities, making the matrix suitable for Markov Chain analysis. The normalization ensures that the probabilities represent a valid probability distribution for each stage, allowing for accurate forecasting and modeling of stage transitions.

4.4.4 Forecasting Function

The forecasting function, as discussed in this section, serves as a cornerstone in this predictive endeavor. Built upon a probabilistic foundation, this function employs a transition matrix, detailing the likelihoods of transitioning from one stage to its successor. The function is designed, incorporating stages of variable initialization, probabilistic stage selection, cumulative probability assessment, and refined prediction that excludes benign activities.

- *Initialization of Variables*: The function begins by initializing variables to keep track of the sequence of stages and the cumulative probability. These variables are crucial for monitoring the evolving state predictions and their associated likelihood.

- *Selection of Next Stage*: The next stage in the sequence is selected based on the transition probabilities provided in the transition matrix. This selection is achieved using a random choice mechanism that is weighted according to the transition probabilities. In other words, stages with higher transition probabilities are more likely to be chosen.

- *Accumulation of Probability*: As the function progresses and predicts future stages, it accumulates the probability associated with the generated sequence of stages. This cumulative probability represents the likelihood of the entire sequence.

- *Identification of Most Probable Non-BENIGN Stage*: To further analyze the results, the function identifies the most probable next stage that is not labeled as "BENIGN." It does so by excluding the "BENIGN" stage from consideration and determining the maximum transition probability among the remaining stages.

Starting with an initial stage, the function uses the transition matrix to probabilistically determine

the next stage, and then the subsequent stage, and so on, until the number of steps (days) is reached.

$$S_{t+1} = \arg\max_{s} \left(P(S_t \to s)\right) \tag{4.1}$$

Where:

- S_{t+1} is the stage at the next step.
- S_t is the stage at the current step.
- The equation is iteratively applied for the given number of steps (days).

4.4.5 Predicting the Next Stage

In order to mitigate future risks we also want to predict, the most likely stage that will occur next. For a given stage (let's denote it as S), the function examines the transition matrix row corresponding to S to identify the stage most likely to succeed it. The transition matrix essentially captures the probability of transitioning from one stage to another.

$$\text{Next Stage} = \arg\max_{s} \left(P(S \to s)\right) \tag{4.2}$$

Where:

- $P(S \to s)$ represents the probability of transitioning from stage S to stage s.
- $\arg\max$ is an operation that identifies the stage s with the maximum transition probability from stage S.

4.4.6 Transition Probability

This function retrieves the likelihood (or probability) of transitioning from a current cyber-attack stage to the most probable subsequent stage, given an activity.

For a given activity (let's denote it as A), the function first identifies the most likely stage associated with that activity (denoted as S). It then looks at the transition matrix row corresponding to S to determine the probability of transitioning to the most probable subsequent stage.

$$P = P(S \to \text{Most Likely Next Stage}) \tag{4.3}$$

Where:

- P is the probability value retrieved from the transition matrix.

- The Most Likely Next Stage is determined by the activity-to-stage mapping and the transition matrix.

4.5 Likelihood Calculation:

The likelihood denotes the probability that a specific APT type exists within the test dataset. This probability is determined based on the activities detected within the test data.

To estimate this likelihood, the Multinomial Naive Bayes classifier is employed. This classifier gauges the probability of each APT type grounded in the observed activities.

Given an APT type, for instance, "APT41", the likelihood, symbolized as $P(\text{APT41}|\text{Activities})$, is mathematically formulated as:

$$P(\text{APT41}|\text{Activities}) = \sum_{i=1}^{n} P(\text{Activity}_i|\text{APT41}) \tag{4.4}$$

In the above equation:

- $P(\text{Activity}_i|\text{APT41})$ stands for the conditional probability of observing each distinct activity i, given the presence of the APT type "APT41".
- "Activities" encompass all the activities pinpointed within the test dataset.

Such likelihood computations are analogously executed for each APT type. The resultant likelihoods for all designated APT types are then displayed in the code's output.

4.6 Score Calculation

In the code, the likelihood and score calculations are performed for each defined APT type, and the results are printed to evaluate the model's performance in identifying these APT types in the testing data. The likelihood represents the model's confidence in each APT type, while the score provides an assessment of how well it performs.

The score is calculated for each APT type to assess how well the model identifies and categorizes APT-related activities in the dataset. It measures the ratio of observed activities associated with an APT type to the total expected activities for that APT type.

The score for a specific APT type, denoted as APT_i, is calculated as:

$$\text{Score}(APT_i) = \frac{\text{Number of Observed } APT_i \text{ Activities}}{\text{Total Expected } APT_i \text{ Activities}} \tag{4.5}$$

In the above equation, the "Number of Observed APT_i Activities" is the count of activities in the testing data that are associated with APT_i. "Total Expected APT_i Activities" is the total number of activities expected to be associated with APT_i based on the dataset's predefined mapping in the dictionary. The score is a measure of how well the model's predictions align with the actual data and can be used to assess the performance of the APT detection system for each APT type.

4.7 Detection Example

In this segment, we provide a demonstrative example to clarify how APTSP functions, in particular we give detail examples for Sections 4.3, 4.4.5, 4.4.6 , 4.5 and 4.6. We consider that our SIEM has given an alert for malicius activities classified as "Account Bruteforce". We now want to discover the stage of the APT we are in and what is the most likely stage coming next, while also calculating the most likely APT that we are facing.

We than proceed to do all the data processing and feature extraction to clean the dataset and keep only the features that will characterize the APT. After we start modeling our data as explained in more detail at Section 4.3, and we obtain the matrix shown in Figure 4.2.

$$\begin{bmatrix} 0 & 0.23077 & 0.53846 & 0.23077 \\ 0.00057728 & 0.35128 & 0.48016 & 0.16799 \\ 0.00063258 & 0.34510 & 0.48387 & 0.17103 \\ 0.00088391 & 0.33795 & 0.48910 & 0.17207 \end{bmatrix}$$

Figure 4.2: Transition Matrix for the example

After concluding the transition matrix we calculate the likelihood of the stage we are in, and subsequently call the forcasting function to determine the most probable next stage, using the equation 4.1. Finally we obtain the Scores presented in Table 4.4 for our APTs, choosing the one with highest score using equation 4.5.

Table 4.4: Results from APTSP implementation

Current Stage	Reconnaissance	0.4839
Next Stage	Establish Foothold	0.3451
APT Highest Score	Target Breach	0.2412

With this chapter we give an extensive and detailed explanation of our implementation. Starting by doing feature extraction and processing responsible for giving insight into which stage of an attack an activity is most likely to correspond. A Markov Chain transition matrix is constructed which captures the likelihood of transitioning from one stage to another; Leveraging the transition matrix, a forecasting function is devised to predict the most probable stages, and finally the most likely APT.

4.8 Implementation of APTSP

The APTSP framework is implemented using the Python programming language in order to leverage Scikit-learn library, offering multiple ML algorithms. Our software is divided into two distinct programs, that provide different functionalities and comprise different components. We named both programs as Stage Prediction, and APT Identification.

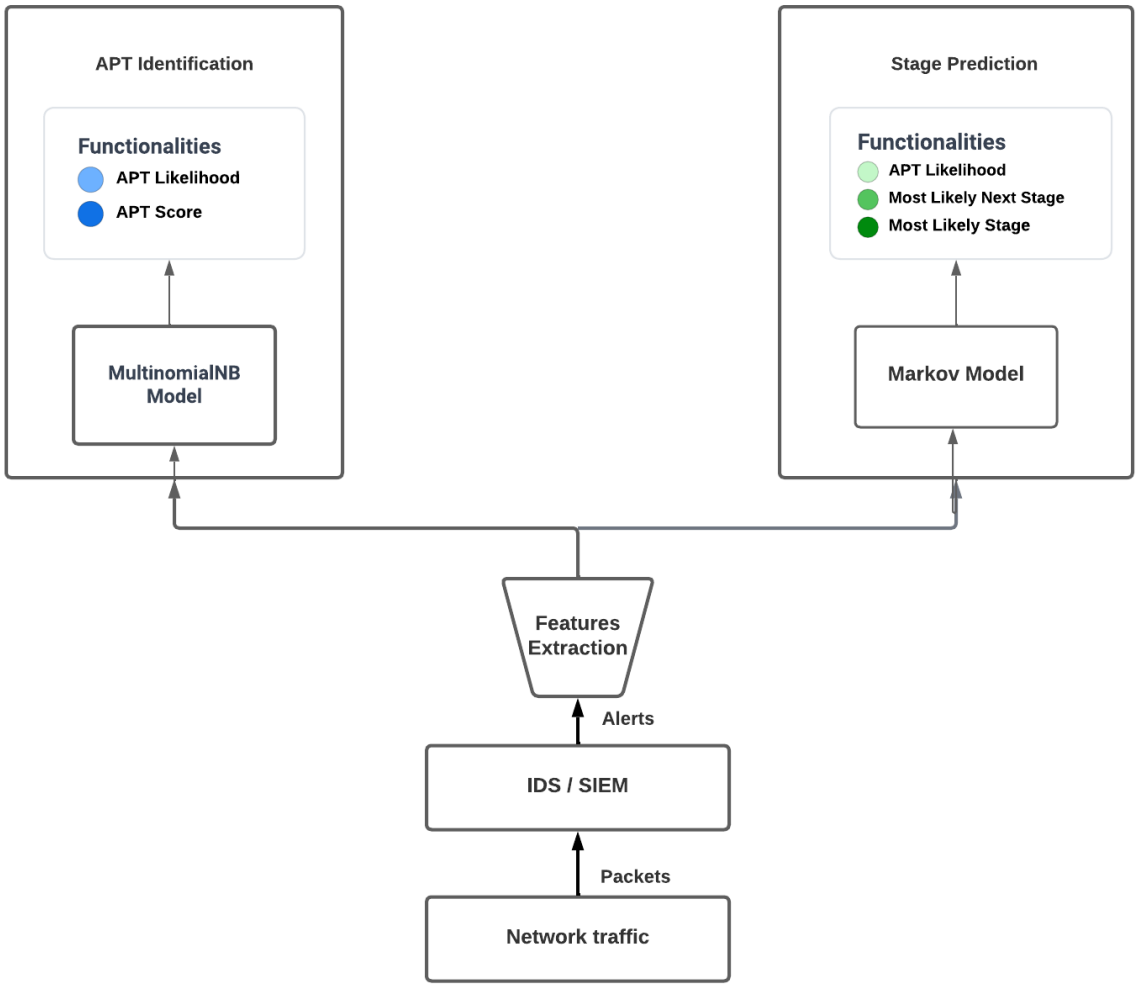


Figure 4.3: *Architecture of APTSP implementation*

Below, we present the various components and functionalities of the different programs that are presented in Figure 4.3.

- **Stage Prediction Program**: In this program we make predictions about the Stage of the APT based on a malicious technique. We are able to calculate the most likely current Stage, and make a prediction for the next stage. The program retrieves both stages with the corresponding likelihood for the user as well as the evaluation metrics for the model.

- **APT Identification Program**: In this program we identify the highest scored APT from our database. Our database consists in the techniques that comprise each of APT41, Target Breach and RSA SecureID. It retrieves to the user this scores, informing the most likely APT that is targeting the network as well as the evaluation metrics for the model.

5

Evaluation

Contents

This chapter provides an evaluation of this framework. Initially, we establish our evaluation metrics and perform dataset characterization. Subsequently, we compare the results obtained using this framework with other implementations. For the development and implementation of this framework in the evaluation process, Python (v3) [36] was utilized. In addition to Python, we made use of widely adopted libraries, including Pandas [37] for data manipulation and Scikit-learn [38] for data processing and mathematical algorithms. The primary objective of these experiments is to assess the performance of this framework by comparing it with various alternative approaches. Unfortunately, due to the scarcity of datasets in this area, we were only able to implement it in one dataset. Moreover, we were only able to compare our results with published work in this area.

5.1 Metrics

The expressions that follow represent the metrics we used to make our evaluation. We will designate **"true labels"** contains the actual stages from the dataset and **"predicted labels"** the stages predicted by our model for the test data.

True Positives (TP): Stages that were correctly predicted by the model. This means for some i, `true_labels[i]` is the same as `predicted_labels[i]`.

True Negatives (TN): In a binary classification, TN would be the instances where both the true and predicted labels are the negative class. But here, because of a multi class scenario, we have compare how each class performs in relation to all the other classes.

False Positives (FP): Stages that were wrongly predicted by the model. This would mean that some i, the model predicted `predicted_labels[i]` while the true stage was different.

False Negatives (FN): In this multi-class setup, FN for a particular stage would mean instances where the true label was that stage, but the model predicted something different.

Precision: The fraction of correctly predicted positive observations out of the total predicted positives.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{5.1}$$

Recall: The fraction of correctly predicted positive observations out of all actual positives.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{5.2}$$

F-Score: The fraction of correctly predicted positive observations out of all actual positives.

$$F\text{-Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.3}$$

Area Under the Precision-Recall Curve (AUC-PR): AUC-PR is a performance metric used to

evaluate the quality of classification models, particularly in scenarios where the datasets are imbalenced, having classes more rare than others. It focuses on the trade-off between precision and recall.

$$\text{AUC-PR} = \int_0^1 \text{Precision}(R)\, dR \tag{5.4}$$

5.2 Dataset Characterization

A critical challenge in research in APTs or APT detection is the bare availability of datasets containing APT activity.This deficit primarily stems from the justified reservations of corporate entities about sharing their network attack data, fearing it might expose significant facets of the organization. We used the dataset DAPT 2020 [6], the dataset was created specifically to train and investigate models for APT detection. The temporal compression discussed in Section 4.1 was designed to encapsulate a vast range of activities and potential anomalies within a manageable time frame for research purposes. This dataset stands out from all others available, because it has also has the Lateral Movement and Data Exfiltration represented, which is rare and enables us to explore deeper into this two phases. As you can see in Table 5.1 outlines the data gathering from a multi-tenant cloud system with identified and unidentified vulnerabilities.

Table 5.1: Various attack methods were employed in DAPT 2020 [6]

Day	**Activity**	**Tools Used**	**Details**
Day 1, 8:00 AM-6:00 PM	Normal Traffic	ping, dig, GET, POST, curl, browsing, files upload, download	Baseline normal traffic based on user activities.
Day 2, 8:00 AM-6:00 PM	Reconnaissance	nmap, webscarab, sqlmap, dirbuster, nikto, burpsuite, application account discovery tools	Reconnaissance on public network, identification of vulnerabilities, directory structure, weak authentication, and authorization.
Day 3, 8:00 AM-6:00 PM	Foothold Establishment	PHP reverse shell, netcat, SQL vulnerability exploitation (sqlmap), XSS exploitation, authentication bypass, metasploitable	PHP reverse shell via DVWA, file upload, adding of malicious users was performed on badstore.
Day 4, 8:00 AM-6:00 PM	Lateral Movement	Nmap scan on local network, vsftpd 2.3.4 vulnerability, weak ssh authentication, mysql script for CVE-2012-2122, metasploit	Exploration of internal network from compromised VMs (Public VM), and obtaining foothold on critical local systems.
Day 5, 8:00 AM-6:00 PM	Data Exfiltration	Data exfiltration to C&C, SMB vulnerability CVE-2017-7494 used to obtain elevated privileges, Google Drive, PyExfil, ftp, scp	FTP put method from local machine to remote server, wput to remote location using anonymous user, scp large files to remote server, web based uploads to Google Drive.

The dataset was developed using VMWare ESXi servers, which hosted the virtual machines (VMs) featuring various services commonly found in an enterprise cloud network. The Public VM hosted susceptible services, including Mutillidae, Damn Vulnerable Web Application (DVWA), Metasploitable, and BadStore. For monitoring malicious traffic patterns, Snort was employed, a Network-based Intrusion

Detection System (NIDS) [39]. Every service was contained within its distinct Docker container [40]. Meanwhile, the private VM was designated for hosting services such as Samba, a Wordpress website, FTP, MySQL, and Nexus for repository management. Both the private and public VMs were interconnected via a private network. Furthermore, each VM came with packet and log capturing capabilities. For log storage and filtering, a log server was employed based on the ELK stack. Network and host logs were sent to the Log Server at regular intervals using the filebeat agent [6].

5.3 Results with the Experimental Dataset

The results obtained from the application of APTSP to the DAPT 2020 dataset are elucidated. The DAPT 2020 dataset was primarily utilized as the foundation to facilitate the training and testing of the APTSP algorithm. The core objective of the algorithm is to predict the stage of an APT when provided with a particular threat technique. These stages can be classified into four primary categories: Reconnaissance, Establish Foothold, Lateral Movement, and Data Exfiltration. To give a better understanding on the output of APTSP we tested it for various malicious techniques calculating for each one the likelihood of the current and next stages of an APT as you can see in Figures 5.1, 5.2 and 5.3, We depict that the highest probability for the APT current stage and the most likely next stage it will transition into. This is based on the occurrence of specific malicious technique, for instance, when the technique is SQL Injection, there's a higher likelihood of the APT being in the Establish Foothold stage.

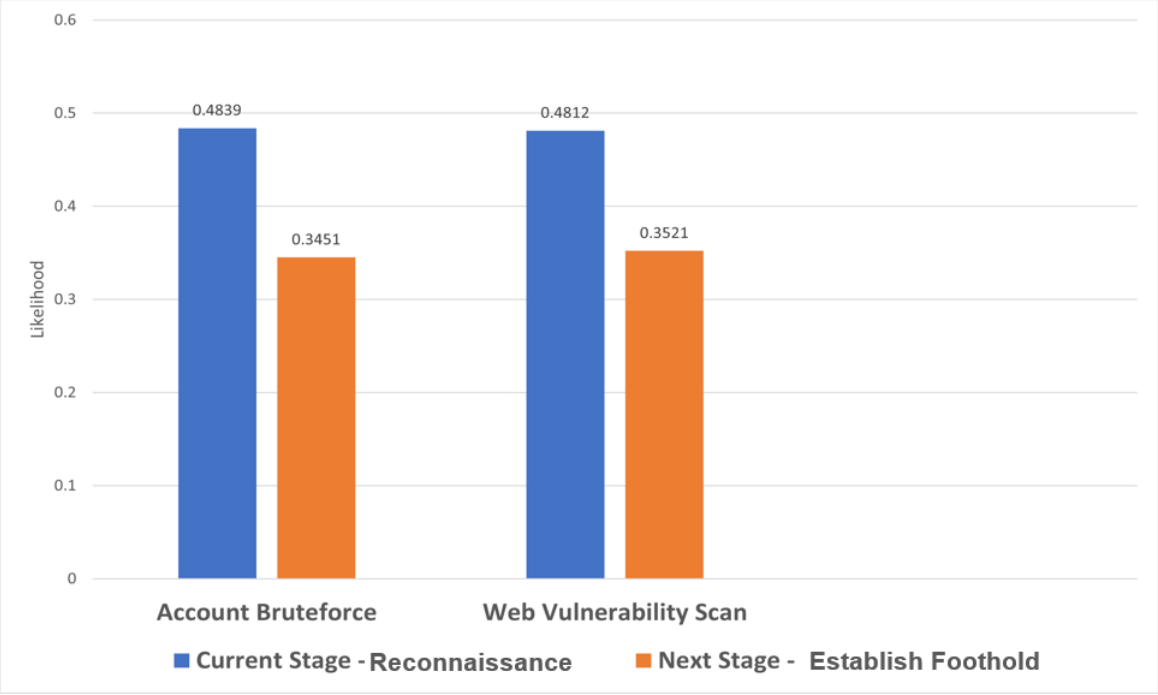


Figure 5.1: *Likelihood based on TTPs for stage APT detection on DAPT 2020 dataset using APTSP*

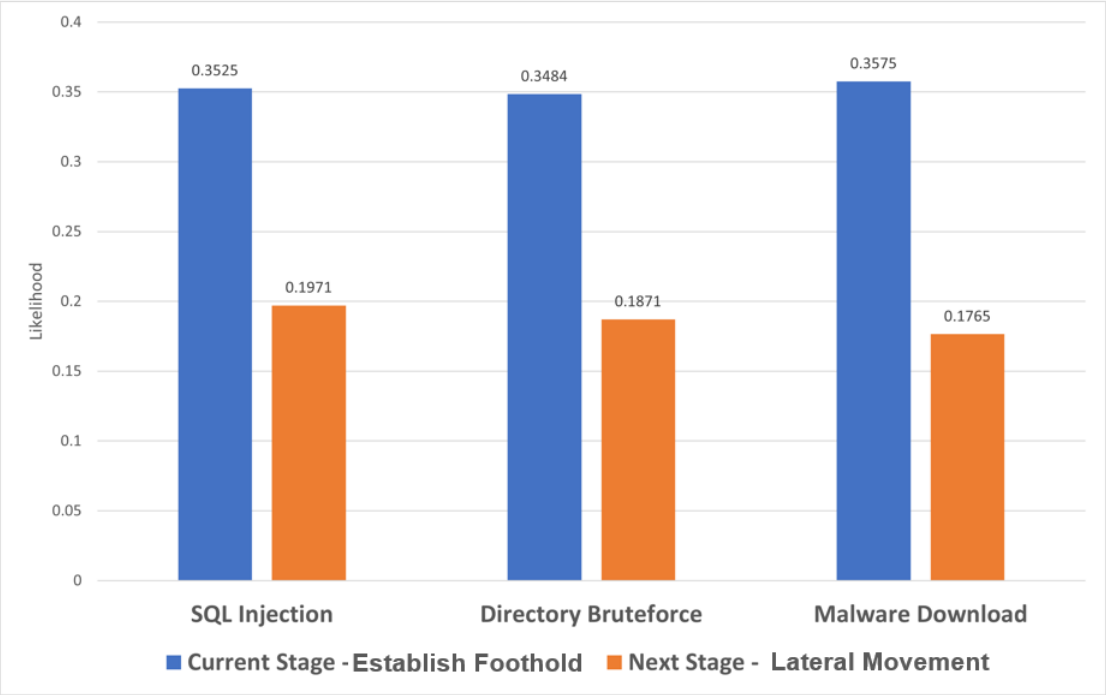


Figure 5.2: *Likelihood based on TTPs for stage APT detection on DAPT 2020 dataset using APTSP*

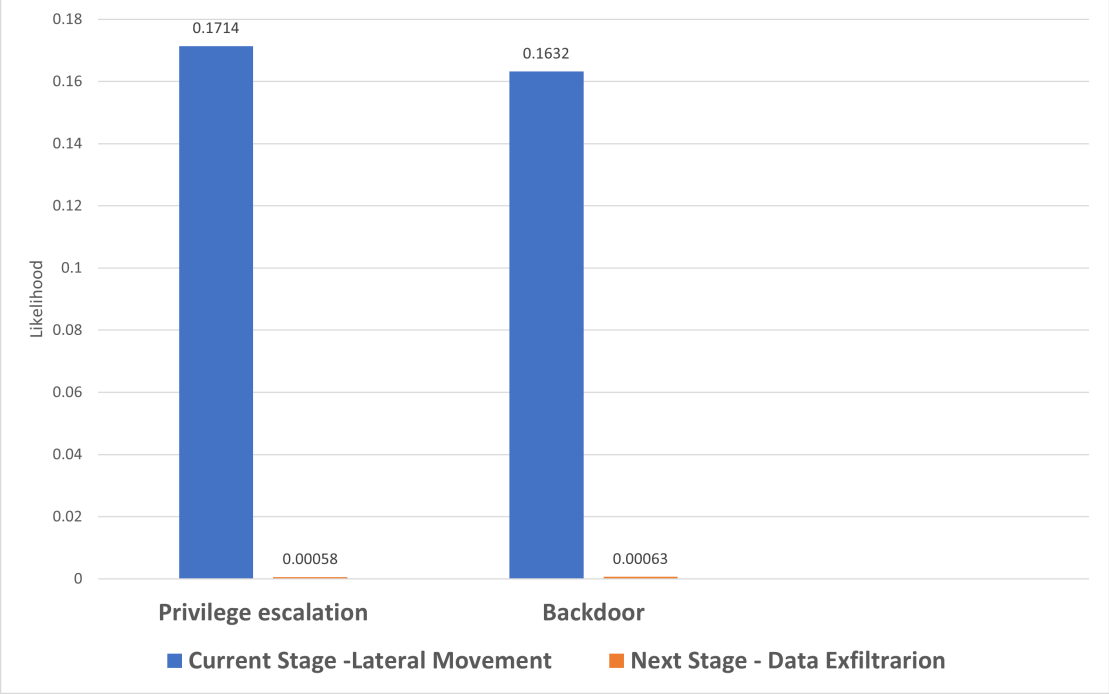


Figure 5.3: *Likelihood based on TTPs for stage APT detection on DAPT 2020 dataset using APTSP*

The Precision-Recall curve depicted in the graph below illustrates the performance of the APTSP algorithm in predicting the various stages of an APT. Each curve corresponds to a specific stage, with

the AUC-PR values denoted in parentheses. It's evident from the graph that the Establish Foothold stage has the highest AUC value of 0.50, indicating relatively better performance in this stage prediction. In contrast, the Data Exfiltration stage yielded an AUC value of 0.0006, suggesting that the algorithm faced challenges predicting this stage accurately, as you can see in Figure 5.4. These results were already to be expected, as the Reconnaissance and Establish Foothold stages are better represented in the dataset and have a significantly higher volume of malicious actions. In contrast, the final two stages of the APT attack are much more discreet and meticulous, making their identification quite challenging.

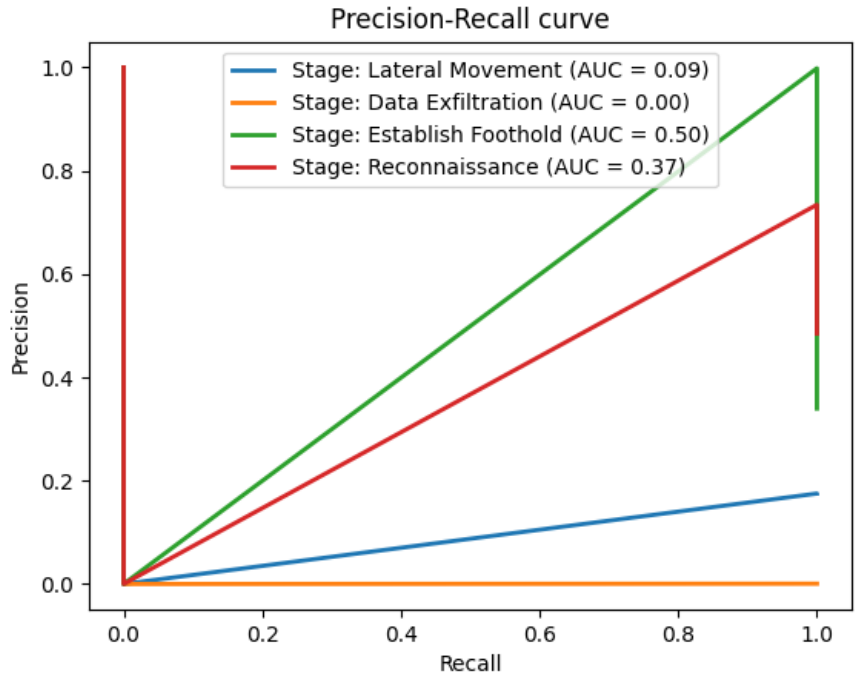


Figure 5.4: *Precision-Recall curve for stage APT detection on DAPT 2020 dataset using APTSP*

To identify a possible APT in this dataset calculated the score for APT41, Target Breach and RSA secureID as explained in Section 4.6. We infer that the most probable APT present is Target Breach given the highest score, leaving APT41 as the second most probable.

Table 5.2: Scores for APT41, Target Breach and RSA secureID

APT41	Target Breach	RSA SecureID
0.2027	0.2412	0.040

To comprehensively assess the performance of the APTSP algorithm, three metrics were computed. Precision measures the accuracy of the positive predictions. A higher precision means that more of the predicted APT stages were correct. Recall represents the proportion of actual positives that were correctly identified. A higher recall signifies that most of the actual APT stages were predicted, and we

were able to avoid False Negatives. F-score summarizes the performance, providing a balance between the two metrics, accomplishing an good value for an unbalanced dataset depict in Table 5.3.

Table 5.3: Summary of the results in the Experimental dataset DAPT 2020

Precision	Recall	F-Score
0.5846	1	0.7378

5.4 Comparison with Different Approaches

This section compares APTSP implementation with One-Class Support Vector Machines (1-SVM), Stacked Auto Encoder (SAE) and Stacked Auto Encoder with Long Short-Term Memory (LSTMSAE), solutions that were presented and published in article [6]. We selected these ones because they were the most recent and available work tested with this dataset, and no other implementations were available. We used their results to measure APTSP performances in comparison.

5.4.1 Comparison

All the three approaches have semi-supervised learning approaches, utilizing these models for anomaly detection involves an initial training phase where the models are exposed to and learn from normal network traffic data. During the subsequent testing phase, when presented with an input, they are subject to a meticulous assessment by passing it through the auto-encoder architecture. This process entails evaluating the normalized reconstruction error, which serves as a critical metric. If the computed reconstruction error surpasses a predefined threshold, the incoming data packet is promptly classified as an anomalous traffic instance. Conversely, if the reconstruction error remains below this threshold, the data packet is confidently categorized as representing normal network traffic.

We compared the evaluations obtained for each stage of an APT, Reconnaissance, Establish Foothold, Lateral Movement and finally Data Exfiltration. In all cases, it was it was possible to obtain a better result with APTSP implementation compared to the other approaches. You can clearly see that APTSP shows better values for the all stages of an APT, as you can see in Figures 5.5, 5.6, 5.7 and 5.8. In the following section, we will explore, elaborate and discuss on each of the associated methods.

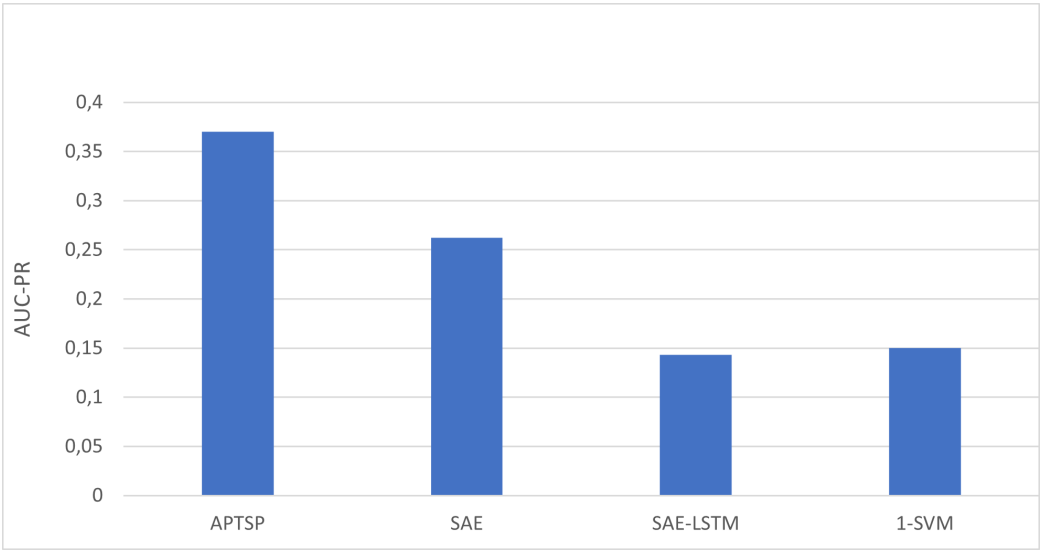


Figure 5.5: *Reconnaissance AUC-ROC values comparison of APTSP, SAE, SAE-LTSM and 1-SVM for DAPT 2020 dataset*

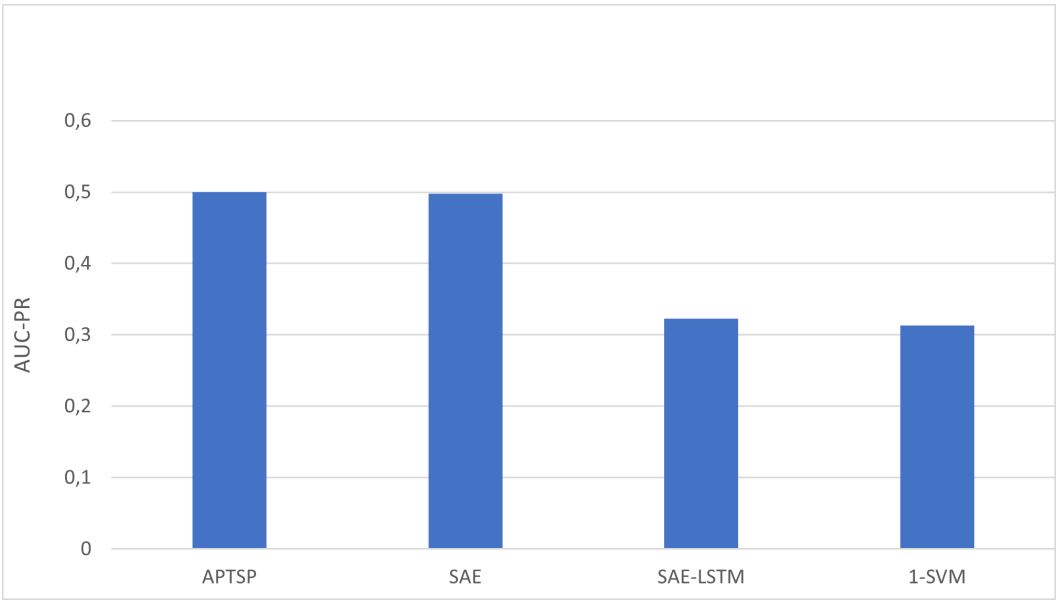


Figure 5.6: *Establish Foothold AUC-ROC values comparison of APTSP, SAE, SAE-LTSM and 1-SVM for DAPT 2020 dataset*

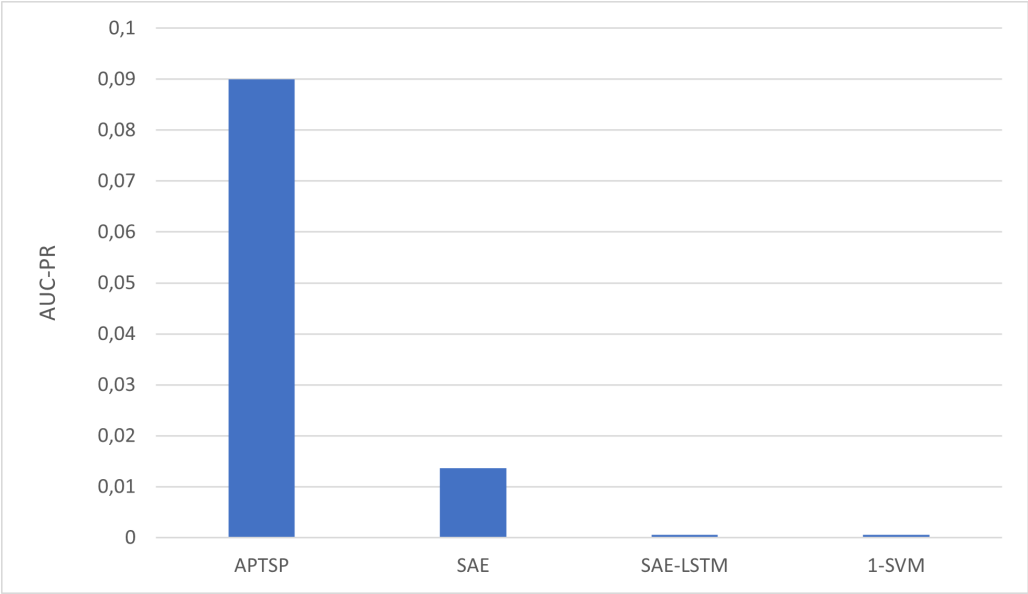


Figure 5.7: *Lateral Movement AUC-ROC values comparison of APTSP, SAE, SAE-LTSM and 1-SVM for DAPT 2020 dataset*

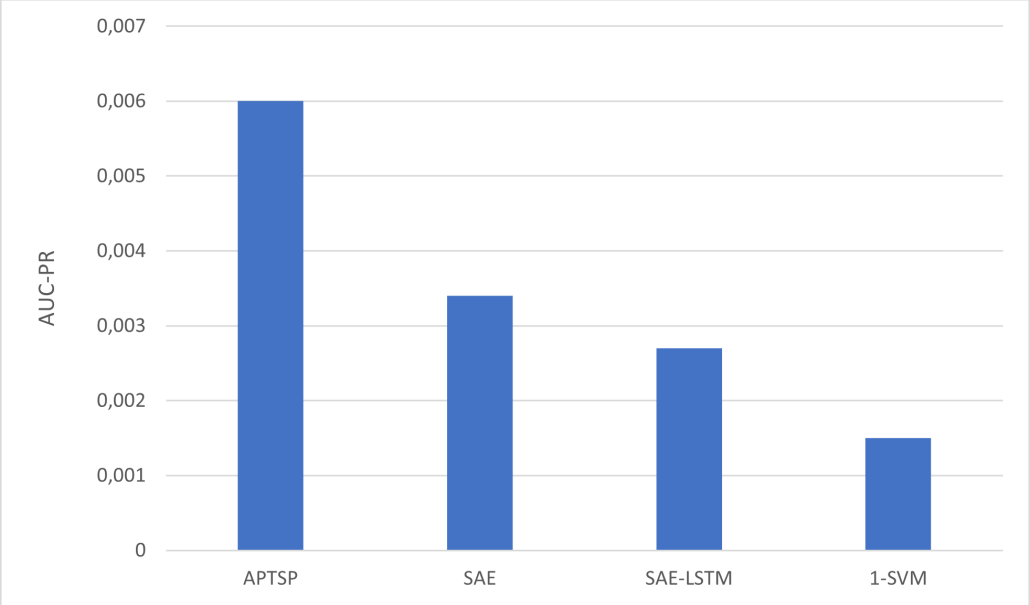


Figure 5.8: *Data Exfiltration AUC-ROC values comparison of APTSP, SAE, SAE-LTSM and 1-SVM for DAPT 2020 dataset*

Uniformly throughout evaluations conducted on DAPT 2020, all machine learning models exhibited subpar performance, whether in the context of lateral movement or data exfiltration. This indicates that the distribution of contextualized attack vectors in these stages closely resembles that of regular network traffic. Additionally, the sustained low performance levels of all unsupervised learning algorithms during these APT phases indicate that the attack methods utilized in this dataset possess a high degree of stealth

and present significant challenges for detection by current classifiers.

5.4.2 One-Class Support Vector Machines

One-Class Support Vector Machines (1-SVM) are renowned for their effectiveness in contexts characterized by an overwhelming majority of standard traffic with a minor proportion of anomalous traffic data. The underlying principle involves training the model predominantly on labeled examples from the class with abundant data. For this study, the 1-SVM model was educated using the prevalent standard network traffic data. Subsequently, during the testing phase, a specified treshold was employed to determine if a reconstruction error is significant enough to be categorized as an anomaly. In Figures 5.6, 5.7 and 5.8 we can see that 1-SVM preformed the worst in all stages prediction. We note that the approach exhibited its highest performance during the establish foothold stage, as anticipated. This aligns with our expectations, as the establishment of a foothold is a critical phase common to all APT attacks.The PR curves demonstrate a significant inadequacy in effectively identifying attacks during the last two stages of APTs. This indicates that these attacks operated in an exceptionally covert manner, closely resembling regular network traffic. It becomes evident that detecting these phases of an APT attack reliably using current anomaly detection models is a challenging task. APTSP was able to outpreform 1-SVM in all metrics.

5.4.3 The Stacked Auto Encoder

The Stacked Auto Encoder (SAE) is a specialized variant of feed-forward neural networks designed to identify a concise latent-space representation of the input, optimizing it for reconstruction purposes. Typically, autoencoders comprise a singular hidden layer where compression transpires between the input and this hidden layer, and reconstruction takes place between the hidden and the output layer. Conversely, in Stacked Auto-encoders, this process of compression and subsequent reconstruction is achieved through a deep neural network instead of a lone non-linear layer. During its training phase, the SAE output is engineered to emulate the input, resulting in a loss function that endeavors to reduce the disparity between the initial input and the reconstructed output. Our method involved priming an SAE on standard traffic data and then testing it on both standard and anomalous data sets. The hypothesis is that while the SAE proficiently reconstructs standard data, it struggles with abnormal data, resulting in an elevated reconstruction error. This property enables classifiers to easily identify anomalous network traffic data by contrasting the reconstruction error against a predetermined threshold. Figures 5.6, 5.7 and 5.8 depict that AUC-PR values were low for all algorithms in the Reconnaissance stage. This implies that the distribution of the attack during the reconnaissance phase was notably scarce, making it challenging to identify. Despite this factor, APTSP was still able to preform better than the others.

5.4.4 Stacked Auto Encoder with Long Short-Term Memory

Stacked Auto Encoder with Long Short-Term Memory (LSTM-SAE) introduces an evolution to the conventional stacked auto-encoders. Though many research initiatives have utilized SAEs, these are intrinsically limited in detecting contextual anomalies - a facet crucial for understanding APTs. This limitation arises since an SAE's input layers only accommodate a singular network packet. To counteract this, this approach integrates a stacked auto-encoder equipped with LSTM cells, replacing the traditional hidden layer cells of SAEs. LSTMs, having proven their mettle in time-series analysis, enable the incorporation of data spanning numerous temporal steps. This modified entity, termed LSTM-SAE, facilitates the compression of network traffic packets over consecutive time intervals and their subsequent reconstruction. Utilizing the familiar strategy of training on prevalent standard data and testing on both malicious and standard data sets, we can identify attacks executed in phases or distributed over several packets. In figures 5.6, 5.7 and 5.8 we can see that LSTM-SAE performed poorly compared to SAE in all stages of APT.suggesting that the dataset anomalies are sparsely distributed. Finally and once more, our novel approach had a better performance for all phases of APT.

6

Conclusion

Despite Advanced Persistent Threats being one of the most challenging types of attacks to defend against, we introduce APTSP as a tool that should be integrated with an IDS/SIEM to assist defenders in understanding the objectives of malicious agents, thereby enabling them to defend against and prevent future actions. Our framework is fully capable of predicting the most likely stage of an APT and simultaneously predict the most likely APT group responsible for the attack campaign.

APTSP was able to perform better in all metrics compared to other approaches as 1-SVM, SAE and LSTMSAE. Proving that our model has brought added value as we obtained better results, especially in the final stages of an APT where the predictions are more challenging. We aimed to compare our results with other works, but we were hindered due to the lack of published results for comparison and the significant scarcity of datasets in this area.

We were able to conclude that the APTSP achieved very good results for the initial stages of the APT, specifically Reconnaissance and Establish Foothold, even when compared to results obtained by other algorithms. This significantly contributed to progress in this area. Although our approach covering the Lateral Movement and Data Exfiltration stages, we conclude that their values are lower compared to the other stages. This results from the fact that these stages are highly stealthy and difficult to detect. Having only one dataset for testing our tool, specifically the DAPT2020, also contributed to these lower results.

Future endeavors will encompass an expansion of the database associated with APT identification, aiming to enhance the breadth and depth of available information. This expansion seeks to heighten the probability of effectively identifying the perpetrators behind APT attacks, thereby bolstering the capabilities for attribution and proactive measures against potential threats.

An additional area for improvement lies in the capacity to train the models using a more diverse array of datasets featuring APTs. This approach seeks to fortify the model's adaptability and preparedness to effectively recognize and respond to a broader spectrum of attack variations. By exposing the models to a wider range of APT scenarios during training, it aims to enhance their robustness and efficacy in identifying and mitigating potential threats across various attack types and tactics. However, datasets in this area continue to be very scarce, further aggravating the challenge and complexity of the problem. Therefore, we also propose the emulation of various datasets with known APTs to train more comprehensive models capable of achieving a higher successful prediction rate.

Bibliography

[1] B. Strom, A. Applebaum, D. Miller, K. Nickels, A. Pennington, and C. Thomas, " MITRE ATT&CK: Design and philosophy," *Technical Report*, Revised March 2020.

[2] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. N. Venkatakrishnan, "HOLMES: Real-time APT detection through correlation of suspicious information flows," *2019 IEEE Symposium on Security and Privacy (SP)*, Jan 2019.

[3] A. Alshamrani, S. Myneni, A. Chowdhary, and D. Huang, "A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities," *IEEE Communications Surveys & Tutorials*, vol. PP, pp. 1–1, 01 2019.

[4] R. Al-Shaer, J. M. Spring, and E. Christou, "Learning the associations of MITRE ATT & CK adversarial techniques," in *2020 IEEE Conference on Communications and Network Security (CNS)*. Avignon, France: IEEE, Jun 2020, p. 1–9.

[5] I. Ghafir, K. G. Kyriakopoulos, S. Lambotharan, F. J. Aparicio-Navarro, B. Assadhan, H. Binsalleeh, and D. M. Diab, "Hidden markov models and alert correlations for the prediction of advanced persistent threats," *IEEE Access*, vol. 7, pp. 99 508–99 520, 2019.

[6] S. Myneni, A. Chowdhary, A. Sabur, S. Sengupta, G. Agrawal, D. Huang, and M. Kang, "Dapt 2020-constructing a benchmark dataset for advanced persistent threats," in *International Workshop on Deployable Machine Learning for Security Defense*. Springer, Cham, 2020, pp. 138–163.

[7] O. Grigorescu, A. Nica, M. Dascalu, and R. Rughinis, "CVE2ATT&CK: BERT-based mapping of CVEs to MITRE ATT&CK techniques," *Algorithms*, vol. 15, no. 9, p. 314, Aug 2022.

[8] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.

[9] L. Yang, P. Li, X. Yang, and Y. Tang, "Security evaluation of the cyber networks under advanced persistent threats," *IEEE Access*, vol. 5, pp. 20 111–20 123, 2017.

[10] T. M. Corporation, "The MITRE corporation our history," last accessed: Jan 2, 2022. [Online]. Available: https://www.mitre.org/who-we-are/our-story

[11] M. Corporation, "The MITRE corporation focus areas," last accessed: Jan 2, 2022. [Online]. Available: https://www.mitre.org/focus-areas

[12] T. M. Corporation, "The MITRE corporation matrix - enterprise," last accessed: Jan 2, 2022. [Online]. Available: https://cwe.mitre.org/about/index.html

[13] J. L. Gjerstad, "Generating labelled network datasets of APT with the MITRE CALDERA framework," Master thesis, University of Oslo, 2022.

[14] H. M. Cisa, "Best practices for MITRE ATT&CK mapping," June 2021.

[15] J. Chen, C. Su, K.-H. Yeh, and M. Yung, "Special issue on advanced persistent threat," *Future Generation Computer Systems*, vol. 79, p. 243–246, 2018.

[16] S. Singh, P. K. Sharma, S. Y. Moon, D. Moon, and J. H. Park, "A comprehensive study on APT attacks and countermeasures for future networks and communications: challenges and solutions," vol. 75, p. 4543–4574, Aug 2019.

[17] D. McWhorter, "APT1: Exposing one of China's Cyber Espionage Units," *Mandiant*, vol. 7908, 2013.

[18] X. Wang, W. Huang, and J. Liu, "A review of machine learning techniques applied in malware detection," in *Machine Learning and Cyber Security*. Springer, Cham, 2016, pp. 1–33.

[19] M. H. Bhuyan, S. Misra, and S. Dash, "Anomaly based intrusion detection system using machine learning techniques," in *Advanced Computing, Networking, and Informatics (ICACNI), 2014 International Conference on*. IEEE, 2014, pp. 401–407.

[20] L. Tong, X. Jin, and G. Qi, "A machine learning approach to predict and prevent cyber attacks," in *International Conference on Computational Science and Its Applications*. Springer, Cham, 2017, pp. 37–49.

[21] Y. Mehmood, U. Habiba, M. Shibli, and R. Masood, "Intrusion detection system in cloud computing: Challenges and opportunities," in *IEEE 2nd Nat. Conf. Inf. Assurance (NCIA)*, 2013, pp. 59–66.

[22] S. Dua and X. Du, *Data Mining and Machine Learning in Cybersecurity*. CRC Press, 2016.

[23] H. Nath and B. Mehtre, "Static malware analysis using machine learning methods," in *Proc. SNDS*, 2014, pp. 440–450.

[24] C. C. Aggarwal, "An introduction to outlier analysis," in *Outlier Analysis*. Springer International Publishing, 2017, pp. 1–34.

[25] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[26] P. Casas, J. Mazel, and P. Owezarski, "Unsupervised network intrusion detection systems: Detecting the unknown without knowledge," *Computer Communications*, vol. 35, no. 7, pp. 772–783, 2012.

[27] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *Journal of Machine Learning Research*, vol. 11, pp. 2837–2854, Dec 2010.

[28] S. Shin, S. Lee, H. Kim, and S. Kim, "Advanced probabilistic approach for network intrusion forecasting and detection," *Expert Systems with Applications*, vol. 40, p. 315–322, 01 2013.

[29] A. Allahdadi, R. Morla, and J. S. Cardoso, "Outlier detection in 802.11 wireless access points using hidden markov models," in *7th IFIP Wireless and Mobile Networking Conference (WMNC)*, 2014, pp. 1–8.

[30] G. Brogi and E. D. Bernardino, "Hidden markov models for advanced persistent threats," *International Journal of Security and Networks*, vol. 14, no. 4, p. 181, 2019.

[31] H. A. Kholidy, A. Erradi, S. Abdelwahed, and A. Azab, "A finite state hidden markov model for predicting multistage attacks in cloud systems," in *IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC)*, 2014, pp. 14–19.

[32] C. C. Aggarwal, *Outlier Analysis*. Springer International Publishing, 2017.

[33] MITRE ATTCK®. Apt41, wicked panda, group g0096 — mitre attck®. https://attack.mitre.org/groups/G0096/. Acedido: 29 de Setembro de 2023.

[34] R. Wagner, M. Fredrikson, and D. Garlan, "An advanced persistent threat exemplar," *MONTH*.

[35] J. Leyden, "Rsa explains how attackers breached its systems," https://www.theregister.com/2011/04/04/rsa_hack_howdunnit/, 2011.

[36] F. Menczer, S. Fortunato, and C. A. Davis, *Python Tutorial*, 2020. [Online]. Available: https://cambridgeuniversitypress.github.io/FirstCourseNetworkScience/

[37] W. McKinney, "Data structures for statistical computing in python," in *Python in Science Conference*, Austin, Texas, 2010, pp. 56–61.

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *The Journal of Machine Learning Research*, vol. 12, 2011.

[39] M. Roesch *et al.*, "Snort: Lightweight intrusion detection for networks," in *Lisa*, vol. 99, 1999, pp. 229–238.

[40] D. Merkel, "Docker: Lightweight linux containers for consistent development and deployment," *Linux J.*, vol. 2014, no. 239, mar 2014.