FRAMEWORK AND STRATEGIES TO MITIGATE ADVANCED PERSISTENT THREATS (APT)

by

Rudra Prasad Baksi

August 31, 2022

A dissertation submitted to the
Faculty of the Graduate School of
the University at Buffalo, the State University of New York
in partial fulfilment of the requirements for the
degree of

Doctor of Philosophy

Department of Computer Science and Engineering

Dedication

To my family, friends and mentors whose support and guidance helped me to pursue my dreams.

Acknowledgments

I would like to express my deepest gratitude to Dr. Shambhu Upadhyaya, my mentor, for his unlimited guidance, care, and support throughout my PhD studies at the University at Buffalo, SUNY. Dr. Upadhyaya has shown me how to conduct groundbreaking research. He not only motivated me to be a better researcher but also a better human being.

I wish to thank Dr. Varun Chandola for his insights on doing high-quality research in cybersecurity using artificial intelligence, and Dr. Wenyao Xu for his invaluable input and support in hardware-assisted security. I would also like to thank Dr. Jun Zhuang for his guidance and encouragement in pursing research in the field of Game Theory.

I want to thank all my former and present colleagues, especially Dr. Gokhan Kul, Dr. Ridwan Al Aziz, and Mr. Vishwas Nalka. I feel extremely lucky to have worked with them during my PhD studies. They made my life at UB more colorful and have enriched me with a lot of precious memories.

Last, but not least, I would like to thank my family and friends for their unconditional love and support.

Table of Contents

List of Tables

List of Figures

Abstract

Advanced Persistent Threats (APT) are not only a matter of concern for the government organizations but also to the industries as well. Recent studies show that many companies have suffered financial damage at the face of an attack by an APT and sustained damage to their reputation and brand value. Detecting the attack and building a defense system against APT is difficult due to the fact that these attacks are very stealthy and targeted. This dissertation lays the path to defense against APTs through identification, detection, deception, and attack mitigation. To begin with, this dissertation presents a holistic approach via a parameterized model to identify an APT and then to review and assess the vulnerabilities, attacker resources, and probable targets. The attacks mounted by APT groups are highly diverse and sophisticated in nature and can render traditional signature based intrusion detection systems ineffective. This necessitates the development of behavior oriented defense mechanisms. Therefore, following the identification of newer features being attributed to APTs, we developed intrusion detection systems (IDS) using Hidden Markov Model (HMM), machine learning (ML) models, and natural language processing (NLP). The IDSes we designed for APT type ransomware needed another layer of protection. This layer of protection was ensured through a deception architecture. In this dissertation, we put-forward a framework called Kidemonas to tackle generic APT attacks with the use of deception. We used a commercial-off-the-shelf (CoTS) hardware component called trusted platform module (TPM) for designing Kidemonas. The idea is to run a generic APT detection system in an isolated environment outside the purview of the attacker. In order to give a layer of protection to the HMM based IDS, we designed a deception based countermeasure called Decepticon. The intrusion detection system and the deception architecture were designed as a defense against APT type malware. But the systems which do not have these defense features, and/or the systems in which these defense features are defeated by the APT type malware, are put to great risk. To counter this problem, we explored the solutions using game theoretic analysis. We analyzed the threat scenario of non-APT type ransomware using a sequential game model. We introduced two parameters which would help the defender in making an informed decision when under attack. We then extended the concept of game theory for more sophisticated APT type ransomware. We designed a more elaborate sequential game model for multi-stage advanced ransomware attacks, analyzed the threat scenario and traced the optimal strategies of the attacker for different conditions while considering the attacker's perception of the defense architecture in the system and existence of a contingency plan of attack on the part of the attacker. We came up with equilibrium conditions to maximize the outcome and minimize the losses of the defender with and/or without the defense features. We also put forward an algorithm, which would help the defender to reach the equilibrium state for a given set of conditions. This helps in both preparedness and mitigation of the sophisticated APT type attacks.

Chapter 1

Introduction

Advanced Persistent Threats (APT) are a form of quiet invaders [1] and are a lingering threat to industries and government organizations. They silently perform reconnaissance, quietly invade, and keep a communication channel open in order to communicate with the command and control (C&C) centers. The attackers control the behavior of the malware from the C&C centers. APTs carry out *targeted attacks* to achieve their goal. They are quite persistent in their efforts of achieving the goals and in doing so they might come with a *contingency plan* to which they may resort to upon discovery [2]. Such a type of attack has become prevalent and frequent, owing to the fact that malware-as-a-service (MaaS) are readily available, which provide the attackers with the necessary framework and infrastructure to create attacks [3], [4]. APTs come in different forms and formats. In this paper we focus on the detection and mitigation of a ransomware that qualifies as an APT [2].

According to FireEye, 4,192 attacks were detected in 2013, which were mounted by groups that can confidently be classified as APT groups [5]. They were also able to detect 17,995 different infections by APT groups. The attacks thereafter have been increasing by leaps-and-bounds. RSA Security LLC suffered financial losses of about $66.3 Million when it became a victim of an APT attack [6]. According to a study by Ponemon Institute, the average financial losses suffered by a company owing to the damaged reputation after an APT amounts to about $9.4 Million [7]. WannaCry, Petya and NotPetya are ransomware campaigns that graduated to become APTs and collected huge amounts of ransom causing considerable financial losses to the victims [2]. WannaCry collected ransom in BitCoins. According to published reports, between May 12, 2017 and May 17, 2017, the attackers collected $75,000 to $80,000 in ransoms [8], [9]. With time the cost of financial damage suffered by the companies is expected to go even higher. If the target of attack is a government agency, the damage could be beyond mere financial losses; the attacks might even threaten national security.

These aforementioned factors and incidents outline a great threat to the critical infrastructure as a whole, be it government or industry. The problems are intense and the attacks are adaptive in nature, requiring a holistic approach to address them. However, it is not necessary to put the entire defense framework into the same defense mode every time the system comes under attack because deploying a sophisticated defense mechanism indiscreetly to fend off attacks will severely affect performance and degrade the quality of service (QoS). A better approach is to deploy the most sophisticated countermeasure against the most severe form of attack. Less sophisticated countermeasures taking care of the less severe attacks would not only be economical but also might help in preserving a good balance between security, performance, and the QoS of the system. In the same vein, system security through different forms of *information isolation* has been studied for quite sometime [10]. Isolation can be achieved through software or hardware [11]. But with advanced attacks from

APT groups which are highly adaptive in nature, they have been successful in attacking physically isolated systems as well. One such example is the Stuxnet campaign that took place in the Iranian nuclear facility [12], [13], [14]. Therefore, a need for a new form of defensive strategy arose. Researchers have looked into various approaches to repel highly sophisticated attacks. One of the approaches is the use of *deception* as a defense tool.

This dissertation investigates in detail the problems due to APT attacks, and the development of a framework and strategies to mitigate the effects of APT attack. The dissertation is organized as follows. Chapter 2 puts forward the problem this dissertation is concerned about. In Chapter 3 APT definitions, AI models, and different techniques traditionally used to defend and mitigate APT attacks are examined. Chapter 4 discusses the newer attributes of APT attacks which have been identified in this research. Chapter 5 presents several intelligent intrusion detection systems (IDS) designed using Hidden Markov Model (HMM), machine learning (ML) models, and natural language processing (NLP) models. Chapter 6 presents deception frameworks to tackle the APT problem. The idea is to provide an added layer of security to the IDSes developed in Chapter 5. But there can be systems which may or may not have an APT detection system and/or their APT detection system has been defeated by a sophisticated malware. The questions then arise "What to do?", "When to do?", and "How to do?" with regard to mitigating APT attacks. To answer these questions, Chapter 7 analyzes both APT and non-APT variants of ransomware to develop optimal strategies to mitigate the attack. Finally, in Chapter 8 we conclude this dissertation and pave the way for future research. Chapter 9 lists the papers published as the outcome of this research.

Chapter 2

Problem Statement

The aim of this dissertation is to identify attacks resulting from Advanced Persistent Threats (APT) and provide a framework and strategies to mitigate APT type Ransomware. By seeking answers to the following questions, the purpose and the solution schemes developed in this dissertation can be outlined.

- *What are APTs?*
- *How are APTs identified traditionally?*
- *In the ever-changing scenario of attack landscape by sophisticated attackers, how can one identify state-of-the-art attack by APT groups?*
- *How to reduce the computational load of running an APT detection system?*
- *APTs infiltrate stealthily and mount sophisticated attacks. Can deception be used as a defensive tool against an APT attack?*
- *Can deception be implemented in a cost effective manner?*
- *Other than having a framework to counter APTs, how can one make informed decision to defend against them?*

- *How can one devise an optimum strategy to defend against APT attacks?*
- *A defender is always faced with the questions, "What to do?", "When to do?", and "How to do?" while an attack is on-going. How to answer these questions?*

2.1 Threat Model

Advanced Persistent Threats (APT) are putting systems at great risk. They stealthily infiltrate the system, perform reconnaissance missions to gather information, gain access to critical infrastructure and mount an attack that poses grave danger to the functioning of the entire system. APTs often carry out targeted attacks to achieve their goal. The APT campaigns are typically carried out against government agencies, military systems and commercial entities. The cost of damage is expected to go higher with time, due to the fact that the attacks from the APTs are becoming deeper and adaptive in nature. There is no formal model in the literature to classify a threat as an APT or not; security experts often apply some ad hoc rules after analyzing the impact of the attack. Without a proper threat or attack model, the defense mechanism also becomes unstructured and ineffective.

Ransomware are a type of malware which infiltrate a system and hold critical data for a ransom. Primarily there are three simpler types of ransomware, namely *the locker*, *the crypto* and *the hybrid* [15]. The locker variant of the ransomware locks the entire system and denies the user access to the system. The crypto form of the malware, targets specific files and/or folders and encrypts them, thereby denying the user any access to those encrypted resources. The hybrid version of ransomware possesses the capabilities of both types of ransomware. Ransomware variants created by APT groups are more advanced and can cause damage beyond mere ransomware type attacks. A formal model is required to classify these APT attacks to build a strong defense framework.

2.2 APT Defense: Detection and Deception

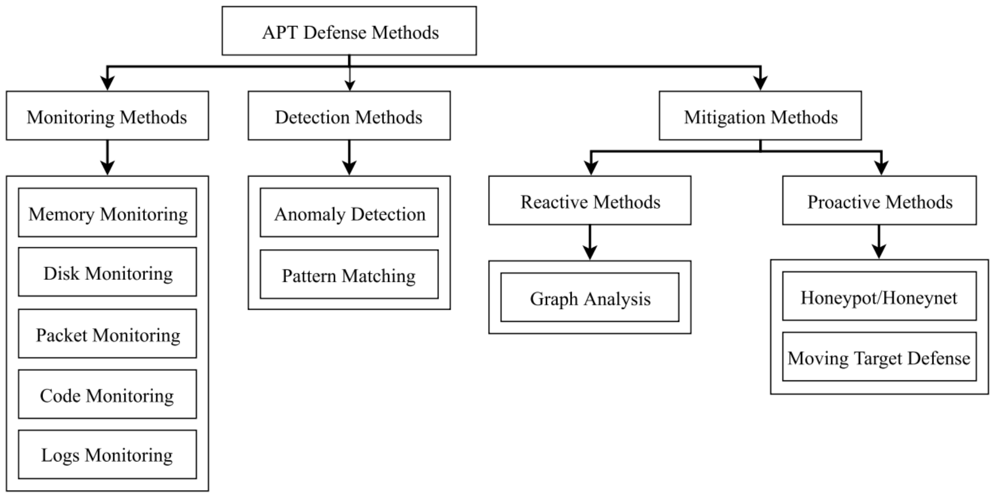


Figure 2.1: Traditional APT Defense Methods

Figure 2.1 shows different methods for defense against APT attacks [16]. These methods require considerable computational capability. Traditionally APT detection systems run using super-computers. This poses an important problem. To tackle newer and more sophisticated attacks, not many intelligent and distributed systems are available so that APT detection systems could be run securely on lesser powerful computers.

2.3 Mitigation

One of the most important factors in defense against APTs is to make proper decisions at the proper time. Answering the questions “What to do?”, “When to do?”, and “How to do?” suitably would help the defender devise a mitigation plan when faced with an APT attack.

2.4 Summary of Contributions

- We have identified several new attributes of APTs to address the state-of-the-art attacks from well-known APT groups. This helps to characterize more sophisticated attacks. An updated feature set helps to build a stronger defense system. This addresses the problem of threat modeling.
- We designed an architecture, Kidemonas, a generic framework to silently detect APTs and surreptitiously report the intrusion to the user. To make the architecture cost effective, commercial-off-the-shelf (COTS) hardware components are used. Kidemonas is a distributed framework to detect generic APT. This reduces the computational load by putting different APT detection system in different nodes of the framework.
- In the aforementioned distributed framework we needed an APT detection system. Therefore, we designed a Hidden Markov Model (HMM) based intrusion detection system (IDS) to detect APT type ransomware. This can be launched from one of the nodes of the framework. If any APT type ransomware is detected at the node, the information is shared with the other nodes of the framework. This reduces the load of computation on other nodes.

- We integrated the HMM based IDS inside Kidemonas along with the smart-box concept by [17] to forward a deception based countermeasure called Decepticon. This addresses the problem of detection and deception as a defense system with respect to APT type ransomware which can be handled by lesser powerful computers.

- We also designed a classifier based IDS using machine learning (ML) and natural language processing (NLP) models, viz., Naive Bayes' Classifier (NBC), gradient boosting (GB) decision trees, random forest (RF), logistic regression (LR), support vector machine (SVM) and BERT.

- We ran several APT type ransomware, viz., Darkside, REvil, BlackMatter, BlackByte, and Diavol, and created a dataset of system call logs. We used that dataset to train, validate, and test our classifier base IDS. This caters to the need of an intelligent detection system for APT type ransomware.

- To answer the questions "What to do?", "When to do?", and "How to do?" competently, we use game theoretic analysis of the attack-defense scenario with respect to APT type ransomware. This would address the final problem of making informed decisions for the purpose of designing an optimum strategy to mitigate APT type ransomware attacks.

- We analyzed the threat scenario of non-APT type ransomware using a sequential game model. We introduced two parameters which would help the defender in making an informed decision when under attack.

- We then extended the concept of game theory for more sophisticated APT type ransomware. We designed a more elaborate sequential game model for multi-stage advanced ransomware attacks, analyzed the threat scenario and traced the optimal strategies of the attacker for different conditions. We came up with equilibrium conditions to maximize the outcome and minimize the losses of the defender with and/or without the defense features.

Chapter 3

Preliminaries and Related Work

"The most important factor in determining whether readers can understand a text is how much relevant vocabulary or background knowledge they have."
– Natalie Wexler,
The Knowledge Gap

In this chapter, we provide relevant background to our research area and describe the related work by other researchers in this domain.

3.1 Advanced Persistent Threats

Malware created by the APT groups typically do not carry out the attacks in a single stage. The "Cyber Kill Chain" framework developed by Lockheed Martin describes an APT through a seven stage life cycle [18]. The model describes the beginning of the attack through a *reconnaissance* phase wherein the malware gathers information about the system. This is followed by the *weaponization* phase, thereupon creating a remote access malware that can be controlled by the attacker. The *delivery* phase denotes the intrusion of the malware into the system. In the *exploitation* phase, the malware exploits the vulnerabilities that exist in the system. The *installation* phase signifies the escalation of privileges on the part of the malware and the installation of back-doors to maintain communication with the command and control (C&C) centers to receive further instructions. The *command and control* phase implies the access of the target system gained by the attackers from the C&C centers. Finally, in the *actions on objective* phase, the intruder mounts the final assault on the system. LogRythm describes an APT through a five stage life cycle [19]. Lancaster University describes APT through a three stage life cycle [20].

3.2 Ransomware

A malware is a software program which is designed with malicious intent to cause harm to the victim. When the intent of a malware is monetary gain by *hijacking* victim's resources for a ransom, it is called a ransomware. Ransomware are a type of malware which infiltrate a system and hold critical data for a ransom. Depending upon the nature and level of sophistication, a ransomware can be of an APT type or of a basic nature. Primarily there are three simpler types of ransomware, viz. *the locker*, *the crypto* and *the hybrid* [15]. The locker variant of the ransomware locks the entire system and denies the user access to the system. The crypto form of the malware targets specific files and/or folders and encrypts them, thereby denying the user any access to those encrypted resources. The hybrid version of ransomware possesses the capabilities of both types of ransomware. It can encrypt and lock targeted resources and/or the entire system. But there can be a variant of ransomware which is a more advanced form of malware. In addition to possessing the features of a basic ransomware, they are more sophisticated by having a contingency plan of attack on being discovered [2]. They also perform the attack through multiple stages and generally are controlled by the attackers from the C&C centers. They qualify as APTs. Recently, an Australian beverage company and educational institutions in India became victims of ransomware attack [21], [22], [23]. Off late, the malware WannaCash is also causing trouble to the cyber-world [24]. Another example of a recent attack is the one on Indian nuclear power plants causing significant data breaches [25]. Zakaria et al. [15] investigated the rise in spread of ransomware, and laid down the main areas for research concerning ransomware starting with the detection from indicators of compromise (IoC), signatures of the malware and analysis of network traffic. This is followed by the two other areas of research identified by the authors, which are recovery from the attack and prevention from future attack. In our research, we investigate the strategies of the attacker as well as the defender, and in doing so we dive into the research territories of "recovery from attack" and "prevention of future attacks" mentioned in [15].

3.2.1 APT type Ransomware

The threat model considered in our research is both basic and advanced type of ransomware but we are focusing primarily on the latter. The APT variants are generally perpetrated by nation state actors with huge amount of resources at their disposal [2]. Following are some of the APT type ransomware used in our research. BlackByte, an APT type ransomware, has two modes of attack. It can either attack directly, or offer services as ransomware-as-a-service (RaaS) [26]. It exploits the ProxyShell vulnerabilities that exists in Microsoft Exchange Server to infiltrate the system. The ransomware has a Russian origin, as it avoids any device that has a language setting in Russian and/or some other language from any of the former Soviet countries. The primary targets includes US-based organizations in critical infrastructure sectors such as government, finance, and food & agriculture [27]. DarkSide is another APT type ransomware which was involved in attacks on Colonial Pipelines and Toshiba [28]. REvil is also an APT type ransomware which was responsible for attacks on entities which are suppliers of Apple Inc. and are responsible for stealing confidential information [29]. Both REvil and DarkSide have similar code base. Just like BlackByte, both REvil and DarkSide allegedly have Russian origin. They avoid devices that have language settings similar to Russian and former Soviet countries [30]. BlackMatter is another APT type ransomware, which targeted multiple U.S. critical infrastructure entities, including two U.S. Food and Agriculture Sector organizations [31]. They are allegedly a "rebrand" of DarkSide ransomware and their main targets include food and agricultural sector [32]. Diavol, an APT type ransomware, is allegedly linked to a cybercrime group called Wizard Spider who are also known as Trickbot. They are also the perpatrators of the ransomware Ryuk, Conti and the spam Trojan Emotet [33]. Diavol, just like the other ransomware created by TrickBot, attacks corporate victims, especially financial institutions which use Windows [34], [35]. TrickBot is a nation state actor with alleged connections to Russian intelligence agencies [36].

3.3 Mitigation Techniques against APT

Traditionally APT attack mitigation is carried out by following some or all of the 12 processes [37]:

Anomaly Detection: Deviation from expected behavior by certain processes or network traffic can help in detecting and thwarting an APT attack.

Whitelists: Whitelisting trusted units in a network and communicating exclusively with them can lower the risk of an APT attack.

Blacklists: Blacklisting known malicious entities in a network and not communicating or blocking them also lowers the risk of an attack by APT groups.

Intrusion Detection System (IDS): Having a strong and state-of-the-art intrusion detection system can inform the user of any form of intrusion and an attack can be thwarted in early stages.

Awareness: Creating awareness among people regarding their behavior in the cyber-physical space can help in minimizing the human error or human-factor induced vulnerabilities in the system. This reduces the chances of an attack on the system due to exploitation of vulnerabilities induced by human beings.

Deception: Deception as a potential weapon against attacks by APTs has been used in various forms. The goal is to deceive the attacker in believing that it has become successful in mounting an attack, while analyzing the malware and preparing a defense action against the attack.

Cryptography: Having a strong encryption scheme increases the security of the system. It can often help in increasing the privacy as well.

Traffic/Data Analysis: Often statistical methods are employed to analyze traffic and data in the system to look for any breach in network protocols or any other form of anomaly in detecting attacks from APTs.

SIEM: In the Security Information and Event Management (SIEM) tool the data is collected and analyzed centrally. It is one of the most popular ways to detect and protect a system against APT attacks.

Pattern Recognition: Looking into the pattern of malicious activities performed by applications can be used to thwart future attacks.

Risk Assessment: Assessing the risks and possibilities of attack on the system can be helpful in preparedness against attacks.

Multi-layer Security: Communications in computer systems happen through multiple layers. Having a strong security system for each layer helps in bolstering the security of the system.

3.4 Trusted Computing

In our research, we aspired to provide a layer of deception with the use of commercially available hardware-based components. In this regard, we explored the following option. The Trusted Platform Module (TPM) is a hardware component designed following the guidelines of the security consortium, the Trusted Computing Group (TCG) [38]. The TPM comes with essential cryptographic potential. It can generate cryptographic keys, both symmetric and asymmetric. It has the capability of generating random numbers when required and can store cryptographic credentials. It also provides hashing capabilities. The primary functionalities of TPM include verification of platform integrity, safeguarding encryption keys, and preservation of password and user credentials. Figure 3.1 gives a simplified schematic of the TPM version 1.2, the specifications of which are laid down by the TCG [39]. TPMs today come in different incarnations that depend on the type of device and the manufacturer. Intel Software Guard Extension (SGX) and ARM TrustZone are versions of TPM-like hardware which come with certain functionalities in addition to the ones already mentioned for TPMs [40], [41], [42], [43]. They provide a *Trusted Execution*

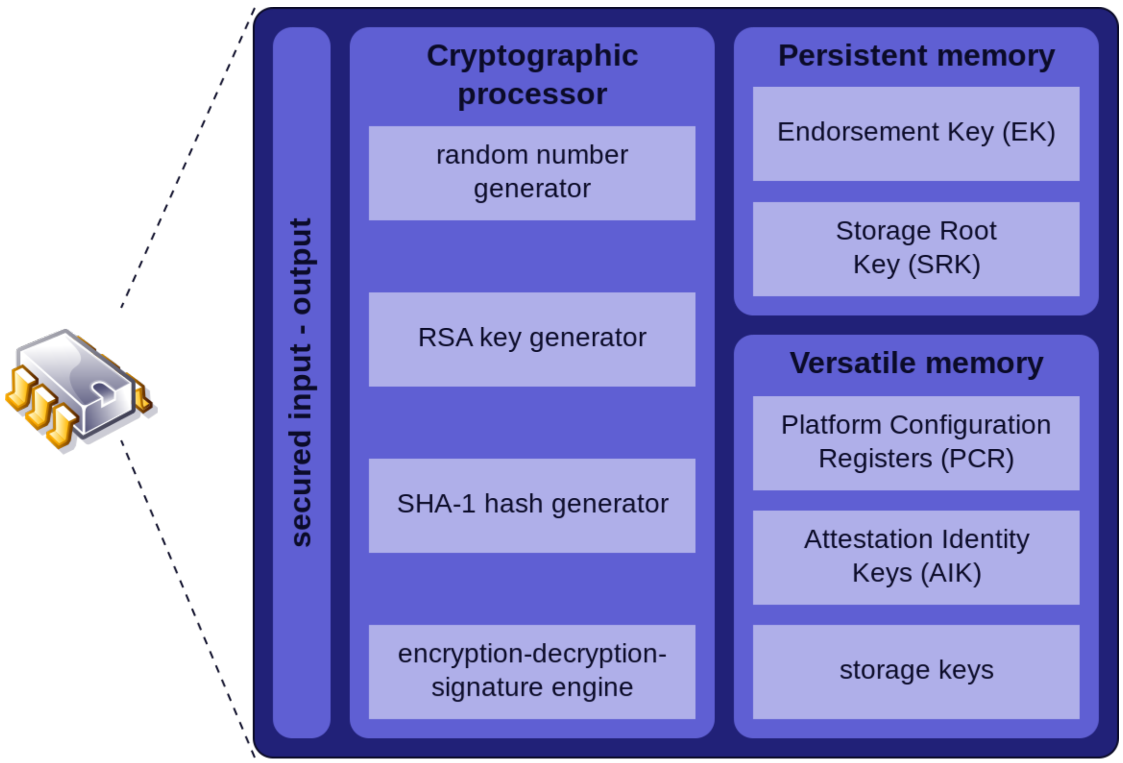


Figure 3.1: A Simplified Schema of TPM

Environment (TEE), which is generally outside the purview of high-priority OS instructions but can be accessed using the user credentials. Therefore, in general it can be assumed, even if the OS is compromised, that the hardware component is outside the purview of the attacker.

3.5 AI Models for Intrusion Detection

In this section, we examine AI models which we use in our research for designing an effective behavior oriented intrusion detection system (IDS).

3.5.1 Hidden Markov Models

Hidden Markov Models (HMM) have been historically used for speech recognition [44], [45]. HMM has also been applied for handwritten character and word recognition [46]. The biggest advantage that comes with HMM is that, in a process wherein the stages are not visible to the observer, certain observable features can be used to predict the stage of the process at a certain instance. Owing to this advantage, HMM-based techniques have often been used for the analysis of sophisticated malware. Metamorphic virus can be an annoyance. A metamorphic virus is capable of changing its code and become a new variant of itself without changing the functionalities. The changes are not exactly visible to the observer and therefore observable characteristics play an important role in the analysis. HMM has been used for detection and analysis of such metamorphic viruses [47].

3.5.2 Machine Learning Algorithms

3.5.2.1 Naive Bayes Classifier (NBC)

Naive Bayes is a supervised machine learning algorithm based on applying Bayes' theorem along with the conditional independence assumption between every pair of features [48]. It assumes that one feature of a class is unrelated to the other features, regardless of any correlation, and will predict the class of unknown datasets based on the Bayes' theorem of probability. This classifier is known to work well in real-world text classification situation such as Spam identification, and Recommendation Systems.

3.5.2.2 Support Vector Machine (SVM)

Support Vector Machine is a type of supervised learning method used for classification, regression, and outlier detection [49]. It is an effective algorithm in high dimensional spaces. It works by creating a hyper-plane or set of hyper-planes in the high dimensional space to separate the data into groups based on the classes and this hyper-plane is used for classification. The hyper-plane is chosen based on the separation achieved between the classes. A good hyper-plane is characterized by a larger margin, i.e., maximum distance between the support vectors (which are the nearest training data points). The hyper-plane or separation line is determined using these support vectors by identifying the hyper-plane that gives the highest margin, since higher the margin the lower the error of classification.

3.5.2.3 Logistic Regression (LR)

Logistic Regression is another supervised learning model to perform binary or multivariate classification [50]. The prediction of a class for a given data point is performed using a linear equation and by passing its output to a logistic function to get a value between 0 to 1. Typically, a sigmoid function is used to restrict the values between 0 and 1.

3.5.2.4 Gradient Boosting (GB) Decision Tree

Gradient Boosting Decision Tree is an ensemble learning model for performing supervised classification tasks [51]. An ensemble model performs predictions using a collection of models. Boosting is one of the ways of building an ensemble model where the collection of models are trained sequentially where each model learns from the previous models' mistakes. In Gradient Boosting, multiple weak learners are used together to build one strong learner (where each weak learner is a decision tree). The trees are combined one at a time and trained to correct the prediction mistakes made by the prior models. These weak learners are combined in a series and each new learner takes into account the errors from previous learner so that the overall model improves.

3.5.2.5 Random Forest (RF)

Random Forest is a supervised learning algorithm that is based on the ensemble model [52]. It derives that predictions by creating multiple decision trees. This algorithm works on the concept of Bagging where multiple models are trained in parallel and each of these models is trained on a random subset of the data. Random Forests can be visualized as parallel combination of decision trees and the results from these trees are combined to come up with the final classification.

3.5.3 ML based IDS

Researchers have long used KDD CUP'99 dataset to design and evaluate IDSes [53]. It is a popular dataset which has five million and two million records for training and testing, respectively. Using this dataset, researchers can design classifiers working on 41 features to detect attacks like denial of service (DoS), probe, remote to local (R2L), and user to root (U2R) [54]. NSL KDD is a more refined version of KDD CUP'99 dataset by removing several of its integral issues. NSL KDD is also a 41 feature dataset and is used for detection of attacks similar to KDD CUP'99 dataset [55], [56]. These datasets are mostly used for designing network based IDS (NIDS) [54]. Researchers have been plagued by the paucity of data regarding APT type ransomware. To help with this problem, for our research, we created a dataset of system call logs for the design and evaluation of host based IDS (HIDS) in the context of APT type ransomware attacks.

Researchers have often used a variety of machine learning (ML) techniques to detect malware. Alkasassbeh et al. [57] used data mining techniques along with classification techniques like multi-layer perceptron (MLP), Naive Bayes' classifier (NBC) and Random Forest to detect distributed denial of service (DDoS) attack. Almaseidin et al. [58] used classifiers like J48, Random Forest, Random Tree, Decision Table, MLP, NBC and Bayes Network on KDD dataset to design and evaluate IDSes to detect attacks like DoS, U2R, R2L and Probe. Halimaa et al. [59] used SVM, and NBC on NSL-KDD dataset to detect DoS, Probe and R2L attacks. Keserwani et al. [60] in order to obtain features from IoT network used a combination of Particle Swarm Optimization (PSO) and Grey Wolf Optimization (GWO). The features obtained are then provided to a random forest (RF) classifier in order to attain a high attack detection accuracy. They used KDDCup99, NSL–KDD, and CICIDS-2017 datasets to design and evaluate their NIDS. Krishna and Arunkumar [61] also proposed a hybrid GWO-PSO optimization algorithm used in conjuction with RF classifier on NSL-KDD dataset to detect DoS, R2L, U2R and Probe attacks. They have even compared their results with the results obtained from classifiers LSTM-RNN and Gradient Boosting Decision Trees. But the existing aforementioned datasets are generally used for designing NIDS. In our research we created a dataset for APT type ransomware and designed a HIDS to detect the same.

3.5.4 Natural Language Processing based Models for IDS

The recent advancements in Natural Language Processing (NLP) has greatly improved the abstract understanding of language and representation of language. The development of state-of-the-art Language Models has enabled setting high benchmarks in various real-world applications such as question answering, machine translation, text summarization, and text classification. Language Models capture the probability distribution of words in a word sequence, i.e., it can predict the probability of a given word sequence being correct. Primarily there can be two types of Language Models, Statistical Language Models and Neural Language Models. The neural models have been traditionally based on Recurrent Neural Networks (RNN) which are widely used for processing sequential data. RNN based architecture is sequential in nature and processing at each time step is dependent on the output/processing of the previous one. This prevents the use of parallelization to improve training time. Then came the Transformer based architecture [62] that could overcome the problem of parallelization using the mechanism of Attention. This allowed the model to get the context for any position in the input and process the input data in parallel without having to wait for previous time-steps to process a time-step. As a result, the parallelization enabled much larger data to be used to train these models and also reduced the training times.

Tran and Sato used NLP-based approach to analyze and classify malware from the data collected from API call sequences [63]. They collected behavioral data from malware from API call sequences. Thereafter, they performed feature extractions and feature vectorization using TF-IDF and Paragraph Vectors. Following that, classification was done using KNN, SVM, RF, and MLP.

Najafi et al. [64] used the events log of SIEM at enterprise systems to model the behavior as directed acyclic graphs (DAG). They used a NLP-based approach to detect graph based outliers. They evaluated their model in enterprise like environment and discussed the feasibility and effectiveness in advanced malware detection in the real-world.

Wang et al. [65] presented a research on the challenging problem of APT attribution. In their research, for the intermediate representations from wide variety of architectures, they used the VEX IR method. They chose two features, viz., string features and code features. They used strings as the behavioral features of the malware. The string features were the texts in the behavior report. They used paragraph vector distributed memory (PV-DM) to generate vectors for code features. They used bag-of-words for the vectorization of string features. They applied random forest classifier (RFC) and deep neural network (DNN) classifier to learn and classify. Finally, they ran RFC and Local Interpretable Model-agnostic Explanations (LIME) to interpret the results from the classification task.

Bidirectional Encoder Representations from Transformers (BERT) is a transformer based language model that revolutionized the field of Natural Language Processing [66]. BERT model differs from the traditional transformer architecture in a way that it uses only the encoder instead of both encoder and decoder design. BERT is designed to pre-train deep bidirectional (as it takes entire sequence at once) representation from text and can be fine-tuned and customized for specific and unique tasks using one additional output layer. Based on this architecture, huge pre-trained models have been trained and open-sourced. These pre-trained models have been trained on humongous dataset (nearly 3.3 billion words using data from Wikipedia, Google's BookCorpus) and this contributes to the deep understanding of the English language of BERT models. BERT's source code and several pre-trained BERT models are available open-source and free. This allows the community to utilize these state-of-the-art models running quickly and tailor, fine-tune the models and its performance for their custom use-case.

Researchers have previously worked on the intersection of NLP and Cybersecurity. Rahali and Akhloufi [67] used transformer-based models to automatically identify malicious software. They processed the source code of Andriod APKs (applications for Android system) and performed static analysis to identify malware and classify them into malware categories. The neural net model LSTM and transformer-based models such as XLNet, RoBERTa, DistilBERT, BERT were used to perform both binary and multi-classification and these models' performances were compared. Andronio et al. [68] presented an NLP based ransomware detection model called HelDroid for mobile device platform. It primarily identifies malicious apps. The authors implemented "Threatening Text Detector" on *OpenNLP*, an NLP library. It was designed for mobile devices running Android OS. They were able to detect ransomware with 99% accuracy on the samples which were not known to HelDroid. Continella et al. [69] put forward a detection model "ShieldFS", which focused on the low-level behavior of the ransomware. The model examines the activity at the low-level file-system and updates the rule-based system profile models. If any process does not comply with the models, then they are considered to be malicious. Scaife et al. [70] proposed "CryptoDrop", a behavior based early warning system for detecting ransomware. They use a set of behavior indicators pertaining to ransomware and in accordance to that the system is parameterized for ransomware detection. Kolodenker et al. [71] provided a proactive defense mechanism called “PayBreak.” PayBreak depends on the insight that ransomware often uses hybrid encryption system with symmetric keys for different sessions on the victim’s computer. It scrutinizes the use of those keys, put them in an “escrow” and thereafter use them to recover the files whenever the need arises to decrypt the resources encrypted by the ransomware. These researchers developed models to detect ransomware intrusion and also ransomware defense mechanisms against basic variant of the ransomware. In our research, we have devised a new way to identify APT type ransomware which are also applicable to the basic variant. The focus is on using system calls and other metadata captured about the processes to create a dataset that allow us to design an IDS to detect ransomware that disguise themselves as legitimate processes.

3.5.5 Evaluation Metrics

The model evaluation is an important step to understand the performance of a model and to evaluate how well a model can classify/predict the values for unknown datasets. The evaluation is done by getting the predictions on a smaller dataset (called the test set) that is held out from the model during the training phase and this test set is not seen by the model before the evaluation phase. The metrics chosen for evaluation also allow us to compare one models’ performance with that of another model. In our experiments, the below explained metrics are used. Let us consider the following notations used in the metrics [58].

1. True Positive (TP): A malicious event is classified correctly as malicious by the model.

2. False Positive (FP): A non-malicious event is classified incorrectly as mali-

cious by the model.

3. True Negative (TN): A non-malicious event is classified correctly as non-malicious by the model.

4. False Negative (FN): A malicious event is classified incorrectly as non-malicious by the model.

3.5.5.1 Accuracy

Accuracy is the proportion of the correct classifications to the total number of classifications performed in the test set, i.e., it denotes the fraction of correct classifications. The value of this metric is between 0 to 1 and a larger value indicates better classification accuracy.

$Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$

3.5.5.2 F1 Score

F1 score metric can be explained in terms of two other commonly used metrics, precision and recall.

- Precision: Precision value denotes the extent to which the positive (malicious events) values classified by the model are correct. It is given by count of true positives divided by the total number of positive classification. So, it is the proportion of positive classifications that are correct to the total positive classifications.
  $precision = \frac{TP}{TP+FP}$

- Recall: Recall is also called as sensitivity of the model and denotes the ability of the model to classify the available positive events (malicious events) correctly. This value indicates how many positive events can be classified correctly by the model over all the available actual positive events.

$recall = \frac{TP}{TP+FN}$

F1 Score is the harmonic mean of precision and recall values and it is a way of getting a weighted average of precision and recall. Since F1 score takes both the precision and recall, higher the score, the better is the performance.

$F1 = \mathbf{2} * \frac{\mathbf{precision.recall}}{\mathbf{precision+recall}}$

3.5.5.3 Receiver Operating Characteristic

The Receiver Operating Characteristic Curve (ROC Curve) is a plot which can be used to evaluate and compare classification models. This graph is plotted against the true positive rate and the false positive rate.

- True Positive Rate (TPR) is the proportion of the true positive classification to the number of actual positive classes available. This is same as the recall of a model.

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

- False Positive Rate (FPR) is the proportion of false positives to the number of actual negative classes that are available. This metric is also called as fall-out of the model.

  $FPR = \frac{FP}{FP+TN}$

Plotting ROC curve and computing the area under the curve (AUC) is a commonly used metric to evaluate the performance of a model. The AUC metric calculates the entire two dimensional area under the plotted ROC curve and it represents the extend to which the model can separate or distinguish between the classes to be classified. This is an indicator of separability of the model.

3.5.5.4 Training Time

Every model is trained using the training dataset. The model parameters and weights are updated as part of this training step and this requires certain compute power. Capturing the total time taken by the model to train indicates the computation time required to get the model ready before we can get classification/predictions using the model.

3.6 Game Theory for Security and Mitigation

Game theory opens up new avenues for malware analysis. Cartwright et al. [72] proposed a game-theoretic model to analyze generic ransomware attacks. They used kidnapping game [73], [74] as the basis for the model. The malware was modeled as the kidnapper whereas the database of the victim was modeled as the hostage. The goal of the paper is to help the defender to make an informed decision regarding the payment of the ransom. Spyridopoulos et al. [75] investigated a game-theoretic approach for the cost-benefit analysis of malware proliferation, and modeled it on the lines of epidemic spread models, SIR and SIS. They applied their models on the well-studied Code-Red worm. The idea was to develop a cost-benefit game-theoretic model to apply malware proliferation strategies including “patching” of infected nodes in a network, “removal” of infected nodes in the network, and/or the combination of both. They used “FLIPIT” game as the basis for the development of their model. But this research was primarily done to analyze and mitigate a generic ransomware attack. In our research, we investigate the sophisticated APT type ransomware attack through a “sequential game”, so that future developments can be incorporated in the ever evolving attack landscape of APTs.

Khouzani et al. [76] used a zero-sum dynamic game-theoretic model as a solution to malware attack. They analyzed the structural properties of saddle-point strategies, which are simple threshold-based policies, and showed the possibility of a robust dynamic defense system against malware attacks. They have investigated the network defense landscape of mobile wireless networks. The strategies on the part of defender those were investigated were reception and patching rates. The strategy of the attacker that was investigated was annihilation rate of the infected nodes. Through the formulation of a dynamic game it was proven that threshold-based policies form an effective robust solution to malware attacks.

In the world of Internet of Things (IoTs), a cloud-assisted malware detection and suppression framework has been put forward by Zhou et al. [77]. The authors have used a support vector machine (SVM) based malware detection technique with sharing of data at the security platform in the cloud. This was followed by an epidemic spread model to show the state transition of the wireless media system (WMS) to emulate the spread of malware in the network. Finally, they used dynamic differential game to calculate the Nash equilibrium as a solution to the attack, based on which the malware suppression framework was designed.

Cekar et al. [78] have used deception to counter denial of service (DoS) attacks. They have also used a game-theoretic model based on signaling game with perfect Bayesian equilibrium (PBE) to investigate the implications of deception to counter the attacks. Deception as a potential defense tool has been used to lure the attackers to high interaction honeypots in [79] and thereby designing an effective malware detection system. The author proposed an adaptive honeypot system based on game-theoretic concepts to lure the attackers leading to the detection of the rootkit malware by the defender. Deception as a defensive tool has been an important area of research, and so are the game-theoretic models for its analysis in the cyber-physical domain [80], [81], [82].

A multi-phase and multi-stage game-theoretic model has been developed by Zhu et al. [83] to systemize the threat mitigation strategies against APTs. A three-phase game was considered in the aforementioned paper which included spearphishing game in phase 1, followed by phase 2 game modeled on learning and penetration behavior of the malware, and finally phase 3 game modeled on the final stage of the APT wherein the damage has been eventually caused to the victim. The spearphishing game was modeled as a simplified Bayesian game, the multi-stage penetration of the networks was modeled as a nested game and final-phase game was modeled as a finite zero-sum game.

The research in the Khouzani et al. [76], Zhou et al. [77], and Zhu et al. [83] papers were aimed at the game theoretic analysis of attacks mounted by malware. In our research, we also use game theoretic model for the analysis of APT type ransomware. We follow our sequential game model by a sensitivity analysis as a means to study the effect of the changes in the values of different parameters on the equilibrium strategies of the players.

3.7 Summary

In summary:

- We looked into a class of malware of the type Advanced Persistent Threats (APT).

- We explored different variants of ransomware including the APT variant.
- We browsed through some of the traditional APT mitigation techniques.
- Before we introduce the concept of deception in our research, we examined the commercially available hardware components used for the same.
- We scrutinized some of the AI models and the evaluation metrics that we used for the purpose of malware detection in our research.
- Finally, we investigated the usage of Game Theory with regards to the development of mitigation strategies for our research.

Chapter 4

Advanced Persistent Threats Redefined

"Every search for a hero must begin with something which every hero requires - a villain."

– Robert Towne,

Mission Impossible II

In this chapter, we identify several new attack attributes with the goal of characterizing the current state-of-the-art advanced persistent threats (APT). APT is the main category of threats this dissertation is primarily focusing on. The attributes identified would help the defender devise a defense mechanism and mitigation strategies when faced with an attack. The following sections discuss the details of the research done in this regard.

4.1 New Definition

Identification of an APT while the attack is unfolding is non-trivial due to the dynamically changing landscape of sophisticated attacks. In order to address this, we first defined a set of characteristics of an attack. If the attack displayed certain well-defined characteristics, we could say that the threat is an APT within the bounds of our definition and an action could be taken to mitigate the after-effects of the attack. The proposed parameterized model is a customization of the stealth attack model of [17] to fit the new breed of attacks, viz. APT.

Most published cyber-attacks in the literature are single-shot, staged by running certain attack scripts and are instantaneous. The typical defense mechanism is reactive in nature. On the other hand, the APTs are generally mutli-shot and spread over multiple stages [17]. The defense against an APT is different from that of the non-APT attacks. It is possible to proactively handle an APT if it is recognized at an early stage. So an early detection and classification of the threat is important from the defender's point of view. For the purpose of classification, we considered five main characteristics shown in Table 4.1. For each characteristic we defined certain parameters, which would help in identifying the threat.

4.1.1 Reconnaissance

Reconnaissance is a characteristic possessed by almost all kinds of attacks. Most of the attacks perform some external reconnaissance on the system and many do internal reconnaissance once inside the system. Therefore, we defined two parameters for this characteristic, namely, P1.1 and P1.2 to recognize both of

Table 4.1: Characteristics and Parameters of an Attack

Characteristics	**Symbol**	**Parameter**	**Description**
Reconnaissance	P1	P1.1	External Reconnaissance
		P1.2	Internal Reconnaissance
Exploit	P2	P2.1	Hardware Vulnerabilities
		P2.2	Software Vulnerabilities
		P2.3.1	HW Timing Constraints
		P2.3.2	SW Timing Constraints
		P2.4	Target Acquisition
		P2.5	Affected Resources
Authorized Access	P3	P3.1	Admin Rights
		P3.2	Backdoor Implants
		$P3_{backdoor}$	Backdoor Implant Tools
Honeypot Interaction	P4	P4.1	Low Interaction Honeypots
		P4.2	High Interaction Honeypots
		P4.3	Activity Logging
Contingency Plan	P5	P5.1	Abort Mission
		P5.2	Different resources, same attack
		P5.3	Different attack, same resources
		P5.4	Different attack and resources
		P5.5	Decoy attack

these activities.

4.1.2 Exploit

The Exploits characteristic encompasses all the exploitable vulnerabilities and the parameters defined here are used to calculate the risk associated with the vulnerabilities. In this characteristic, five parameters are defined. Each parameter enlists the vulnerabilities associated, which can be exploited by the attacker to mount an attack. Parameter P2.1 enlists all the exploitable hardware vulnerabilities. Parameter P2.2 encompasses all the exploitable software vulnerabilities. Parameter P2.3 gives the time constraints regarding exploitation of the vulnerabilities. It denotes the time required for exploiting the hardware as well as software vulnerabilities. This parameter is particularly very important from the attacker's point of view. It gives them an idea to run a feasibility test. It is also important from the defender's perspective since this parameter helps a defender to understand whether a system is unconditionally secured, computationally secured or not secured. Parameter P2.4 enlists all the target components, which had either been acquired or will be acquired. This parameter gives an idea to the defender about the components, which are to be secured on a priority basis at the event of an attack. The last parameter of this characteristic is P2.5, which lists the affected resources.

4.1.3 Authorized Access

Reconnaissance and Exploit characteristics are generic and common to most threat families. On the other hand, Authorized Access is a major characteristic of a quiet invader [1] and/or an APT. This characteristic signifies the capability of the threat to gain authorized access to the system and maintains the foothold. The parameters defined in these characteristics are P3.1 and P3.2. P3.1 is different from the other parameters in the sense that it is a Boolean parameter. P3.1 is True if the attacker is able to gain administrative privileges, else it is False. P3.2 enlists all the backdoor implant tools used to install backdoors. This parameter is very important from defender's perspective as it gives an idea about the nature of attack being carried out. This parameter also opens up the avenue for the defender to snoop into the communication between the malware installed by the attacker and the Command and Control (C&C) centers. If this communication between the malware and the C&C centers can be breached, then the attack can not only be thwarted but also the family as well as the nature of the attack can be known easily.

4.1.4 Honeypots

For the purpose of defense many systems use deception as a tool. Use of honeypots and honeypot farms is such an application [17]. Parameters P4.1 and P4.2 enlist the threat and honeypot pairs and log their interaction. Parameter P4.3 is another Boolean parameter, which is True if the honey-pots are logging the activities of the malware and is False otherwise.

4.1.5 Contingency Plan

A distinguishing feature of our APT model is the recognition of the presence of a contingency plan in an attack campaign [84]. A defender might come up with a defense mechanism while under attack, but the attacker realizing that a defense is building up might change the mode of attack. The renewed attack might be totally different in nature throwing the defender off guard and making the system more vulnerable than it was before. Therefore, to understand this characteristic, we defined 5 sub-parameters. The parameter P5.1 is a Boolean parameter. P5.1 is True when the contingency plan of threatening attacker is to abort mission when discovered, and False otherwise. This may be viewed as a default plan on the part of the attacker. Parameter P5.2 enlists all the resources, which can be attacked using the same mode of attack. Parameter P5.3 enlists all the different modes of attacks, which can attack the same resources, which are currently under attack. Parameter P5.4 enlists all the attacks and resources, which can be attacked except for the current mode of attack and the resources, which are currently under attack. Parameter P5.5 addresses a certain plausible scenario. Though this parameter is enlisted under Contingency Plan it can become the main plan as well depending upon attacker skills and the dynamics of the application environment. It enlists all the probable decoy targets, which are currently under attack in order to take the focus away from the main target. It primarily enlists all the sets in which one or more targets are the main targets and other targets, which can be probable decoys for the main target. It also states which sets are currently active, which means that the decoys are currently under attack and the main target is at risk.

4.2 Summary

In summary:

- In this chapter we put forward a research which identifies new attributes to characterize state-of-the-art APT attacks.
- These attributes play a key role in differentiating APT type attacks from non-APT type attacks.
- Owing to this update, we were able to differentiate between APT type ransomware and non-APT type ransomware.
- These attributes help us build better defense architecture and make informed decisions when faced with an attack.
- More details can be found in the published work [2].

The new attributes identified help to account for the more recent and sophisticated attacks. The initial step in designing a defense system is threat identification. These attributes would not only help in proper identification of the threats but also help in designing strategies to mitigate the attacks.

Chapter 5

Attack Detection

"He will win who knows when to fight and when not to fight."

– Sun Tzu,

Art of War

The initial definition of APT in the literature had become outdated and would not account for the most recent state-of-the-art sophisticated attacks. In Chapter 4, we updated the literature by redefining an APT. With the newer features being attributed to the APT attacks, a new system is needed to detect an intrusion by an APT type malware. We designed a Hidden Markov Model (HMM) based intrusion detection system (IDS) for APT type ransomware [85]. HMM has traditionally been used for handwritten character recognition, speech recognition, analysis of polymorphic and metamorphic virus. The advantage of using HMM in our model was to identify the state of the attack (which are generally hidden owing to the stealthy nature of the malware) from the observable features. This would help the defender tailor a defense response best suited to that particular state of attack. This way the defense response would be more effective and would affect the performance to a lesser extent. Another advantage of using HMM for the IDS is that it not only characterizes the attack but also the attacker. This helps the defender to be better prepared for similar future attacks. Additionally, we showed that malware spread models can be used to obtain the indicators of compromise (IoC) and can be effectively used to aid the HMM based IDS in detecting an intrusion by an APT type ransomware [86]. Furthermore, we present multiple models to make an IDS smarter with the use of machine learning (ML) and natural language processing (NLP) models. We create a dataset by collecting system call logs from running a few APT type ransomware, viz. Black- Byte, BlackMatter, Diavol, REvil and Darkside using a commercial tool. Then we run our ML and NLP based detection models and compare their accuracy and performance.

5.1 Malware Spread Model

The threat being considered in this dissertation is a ransomware type malware created by the APT groups. A careful analysis of this threat will reveal the types of vulnerabilities, the systems that are at risk and the spread of the malware. This knowledge would help the defender to identify the indicators of compromise for the analysis of the HMM based APT detection system. The spread model would help the defender in detecting the malware in its early stages. For the spread of the ransomware we consider the following spread and growth models, viz., exponential growth, epidemic spread model and random walk.

5.1.1 Model 1 - Exponential Growth

The exponential growth model [87], [88] can be applied to those malware which infect the systems and/or devices that communicate directly with the infected systems and/or devices. A number of new infections are caused by already infected cases. If we assume that the population is large enough then we can use the exponential model. Let us assume that N_t nodes in a networked system of a very large enterprise are infected at a given time t. Let E be the average number of nodes which are directly interacting with an infected node, and let p be the probability of an exposed node becoming infected from its interaction with an infected node. Then the increase in the number of infected nodes is given by

$$\Delta N_t = E \cdot p \cdot N_t$$

At time t+1 we have

$$\Delta N_{t+1} = N_t + E \cdot p \cdot N_t$$

$$\Delta N_{t+1} = (1 + E \cdot p) \cdot N_t$$

With the aforementioned logic we can have

$$\Delta N_t = (1 + E \cdot p)^t \cdot N_0$$

where N_0 is the initial number of infections or the number of infected nodes at $t = 0$.

Therefore, $(1 + E \cdot p)$ is a constant greater than 1, which means the infection is forever exponentially increasing. But that is not always the case. With time, either a kill-switch is discovered or the non-infected nodes stop interacting with the infected nodes.There can be another possibility, wherein the infected nodes are interacting only with the infected nodes, and thus, no new infections are found. Taking these factors into account, one can introduce new parameters in the equation to form a logistic curve from the rate of change of infected nodes given by the following equation

$$\frac{dN}{dt} = c(1 - \frac{N}{PopulationSize})N$$

wherein N is the number of infected nodes at a given point of time, c is the constant of proportionality and $PopulationSize$ denotes the total number of nodes in a networked system. Initially the the logistic curve is generally indistinguishable from an exponential growth when the slope is increasing till the time it reaches the “inflection point”, when the slope is 1. After crossing the inflection point, the slope is decreasing and the rate of change of infected nodes decreases till it saturates out or becomes 0. The growth factor is generally taken into account which is given by

$$GrowthFactor = \frac{\Delta N_{t+1}}{\Delta N_t}$$

which is number of new infections at time $t + 1$ divided by number of new infections at time t. Before the inflection point, the growth factor is generally greater than 1 and it is less than 1 after the inflection point. At inflection point the growth factor is 1.

5.1.2 Model 2 - Epidemic Spread Model

The spread of diseases which become epidemic are perceived mathematically using the epidemic spread model [89]. It helps in assessing the risk to the uninfected and measure the spread of the disease. Inspired from real life, the epidemic spread model can be applied to the cyber-physical world [90], [91]. It is similar to the exponential growth model but a bit more detailed. It takes into account both the susceptible systems and/or devices as well as the recovered ones. It gives a more comprehensive idea of the nature of the malware spread which could help the defender to apprehend the spread. There are two popularly used epidemic spread models, which are Susceptible-Infected-Recovered (SIR) and Susceptible-Infected-Susceptible (SIS). In the SIR model, the susceptible object can get infected and after infection a chance of recovery by some means is possible. It is assumed in this model, that once the object has recovered, it cannot be infected by the same infectious malware. But in the SIS model, an infected object who has recovered can become susceptible to the same malware after a duration called incubation period. Generally, if a system has been attacked by a certain type of malware and has recovered, then there is a very low chance that it will be affected by the same malware again. After the attack, the vulnerabilities are generally taken care of through patch release and/or taking back-up of the critical data of the system. With these assumption, we moved forward with the SIR model.

For the model we use $N(t)$, $S(t)$, $I(t)$, and $R(t)$ for the total number of nodes, number of susceptible nodes, number of infected nodes, and number of recovered nodes, respectively in a networked system. This give us $N(t) = S(t) + I(t) + R(t)$. We use the standard notations for infection rate and recovery rates which are δ and γ, respectively. For the type of malware under consideration, susceptible nodes are the ones which have the relevant vulnerabilities, that are being exploited for infiltration, and are communicating directly with the infected nodes. Recovery rate indicates the removal of the infected nodes and/or removal of the malware from the infected nodes. Once a malware starts spreading, the susceptible nodes can get infected if they communicate directly with the infected nodes or are in the same network as the infected nodes. From these assumptions we can calculate the following rates of change of susceptible, infected and recovered nodes, respectively:

$$\begin{aligned}\frac{dS(t)}{dt} &= -\delta \cdot S(t) \cdot I(t)\\ \frac{dI(t)}{dt} &= \delta \cdot S(t) \cdot I(t) - \gamma \cdot I(t)\\ \frac{dR(t)}{dt} &= \gamma \cdot I(t)\end{aligned}$$

The rates of change of number of susceptible nodes, infected nodes and recovered nodes help the defender know the behavior of spread and the nature of vulnerabilities being exploited. This would help in an early detection of the malware and risk assessment of the system for the attack. The Delta value (δ) also gives the time constraint for exploitation of the vulnerabilities that exist in the system [2]. The Gamma value (γ) gives a fair idea about the weaknesses of the malware and subsequently might help in discovering the kill-switch to thwart the attack [2]. Both δ and γ values, if correctly estimated, then become important elements in designing a behavioral based intrusion detection and/or intrusion prevention systems.

5.1.3 Model 3 - Random Walk

In a networked environment, if the malware can spread to the nearest neighbors, with equal probabilities for each of the neighbors then one can model the spread with the Random Walk model [92], [93]. The APT type ransomware, which is being considered as the malware for this dissertation, can move around randomly in a networked environment, if all the nodes have equal probabilities of possessing the vulnerabilities it exploits to infiltrate.

For smaller and simpler networked systems, one can use the 1-D random walk model to show the spread of the malware within the network. It can be as simple as the walk on the integer line. Just like the integer line, if we move our frame of reference with the first infected node at point zero, then the next move which the malware makes towards the nearest neighbor can be a random variable Z_i of value -1 or +1 with probability 50%. We set the value $S_0 = 0$ and then we have $S_n = \sum_{i=1}^{n} Z_i$. The series $\{S_n\}$ is known as the simple random walk on $\mathbb{Z}$, where $\{Z_0, Z_1, ..., Z_n\} \in \mathbb{Z}$. Such a series gives an idea of the distance traversed by the malware, if each of the hops made by the malware is of equal distance and is made with equal probability.

5.1.4 Usability of the Models

An early detection of malware created by APT groups gives the defender a leverage in effectively thwarting the attack. The spread of a particular malware often reveals the type of vulnerabilities that are being exploited for infiltration. It also helps in giving the defender an idea about the systems at risk and the timing constraints regarding the exploitation of the vulnerabilities [2]. The exponential growth model gives an idea about the growth rate, the inflection point and the extent of the infection. The biggest drawback of this model is that it is only applicable to a very large population size. It means that systems with very large number of nodes in a networked system might manifest such characteristics. The DDoS attacks on Dyn servers by the family of Mirai botnets infected around half-a-million IoT devices in order to carry out the attack in 2016 [94]. The networked systems with fewer nodes wouldn't be efficiently portrayed by this model. The SIR epidemic spread model suggests the estimate regarding the nodes at risk which are denoted by the susceptible nodes. It also gives an account of the infected and the recovered nodes in the system. It outlines the extent of nodes at risk, the vulnerabilities being exploited and the rate of infection. If the spread model is assessed in detail, then it might even give the rate of recovery and weaknesses on the part of the malware. This helps in building a behavior oriented countermeasure to thwart the attack mounted by malware created by the APT groups. If the malware spread can be modeled by random walk, then the primary advantage is that it gives the probability of a particular node being infected, once the attack has begun. It also gives an estimate of the time, within which a particular node can be infected with certain probability, from the distance of the node from the latest infected node. This gives the defender a valuable information regarding the time for preparedness.

There can be other models which can be used to analyze the spread of the malware, e.g., neighbor-based worm propagation model, Bernoulli-distributed neighbor-based worm propagation model, modeling worm propagation using Weibull distribution and random worm propagation model [95]. But for these models, a detailed information regarding the nature of the malware is required which generally is not available for the malware created by APT groups. Therefore, for this dissertation we stick to the observable features exhibited by APTs.

5.2 HMM based Ransomware Detection Model

5.2.1 The Model

The threat model under consideration in this dissertation primarily deals with ransomware which qualify as advanced persistent threats. This means that the attack mounted would be highly sophisticated and persistent in nature. Such attacks can render the traditional signature based intrusion detection systems ineffective. To deal with APTs that have no prior history, behavior-oriented defense systems are a necessity. APTs are generally mounted in multiple stages unlike more common threats. The knowledge of the stage in which an APT is currently in, is a utilitarian information for the defender to make an informed decision about the defense strategy. A crucial feature manifested by APTs is the existence of a contingency plan of attack [2]. A simple ransomware can be taken care of with the existing infrastructure and defense strategies but an APT with a contingency plan needs special attention. A contingency plan of attack is an alternate attack strategy, which attackers might resort to, if they believe that the defender is able to thwart the primary attack campaign. The type of alternate campaign the attacker might resort to can be completely different from the primary attack strategy. If the attacker is spooked, they can execute the contingency plan and that can inflict unwanted but significant damage to the victim.

The APT type ransomware are typically mounted by quiet invaders [1] and they subtly graduate through different stages. Therefore, difficulty arises in figuring out the status of the malware. One can look into the behavioral changes and using those as *observables* can help make an informed decision. To help the defender in making that informed decision, we develop a Hidden Markov Model (HMM) based intrusion detection tool. This tool will help the defender discern the status of the malware with certain probability, which would define its confidence in choosing the defensive action.

The proposed HMM has N number of hidden states and M number of observables. The model can be denoted by $\lambda = (A, B, \pi)$, where

- A is an $N \times N$ matrix that gives the transition probabilities, characterizing the transition of each hidden state to another. Hence, it is called the transition matrix.
- B is an $N \times M$ matrix that gives the emission probabilities for each hidden state. Hence, it is called the emission probability matrix.
- π is a $1 \times N$ matrix that contains the initial probability distribution for each of the hidden states.

This detection model strictly deals with ransomware. It intends to figure out whether a malware is a ransomware or not, and if it is a ransomware then is it a ransomware that has graduated to become an APT. Moreover, the model also investigates that if the ransomware is an APT then is it still pursuing its attack as a ransomware or would resort to a contingency plan of attack. Taking all these into consideration, we formulate the model using the following parameters:

- The value of N is 4 which denotes that there are 4 hidden states being considered in this model, termed as $Z = \{z_1, z_2, z_3, z_4\}$
- The value of M is 5 which denotes that the number of observable random variables is 5, termed as $X = \{x_1, x_2, x_3, x_4, x_5\}$

- α_{ij} denotes the transition probability of the malware from i^{th} latent state to j^{th} latent state, where $i \in \{1, 4\}$ and $j \in \{1, 4\}$
- β_{ir} denotes the emission probability of i^{th} latent state manifesting r^{th} observable behavior, where $i \in \{1, 4\}$ and $r \in \{1, 5\}$

The hidden or latent states of the malware are as follows:

- The first state z_1 is where it is just a malware, regardless of the fact whichever form of malware it graduates to.
- The second state z_2 is where the malware becomes a ransomware.
- The third state z_3 is the one wherein the ransomware has graduated to become an APT.
- The fourth and the final hidden state in this model is denoted by z_4, wherein the attacker chooses to execute the contingency plan of attack instead of mounting a ransomware attack on the victim.

The hidden states of the malware are often outside the purview of the defender's intrusion detection system and hence, the term hidden state, which entails the use of HMM based intrusion detection model for ransomware. For the model, as discussed earlier, the observable behavioral features, $X = \{x_1, x_2, x_3, x_4, x_5\}$, are used to ascertain the status of the malware. Following are the details regarding individual observable features used to design the model:

- x_1 : Reconnaissance
- x_2 : Interaction with honeypots or real-databases which are of high value
- x_3 : Backdoor implants and/or back-channel traffic

- x_4 : If the strategy of "Campaign Abort" exists
- x_5 : Existence of any other contingency plan of attack

The observable features help the defender to discern the latent state of the model, which the malware is in, while an attack is ongoing. It starts with the feature of reconnaissance. The prior knowledge about the spread of a malware created by same or similar APT groups (using one of the models described in Section 5.1) help the HMM model to differentiate the interaction of a legitimate process with the system from the interaction of a malicious process with the system. Moreover, the spread models of the malware also give an insight regarding the frequently exploited vulnerabilities of a system. Systems with very high value resources often deploy honeypots and/or honeypot farms. The interaction with the honeypots and the activity logs often reveal the nature of the malware and help in understanding whether the malware is a ransomware or not. The manifestation of other observable features like existence of back-doors and back channel communications, and existence of other plans of attack including a "Campaign Abort" strategy are critical features often portrayed by malware created by APT groups. Hence, these features are important in ascertaining the latent state of the malware in the HMM based detection model, while it is performing an attack.

Figure 5.1 shows the HMM based ransomware detection model. With the latent states, observable features, and the associated parameters, we can determine the transition probability matrix A and the emission probability matrix B. We have the following for matrices A and B, respectively:

$$A = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix}$$

$$B = \begin{bmatrix} \beta_{11} & \beta_{12} & \beta_{13} & \beta_{14} & \beta_{15} \\ \beta_{21} & \beta_{22} & \beta_{23} & \beta_{24} & \beta_{25} \\ \beta_{31} & \beta_{32} & \beta_{33} & \beta_{34} & \beta_{35} \\ \beta_{41} & \beta_{42} & \beta_{43} & \beta_{44} & \beta_{45} \end{bmatrix}$$

Transition probability:

$$T(i|j) = \alpha_{ij} = p(z_{k+1} = j | z_k = i)$$

where $i \in \{1,2,3,4\}$ and $j \in \{1,2,3,4\}$

Emission probability:

$$\varepsilon_i(x_k) = p(x = x_k | z_k = i) = \beta_{ik}$$

where $\varepsilon_i(x_k)$ is the probability distribution on X, such that $x_k \in X, k \in \{1,2,3,4,5\}$ and $i \in \{1,2,3,4\}$

Initial probability distribution:

$$\pi(i) = p(z = i)$$

where $i \in \{1,2,3,4\}$

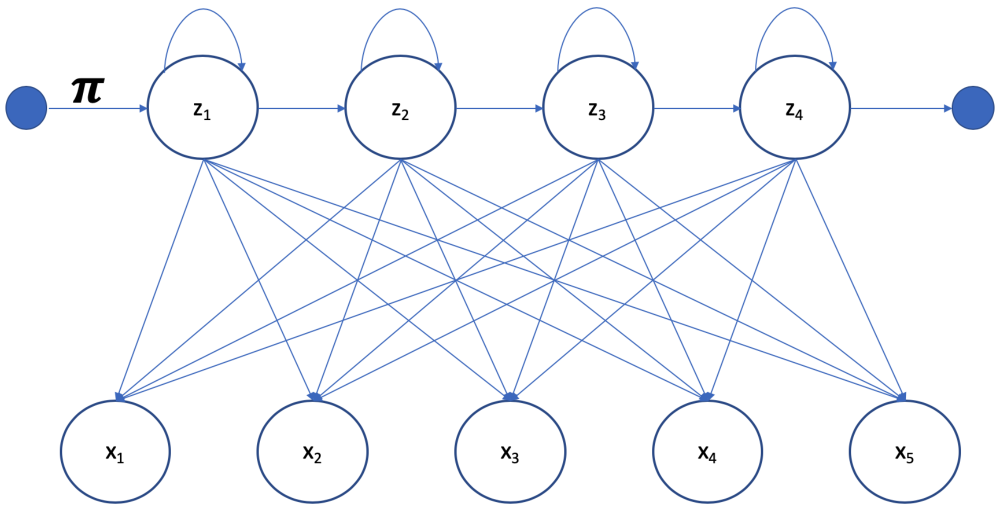


Figure 5.1: HMM based Ransomware Detection Model

The joint probability distribution is given by:

$$p(z_1, ..., z_4, x_1, ..., x_5) = \pi(1) \prod_{k=1}^{3} T(z_{k+1}|z_k) \prod_{n=1}^{5} \varepsilon_z(x_n)$$

The transition probabilities considered for this research are updated as in the following matrix:

$$A = \begin{bmatrix} \alpha_{11} & \alpha_{12} & 0 & 0 \\ 0 & \alpha_{22} & \alpha_{23} & 0 \\ 0 & 0 & \alpha_{33} & \alpha_{34} \\ 0 & 0 & 0 & \alpha_{44} \end{bmatrix}$$

The transition probability from state z_1 to state z_3 is 0 owing to the fact that it has to first go through state z_2 as it will portray the features of ransomware anyway. If it portrays features of any other form of malware, then it stays in this state as the detection of other forms of malware is outside the scope of this model. Similarly, the transition probability of state z_1 to state z_4 is also zero, as the malware cannot directly make a transition to the final state without becoming a ransomware first. According to the assumption made in this model, effectively the malware can remain in some other form of malware or become a ransomware.

The transition probability of state z_2 to state z_1 is assumed to be zero. The basis for the assumption is, if the model can depict characteristics of some other form of malware, which is not a ransomware, then it is effectively state z_4. Hence, any behavior of this type is categorized under state z_4. The same reasoning applies to the transition probabilities of states z_3 to z_1 and states z_4 to z_1. The transition probability of states z_2 to z_4 is 0, owing to the fact that in state z_2 it is already a ransomware, and if the attacker is planning to execute a contingency plan of attack then it is effectively state z_3 as it has already graduated to become an APT [2].

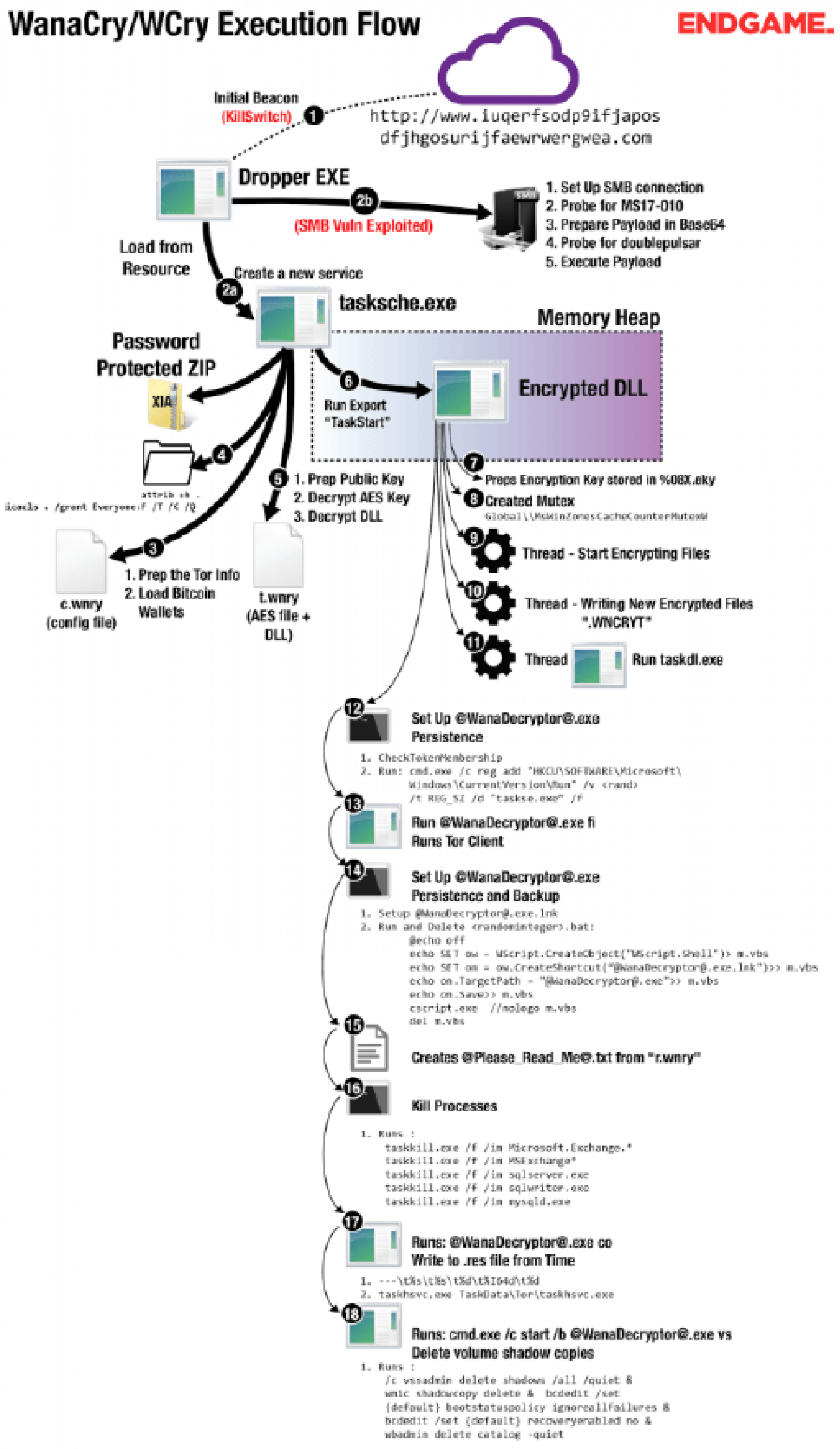
The transition probabilities of states z_3 to z_2 and z_4 to z_2 are assumed to be 0. In state z_3 the ransomware has graduated to become an APT. On reaching this state, the ransomware will execute APT type attack and/or will abort the campaign upon discovery. In state z_4 the APT type ransomware has decided to execute some other form of attack as a contingency plan of action owing to a belief of being discovered by the defender. The assumption here is that once a ransomware has graduated to become an APT, it cannot be considered as a simple ransomware, even though it executes a ransomware style attack and/or resorts to a contingency plan. Even if the attacker executes a contingency plan, which effectively is a ransomware attack, then there is a high possibility that the newer form of ransomware attack would be somewhat different from the primary form of attack, and therefore we assume this as an alternate form of attack and the model denotes the state to be z_4.

The initial probability distribution depends on the type of attacking group and the malware created by them. The probabilities with which the observable features are visible constitute the emission probability matrix and it depends on the type of resources, the system, the attack framework and the duration of attack.

5.2.2 WannaCry: A Use Case

We considered WannaCry [9], [96] to illustrate the effectiveness of the models developed in this research and the associated detection framework. It was created by the APT group named Lazarus from North Korea [96]. We first looked into the spread model for probable indicators of compromise (IoC). This is followed by HMM based ransomware detection taking over the control of predicting the stage of attack, so that the defender can tailor a defense strategy best suited to the stage of attack.

The series of attacks carried out by WannaCry in 2017 is known as "WannaCry Campaign." Figure 5.2 shows the execution flow of WannaCry [96]. The attack started on May 12, 2017 and ended on May 17, 2017. Over this period, the attackers earned somewhere between \$75,000 to \$80,000 from ransom [9]. The execution flow of WannaCry malware gives an insight to the indicators of compromise (IoC) to look for, which in turn would help the defender to look for the observable features for the HMM based intrusion detection system. The systems which contained the *Eternal Blue* vulnerabilities of SMBv1 are the ones which are *susceptible* to the WannaCry attacks. The dropper exploits the vulnerability to infiltrate the system as shown in Fig. 5.2. The ones which have already been locked out or the ones which have the *DoublePulsar* back-door implant tools in the system and some form of back-door communication is going on are the ones which can be termed as *infected* systems. Figure 5.2 shows that after infiltration, the main task of the malware is to encrypt the system using the AES encryption scheme [97], [98]. The WannaCry execution flow diagram also shows, how it queries a bogus domain in order to be certain that it is not being run in a controlled environment. If the malware believes that it is being run inside a sand-box or any controlled environment, it resorts to a contingency plan of attack. The systems which have received the decryption key after the payment of the ransom or the systems which earned some time once the *kill-switch* (or faking the initial beacon as shown in Fig. 5.2) has been triggered, are the ones that formed the population of *recovered* systems.


WanaCry/WCry Execution Flow
ENDGAME.
Initial Beacon
(KillSwitch)
1
http://www.iuqerfsodp9ifjapos
dfjhgosurijfaewrwergwea.com
Dropper EXE
2b
(SMB Vuln Exploited)
1. Set Up SMB connection
2. Probe for MS17-010
3. Prepare Payload in Base64
4. Probe for doublepulsar
5. Execute Payload
Load from Resource
Create a new service
2a
tasksche.exe
Memory Heap
Password Protected ZIP
6
Run Export "TaskStart"
Encrypted DLL
4
5
1. Prep Public Key
2. Decrypt AES Key
3. Decrypt DLL
3
1. Prep the Tor Info
2. Load Bitcoin Wallets
c.wnry (config file)
t.wnry (AES file + DLL)
7
Preps Encryption Key stored in %08X.eky
8
Created Mutex
9
Thread - Start Encrypting Files
10
Thread - Writing New Encrypted Files ".WNCRYT"
11
Thread
Run taskdl.exe
12
Set Up @WanaDecryptor@.exe Persistence
13
Run @WanaDecryptor@.exe fi
Runs Tor Client
14
Set Up @WanaDecryptor@.exe Persistence and Backup
15
Creates @Please_Read_Me@.txt from "r.wnry"
16
Kill Processes
17
Runs: @WanaDecryptor@.exe co
Write to .res file from Time
18
Runs: cmd.exe /c start /b @WanaDecryptor@.exe vs
Delete volume shadow copies

```
attrib +h .
icacls . /grant Everyone:F /T /C /Q
```

```
Global\\MsWinZonesCacheCounterMutexA
```

```
1. CheckTokenMembership
2. Run: cmd.exe /c reg add "HKCU\SOFTWARE\Microsoft\
        Windows\CurrentVersion\Run" /v <rand>
        /t REG_SZ /d "taskse.exe" /f
```

```
1. Setup @WanaDecryptor@.exe.lnk
2. Run and Delete <randominteger>.bat:
        @echo off
        echo SET ow = WScript.CreateObject("WScript.Shell")> m.vbs
        echo SET om = ow.CreateShortcut("@WanaDecryptor@.exe.lnk")>> m.vbs
        echo om.TargetPath = "@WanaDecryptor@.exe">> m.vbs
        echo om.Save>> m.vbs
        cscript.exe  //nologo m.vbs
        del m.vbs
```

```
1. Runs :
    taskkill.exe /f /im Microsoft.Exchange.*
    taskkill.exe /f /im MSExchange*
    taskkill.exe /f /im sqlserver.exe
    taskkill.exe /f /im sqlwriter.exe
    taskkill.exe /f /im mysqld.exe
```

```
1. ---\t%s\t%s\t%d\t%I64d\t%d
2. taskhsvc.exe TaskData\Tor\taskhsvc.exe
```

```
1. Runs :
    /c vssadmin delete shadows /all /quiet &
    wmic shadowcopy delete &  bcdedit /set
    {default} bootstatuspolicy ignoreallfailures &
    bcdedit /set {default} recoveryenabled no &
    wbadmin delete catalog -quiet
```

Figure 5.2: WannaCry Execution Flow

The spread model used for WannaCry in this research is the SIR model. The important data we needed were the total number of susceptible nodes, the infected nodes and the recovered nodes. But it is difficult to get all the data when the attack is in progress and one could only make an estimation. The estimates were then used to compute the rates of change of susceptible, infected and recovered nodes as described in Section 5.1.

The spread model gives a perception of the vulnerabilities being exploited from the Delta value (δ), the weaknesses of the malware including the *kill-switch*, and the recovery rate from the Gamma value (γ) once the ransom is paid. All these information from the spread model gives the defender useful insight of the behavior of the malware, which helps in understanding the observable features so as to predict its clandestine states, to build a Hidden Markov Model based detection tool.

In order to perceive the IoCs, we assume there are "n_s" susceptible nodes and "n_i" infected nodes and "n_t" total nodes, then the total recovered nodes will be $n_t - (n_s + n_i)$. So, the fraction of susceptible, infected and recovered nodes are given as follows:

$$Fraction\ of\ Susceptible\ Nodes = n_s/n_t$$

$$Fraction\ of\ Infected\ Nodes = n_i/n_t$$

$$Fraction\ of\ Recovered\ Nodes = (n_t - (n_s + n_i))/n_t$$

- The *Fraction of Susceptible Nodes* gives an estimate of the vulnerable nodes which might come under attack. The total number of susceptible nodes is estimated to be a few orders of magnitude more than that of the total number infected nodes. Through further investigation, one can approximate the type of resources under attack.

- An analysis of *Fraction of Infected Nodes* and *Fraction of Recovered Nodes* would help the defender to gauge the types of vulnerabilities being exploited for the attack.
- If we observe the attack for a brief period, we can obtain the δ and the γ values. From these values we can calculate the rates of change of susceptible, infected and recovered nodes.
- This would help us to calculate the time taken for recovery if ransom is paid, and/or if there are other ways to tackle the attack on the system including the existence of a kill-switch (if there exists one). The information about the resources under risk can be obtained from analysis of the fractions of susceptible and infected nodes.
- We then can calculate the emission probability matrix that accounts for the behavioral aspects manifested by the attacking malware while interacting with the system.
- Moreover, the information regarding the δ and the γ values gives us the R_0 value. R_0 is the number of nodes that are at risk because of one infected node and is given by $R_0 = \delta/\gamma$.
- The δ and the γ values denote the intrinsic nature of the malware and are "generally" different for different malware. But if they are same or similar in value for two different malware, then there is a very high probability that they belong to the same family of APTs and/or are created by the same attackers.

- Using the aforementioned information, the systems on which the attack is on-going, the state of the malware can be discerned, which gives the information about the stage of the attack and a suitable defense strategy can be tailored for that particular stage of attack
- Table 5.1 gives us the parameters required for both the spread model and the HMM based detection model.

Table 5.1: Parameters for Spread Model and the Detection Model

Spread Model	HMM based Detection Model
Number of Susceptible Nodes [S(t)]	Initial Probability Distribution (π)
Number of Infected Nodes [I(t)]	Emission Probability Matrix(B)
Number of Recovered Nodes [R(t)]	Transition Probability Matrix(A)

The first step is the identification of the observable features/states (as discussed in Section 5.2.1).

- x_1 in this case would flag any process or program searching for the *EternalBlue* vulnerabilities if at all they exist in the system.
- x_2 would flag any process that is actually interacting with the SMBv1 vulnerabilities [9]. It can also denote any process that is interacting with honeypots with similar vulnerabilities.
- x_3 feature manifests the existence of *DoublePulsar* back-door implant tool in the system and/or existence of a back-channel communication between the malware and its command and control (C&C) centers.
- x_4 feature denotes the “Campaign Abort” strategy by the malware if it finds itself in a sand-boxed environment.

- x_5 feature is a bit tricky to predict or discern before it has actually been manifested by the attackers. In the context of WannaCry this can be the DDoS attack mounted on the server that hosted the “Kill-Switch” [99].

Once we have the observable features, we can use the HMM based detection system to predict the hidden/latent states for the WannaCry campaign.

- z_1 denotes the state where it can be any malware.
- z_2 denotes the state where it has manifested the features of being a ransomware.
- z_3 signifies the state where the ransomware has qualified to become an APT with primary intention of executing ransomware attack or aborting the campaign upon discovery (which in this case is a “do nothing” strategy when the malware “believes” that it is being run in a sand-boxed environment).
- z_4 manifests the intention of the attacker of executing some other form of attack as a contingency plan of attack. In the context of WannaCry the contingency plan of attack is the DDoS attack mounted in the server hosting “Kill-Switch.”

The HMM based detection model takes over once we have the emission probabilities of initial compromise from the spread model. The initial probability distribution of the malware would depend on the APT group which created the malware. Initially, the origin of the attack may not be known, so we use a standard probability distribution from past attacks of similar nature. As and when the details of the attack are revealed, we use the proper probability distribution values of the malware attack created by a particular APT group and then modify the distribution if needed. The transition probabilities for different hidden states are characteristics of the creators of the malware, which in our case are the APT groups. To begin with, our model uses standard transition probabilities learned from similar attacks in the past based on the emission matrix. The emission probabilities for each state will be given as stated in matrix B in Section 5.2.1 and the transition probabilities will be given as stated in matrix A in Section 5.2.1.

In order to show the working of our model in the absence of experimental data, we make some assumptions regarding the probability matrices, with WannaCry as the malware. The transition probabilities manifest the transition of the malware from one state to another given the last state. WannaCry would behave simply as a malware in the first state. But it is essentially a ransomware, hence it would behave like one rather than be a simple malware. Therefore, the transition probability of WannaCry will be considerably higher for the transition from State 1 to State 2, wherein it is a ransomware, as compared to the transition to State 1 from itself. Inherently WannaCry is an APT type ransomware. So, the transition probability of WannaCry will be considerably higher for the transition from State 2 to State 3, wherein it behaves as an APT, as compared to the transition from State 2 to itself. In State 3, WannaCry behaves as APT and has higher advantage if it remains in this state. The pay-off for the malware is highest in this state until it is discovered. On being discovered, it makes a transition to State 4, at which point it executes a contingency plan of attack. The malware on assuming that it has been discovered, makes a transition to State 4. Once in State 4 it either aborts the campaign or executes a contingency plan of attack. Therefore, WannaCry in State 4 remains in State 4. With these assumptions, we created an illustrative transition probability matrix as shown in Table 5.2.

Table 5.2: Illustrative Transition Probabilities for Each Hidden State

State to State Transition	Probability
State 1 to State 1	$p(z_{k+1} = 1 \vert z_k = 1) = \alpha_{11} = 0.3$
State 1 to State 2	$p(z_{k+1} = 2 \vert z_k = 1) = \alpha_{12} = 0.7$
State 2 to State 2	$p(z_{k+1} = 2 \vert z_k = 2) = \alpha_{22} = 0.4$
State 2 to State 3	$p(z_{k+1} = 3 \vert z_k = 2) = \alpha_{23} = 0.6$
State 3 to State 3	$p(z_{k+1} = 3 \vert z_k = 3) = \alpha_{33} = 0.8$
State 3 to State 4	$p(z_{k+1} = 4 \vert z_k = 3) = \alpha_{34} = 0.2$
State 4 to State 4	$p(z_{k+1} = 4 \vert z_k = 4) = \alpha_{44} = 1$

We now look for the observable features which are manifested with certain probabilities given by the emission probability matrix. The observable features in this context are given by the feature set X in Section 5.2.1. In State 1, WannaCry behaves mostly as a generic malware, so the feature which is manifested with highest probability is x_1. It, being a ransomware, features x_2 and x_3 are expressed with lower probabilities. The features x_4 and x_5 may not be expressed at all. In State 2, WannaCry being a ransomware, will manifest the feature x_2 with the highest probability and other features will be manifested with lower probabilities. In this state, it has not graduated to express all the features of an APT malware, therefore, feature x_5 will be manifested with lowest probability. In State 3, it has graduated to express all the features of an APT, so all the features will be observable with certain probabilities. In State 4, WannaCry is an APT malware with a contingency plan of attack. Thus, in this state, the feature which will be observable with highest probability is x_5, while the remaining features will be manifested with certain probabilities lower than that of feature x_5. The resulting emission probability matrix is illustrated in Table 5.3.

The initial probability distribution of the states of a malware created by an APT group generally depends on the group. An APT group can create any ran-

Table 5.3: Illustrative Emission Probabilities for Each Hidden State

	x_1	x_2	x_3	x_4	x_5
State 1	0.65	0.25	0.1	0	0
State 2	0.2	0.4	0.3	0.1	0
State 3	0.1	0.3	0.3	0.2	0.1
State 4	0.1	0.2	0.2	0.2	0.3

dom malware which can be used to mount targeted attacks on certain industry and/or institution. Therefore, the initial probability matrix will have the highest probability for State 1 for all types of malware. In the context of WannaCry, State 2 would have the second highest probability. The states 3 and 4 would have lower probabilities. The resulting initial probability matrix is shown in Table 5.4.

Table 5.4: Illustrative Initial Probability Values for Each Hidden State

State	Initial Probability
State 1	0.6
State 2	0.2
State 3	0.1
State 4	0.1

Once we have all the three matrices for the HMM based ransomware detection model, we can use either of the Forward-Backward Algorithm, Baum-Welch Algorithm or Viterbi Algorithm to predict the sequence of states for the malware and the defender can tailor a response based on the outcome of the algorithms [44], [100], [101], [102].

Through detailed experiments, the transition probability matrix, the emission probability matrix and the initial distribution matrix can be calculated and put to application in the real world scenario. Every time the status of the malware is detected, a cost-effective countermeasure could be deployed. In the context of WannaCry, the following are the countermeasures that could be employed once the status of the malware is known:

- When it is at the state of malware, simple patching of the system would help. Microsoft had released a patch update as soon as it had learned of the vulnerability.
- When the malware is graduating to become a ransomware then backing-up of the important databases would help.
- As the ransomware graduates to become an APT, blocking back-door traffic along with patching the system as well as maintaining a back-up of the database would help. Also triggering the "Kill-Switch" might help.
- In the final state, APT proceeds to execute the contingency plan of attack which in this case is the DDoS attack mounted on the server hosting the "Kill-Switch." The countermeasure in this case is all the countermeasures applicable for the previous state as well as another defensive action would be to protect the server which hosts the "Kill-Switch."

5.3 ML and NLP based IDS

Intrusion Detection Systems (IDS) are responsible for identifying any form of infiltration in the system with malicious intent. In this section, we present the design of an IDS to detect intrusions by identifying malicious processes. Microsoft Windows systems capture various metadata about a process that is running such as process name, description, command line system call executed, operation performed, and so on. Analyzing this metadata can give critical insights on determining whether the process is malicious or not. Analyzing the metadata using ML and NLP models will help in the detection of malicious processes in the system which stealthily disguise themselves as legitimate processes. This can be viewed as a classification task where a system call metadata can be classified into “Malicious” or “Non-Malicious.” The IDS developed in this section, uses the aforementioned approach of using the system call metadata to differentiate between malicious and non-malicious processes. We used a few ML models and one NLP model for the classification task. The IDS has been designed to identify APT type ransomware.

The designing of AI based IDSes required a dataset to be used for the purpose of training, testing and validation. For this purpose, we used the simulator provided by Picus Security [103]. The simulator provided us with real-world APT type ransomware and we ran that inside the Windows sandbox environment. We then collected the system calls metadata and created the dataset to be used for designing the IDSes.

The ML based IDS was designed using multiple ML models and the results were compared. The models used were NBC, LR, RF, SVM and GB decision tree. These models were trained using the dataset created from the simulator. The TF-IDF based vectorization was used to feed the data for the Naive Bayes Classifier and the word embeddings based vectorization was used to train the models SVM, RF, GB decision tree, and LR.

The NLP based IDS was designed using a BERT model for sequence classification. This transformer model contains the bare BERT Model architecture with a linear layer on top of the pooled output layer. This linear layer can be trained for sequence classification. The BERT Model is loaded from the pre-trained model configuration that is available open-source. The weights/parameters for the bare BERT Model architecture is instantiated from the pre-trained model and the liner layer on the top can be fine-tuned and the weights can be trained with the dataset available for the classification task.

Figure 5.3 is the representation of the BERT architecture for sequence classification. This model has an encoder layer consisting of 12-layers, 768 hidden nodes in each layer, 110M parameters in total (only 12-layers are available as the "bert-base-uncased" pre-trained model is utilized). Each BertLayer internally has 3 sub-layers, Attention (BertAttention), an intermediate representation layer (BertIntermediate), and an output layer (BertOutput). The input to the model is given in form of BertEmbeddings in the embeddings layer. The pooler layer is responsible for taking the output corresponding to the token and using it for further downstream tasks (such as classification, sequence generation and so on). Finally, to perform the classification, the architecture has a linear layer on the top that takes the output from the pooler layer as an input and has 2 output features since we need to perform a binary classification.

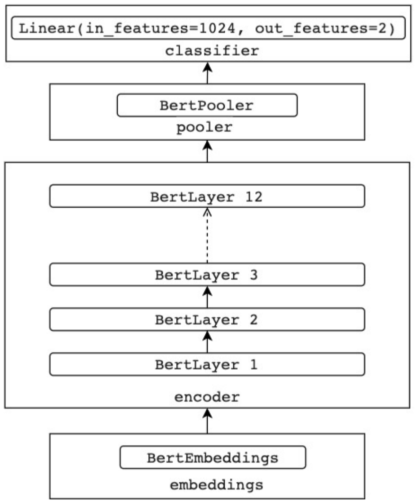


Figure 5.3: BERT Architecture for Classification

Figure 5.4 represents the various machine learning classifiers and the NLP model used in the IDS design. The BERT based classification model is compared with the machine learning models - Logistic Regression, Gradient Boosting Decision Trees, Random Forest, Naive Bayes, and Support Vector Machine and the performance metrics of these models are compared.

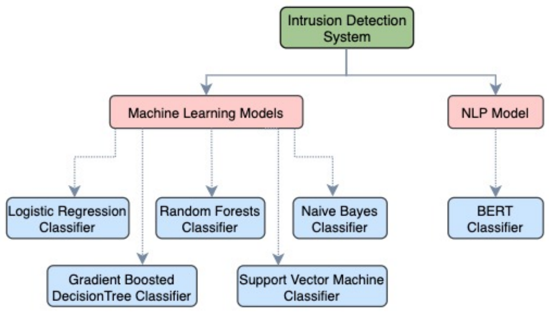

Intrusion Detection System
Machine Learning Models
NLP Model
Logistic Regression Classifier
Random Forests Classifier
Naive Bayes Classifier
BERT Classifier
Gradient Boosted DecisionTree Classifier
Support Vector Machine Classifier

Figure 5.4: Models used in the IDS design

5.3.1 Experimental Setup

The experimental setup consists of the following steps:

1. Ransomware attack simulation in a sandbox environment.
2. Capture system calls, metadata of non-malicious processes and the system calls during the ransomware execution.
3. Create the dataset from the system call and process metadata dumps.
4. Develop the machine learning models for classification of malicious/non-malicious calls.
5. Evaluate and compare the models.

Figure 5.5 gives a pictorial view of the experimental setup beginning with the ransomware simulation to the final step of model evaluation.

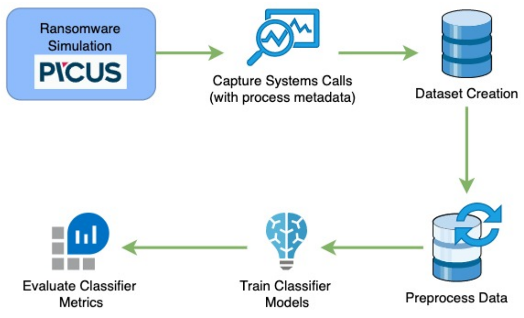


Figure 5.5: Experimental Setup

5.3.1.1 Ransomware & Simulated Environment

Picus Security is a company providing solutions on breach and attack simulation, security posture management, and complete security control validation. The platform allows users to perform Threat Emulation, Detection Analytics, Security Control Validation, and Mitigation. The platform has an extensive threat database and attack scenario simulations for various types of malware. It contains simulations that are multi-phased attacks with several tactics. The platform contains various attacks which denotes the different stages in an Unified Kill Chain and contain multiple attack scenarios as defined in the MITRE ATT&Ck tactics. They also have sophisticated and APT type ransomware in their database. This allows us to simulate APT type ransomware that carry out multi-staged attacks with various objectives. In our controlled experiment inside the sandbox, we used few APT type ransomware, viz. BlackByte, BlackMatter, Diavol, Darkside, and REvil.

The simulation was set up in a Dell OptiPlex 7010 system with Intel core i7-3770 @ 3.40GHz processor, 16.00 GB memory running a 64-bit Windows 10 Education OS. The system had Intel HD Graphics 4000. The simulations were run in a safe sandbox environment. We used Windows Sandbox application feature (available from Windows 10). Windows Sandbox is a lightweight environment for desktop intended for safely running software in isolation. The configuration of spawned Windows Sandbox was Intel core i7-3770 @ 3.40GHz processor, 4.00 GB RAM running a 64-bit Windows 10 Enterprise OS. The Picus platform requires the installation of an agent, which is used to run the simulations. This agent was installed in the Windows Sandbox and all simulations were run inside of it. After the agent is installed, the platform aids in simulating real-world attacks and threats against our system. It can simulate various exploits and attacks that operationalize the frameworks such as MITRE ATT&CK techniques. Furthermore, the ransomware simulations are made of independent adversary techniques (mapped to MITRE ATT&CK) but these scenarios (exploits, attacks) do not run any malicious code but notepad-like "safe" apps to prove code execution, i.e., it does not actually lock a user out of a system or encrypt the entire system and ask for a ransom to regain access. Instead, it performs these in a controlled manner on minimal files/volumes and uses notepad-like applications to notify the user that the exploit or attack was successfully carried out. It is like a scaled down version of the ransomware that can still execute and simulate the attacks or exploits but in a less destructive fashion. This allows us to capture enough data on the various kinds of actions performed by a ransomware such as encrypting the drives (files, volumes) or deleting the shadow copies or unblocking access to files and deleting them from the system. After each attack simulation, the Picus agent follows up with clean-up functions to restore, clean-up the OS to the last known state. For designing the IDS and performing our experiments, we needed the dataset. The creation of the dataset and preprocessing of the data are described in the next section.

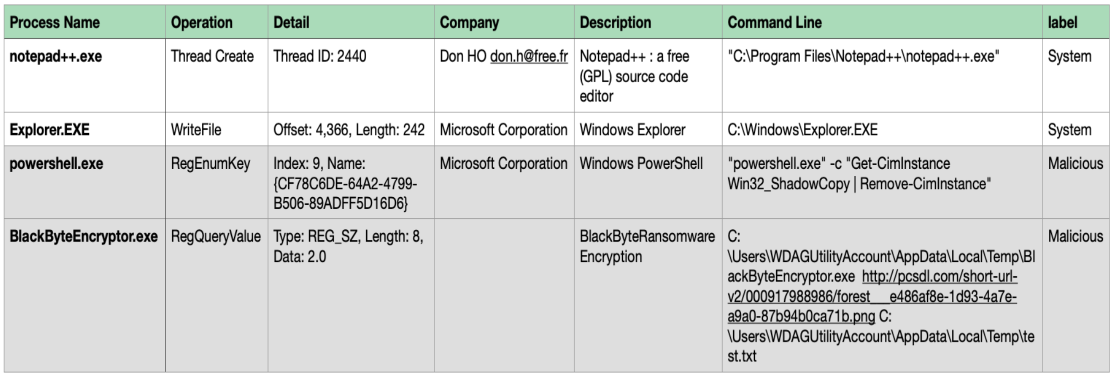
5.3.1.2 Dataset

Windows Sysinternals offers a tool called Process Monitor (Procmon). It is an advanced monitoring tool, that shows real-time file system, registry, and process/thread activity. It can capture reliable process data such as process name, command line system calls executed, user ID, operations performed, description, company name, and many more. It can capture these details for every process that is running in the system even if it is spawned in the background. This extensive metadata information can be dumped into a csv file. When a ransomware is running, it executes multiple malicious processes that perform actions such as encrypting a file, deleting shadow copies, deleting files, and so on. Procmon is used to capture a snapshot of the system while simulation is running. This contains all the metadata about the processes running during the ransomware simulations. Similarly, to capture the non-malicious and default system processes, Procmon is used to capture all the process events when the simulation is not triggered. This captures all the processes that run in the system in the background and user actions are performed to capture the normal state of the system. This involves performing few user actions such as copying files, create and delete text files, opening few applications such as notepad and browser. These Procmon event details are converted into csv files and stored.

The process events captured during ransomware simulations and the ones captured without the ransomware attacks are used to create the dataset for our classification tasks. The features of interest in these events are the Process Name, Operation, Detail, Description, Command Line, and Company. The events captured with no ransomware attacks are de-duplicated and labelled as 'System', i.e., non-malicious and this dataset is referred to as System dataset. The events captured during the ransomware simulation cannot be marked entirely as 'Malicious' as background Windows processes and other non-malicious processes would still be running. Hence, the events captured in the System dataset are referenced to identify the malicious processes. Every event in the data captured during ransomware simulation is compared with the events captured for the System dataset. Based on the uniqueness of the Process Name, Command Line call executed, Operation and Company features, the events are tagged as 'Malicious.' For instance, if all of these features are already available in the System dataset, then that event is ignored since it denotes that it is a background system event. If any of these four features are not already available in the System events dataset, then that particular event is tagged as Malicious. The Detail and Description features are not taken into account while this tagging is performed. If any process event has Detail and Description values which are not there in the System dataset, they may or may not be malicious in nature. If the four features Process Name, Operation, Command Line, and Company denote a process event as safe then those events, even with different Detail and Description features, can be labelled as non-malicious in nature.

The dataset is tagged as 'System' and 'Malicious' as explained above. It is used to build the classification model to identify malicious process events. The dataset created contains 7 columns viz., Process Name, Operation, Company, Detail, Command Line, Description, and Label. It contains a total of 90,841 rows with 43,487 events tagged as 'Malicious' and 47,354 events tagged as 'System.' In the following subsection, we elucidate the pre-processing of the data to make it suitable for the training of the AI models.

Figure 5.6 shows a sample of the created dataset with four rows. The first two rows show non-malicious System processes that were captured when no



Process Name	Operation	Detail	Company	Description	Command Line	label
notepad++.exe	Thread Create	Thread ID: 2440	Don HO don.h@free.fr	Notepad++ : a free (GPL) source code editor	"C:\Program Files\Notepad++\notepad++.exe"	System
Explorer.EXE	WriteFile	Offset: 4,366, Length: 242	Microsoft Corporation	Windows Explorer	C:\Windows\Explorer.EXE	System
powershell.exe	RegEnumKey	Index: 9, Name: {CF78C6DE-64A2-4799-B506-89ADFF5D16D6}	Microsoft Corporation	Windows PowerShell	"powershell.exe" -c "Get-CimInstance Win32_ShadowCopy | Remove-CimInstance"	Malicious
BlackByteEncryptor.exe	RegQueryValue	Type: REG_SZ, Length: 8, Data: 2.0		BlackByteRansomware Encryption	C: \Users\WDAGUtilityAccount\AppData\Local\Temp\BlackByteEncryptor.exe http://pcsdl.com/short-url-v2/000917988986/forest__e486af8e-1d93-4a7e-a9a0-87b94b0ca71b.png C: \Users\WDAGUtilityAccount\AppData\Local\Temp\test.txt	Malicious

Figure 5.6: Dataset Sample

ransomware was run in the Windows sandbox. The subsequent two rows show Malicious processes that were captured as part of the ransomware simulation in the sandbox. The first row depicts the notepad++.exe application running in the system. The data row captured is a thread create event triggered by the notepad++ application process. The other columns of that row show the meta-data related to this process. The second row depicts a Windows Explorer.exe application running in the system. An event related to file write is captured and this is a benign data row as well. The third row represents a malicious process running in the system. This was triggered as part of a ransomware and it is using the powershell.exe application to carry out an unwanted operation in the system. The shadow copies are storage extents that are duplicate copies of the original volumes and can be used for back-up/restore in case of system failures. Using the powershell.exe application, the ransomware is trying to delete the shadow copy of the volumes in the system. We can see in the command line that it is using a combination of Get-CimInstance and Remove-CimInstance scripting tools of the powershell.exe to carry out its malicious task. The last row of Figure 5.6 represents a malicious process named BlackByteEncryptor.exe which is part of the BlackByte ransomware campaign. As the name suggests, this is a dangerous process that is trying to encrypt the files in the system. Encrypting the files, volumes in system is a common technique used by ransomware to lock the user out of the system and prevent access to the drives (system storage). The command line columns of the fourth row shows how this encryptor process is trying to encrypt a text file available in a certain location.

5.3.1.3 Data Preprocessing

The dataset contains seven columns as depicted in Figure 5.6 with six columns for features and one column representing the label. Each of these six features are text based data. The columns Process Name and Operation are categorical features, and the columns Detail, Company, Command Line, Description are text data. These text based features cannot be used directly for training the models. Instead they need to be converted into numerical features. The categorical features are converted into onehot encoding and the text data columns are converted into vectors of word embedding. To get the word embedding, pretrained Stanford Glove word embeddings are used [104]. The pretrained word embedding provide vector representation for several words. The package glove.6B.300d, that contains pre-trained word vectors with 6B tokens, 400k vocab with a vector of size 300 was used. The Out-Of-Vocab words (any word for which vector representation is not available in the package) is initialized to a random vector of dimension 300 and stored, so that the same vectors are used for a word. The sentences are converted to vector by average pooling the word embedding of each word in the sentence to get a final sentence embedding. Finally, to reduce the number of dimensions, the sentence vector of each of the feature is again average pooled to get one final vector of size 300 that represents the four columns. The onehot encodings of the categorical columns are merged and combined into one numerical array and further this is extended with the vector representation of the four text columns to get a numerical array of dimension 457 that represents one event in the dataset. This is used to train Naive Bayes, SVM, Random Forests, Gradient Boosting Decision Tree, and Logistic Regression models.

The dataset is processed in a different manner for the Naive Bayes model. The features are joined by whitespace into a single text sequence and is converted into meaningful representation of numbers based on the TF-IDF (Term Frequency Inverse Document Frequency) vectorization method. This is a method that takes into account the relevance and importance of words in a corpus (collection of documents which is the collection of all text sequences in this case). The relevance and importance of words are derived from the occurrence count of that word in a document and the number of documents in which the word occurs in. Each text sequence is converted into a vector of length 17,125 (this is the number of unique tokens in the document corpus).

In the case of BERT model, the input is expected to be a text sequence. Since, each of the feature is of type text data, all the features are joined by whitespace into one text sequence which is fed to the BERT based tokenizer for encoding. The BERT tokenizer can then use this text sequence to perform extract word pieces that is encoding into numerical values and can be fed into the model.

5.3.1.4 Model Training and Evaluation

The IDS requires to perform a binary classification to identify a system call along with other metadata as malicious or non-malicious. Since this is a binary classification task, models such as Naive Bayes, Support Vector Machines, Logistic Regression, Random Forests, and Gradient Boosting Decision Trees classifiers are trained using the dataset. We also trained NLP based BERT model on the same dataset. We then compared the results from all the aforementioned AI models.

The preprocessed dataset was split into training, validation and test sets in the ratio 7:1:2 respectively. This means 70% of the data was used to train the model, 10% of the data was used for validation, and the remaining 20% was used for testing and computing the accuracy metrics of the models.

The Machine Learning models Naive Bayes, Support Vector Machines, Logistic Regression, Random Forests, and Gradient Boosting Decision Tree were used. For these models, the entire training data was introduced during the training stage. The validation and test sets were used to evaluate the model after the training qas complete. These models were trained and evaluated on an Apple MacBook Pro with M1 chip processor and 16GB memory.

For NLP based BERT model, similar split of the dataset was done with 70%, 10% and 20% of the data for training, validation and testing respectively. The BERT based model for sequence classification BERTForSequenceClassification was used for the task. This model was a slightly modified version of the BERT base model with a linear layer over the pooled output layer that was fine-tuned and trained for the classification task. The BERT base model was loaded from a pretrained model that was available open-sourced "bert-base-uncased." This is a pretrained model over the English language (Wikipedia and Google BooksCorpus) and it is a case insensitive model. This pretrained model can be used to load the models' weights, parameters, and configuration for the BERT base model encoder. The linear layer on the top intended for classification was loaded with random weights. The entire model (the embeddings layer, the encoder layers, and the linear layer on top) was tuned and the weights were updated as part of the model training step with the created dataset in hand. As part of the training step, the data was fed to the model in batches of 8 and the model was trained for 4 epochs. The BERT model was trained and evaluated in Google Colab using a GPU. The Nvidia Tesla T4 was available in Google Colab for training and testing the BERT model.

The trained models were then evaluated against the validation and test dataset. Both of these datasets were unseen by the model before this step and each event in these datasets was classified as malicious or non-malicious using the models. The classification performed using the model was compared with the actual labels of these events. The predicted labels and the actual labels are assessed to get the evaluation metrics that allowed us to compare and evaluate the models.

5.3.2 Results and Discussion

After performing the experiments and generating the predictions on the test dataset of size 18,169 records, we compared the results on the basis of F1 score, Accuracy, and the ROC plots.

Table 5.5 summarizes the performance of each classifier in terms of accuracy and F1 scores. We can see that the BERT classifier has the highest accuracy and F1 score followed closely by the Naive Bayes classifier. These were followed by Gradient Boosting Decision Tree, Random Forests and Logistic Regression classifiers in the order of decreasing accuracy. The Support Vector Machine classifier had the least accuracy and F1 score.

Figure 5.7 shows the receiver operating characteristic (ROC) curves plotted for each model. The area under the curve (AUC) computed for BERT was 1 indicating that is able to distinguish between the two classes clearly. The AUC

Table 5.5: Model Accuracy and F1 Score

Classifier	**Accuracy (%)**	**F1 Score**
BERT	99.98	1.00
Naive Bayes	98.55	0.984
Gradient Boosted DT	97.21	0.971
Random Forest	96.04	0.958
Logistic Regression	94.47	0.943
SVM	74.51	0.694

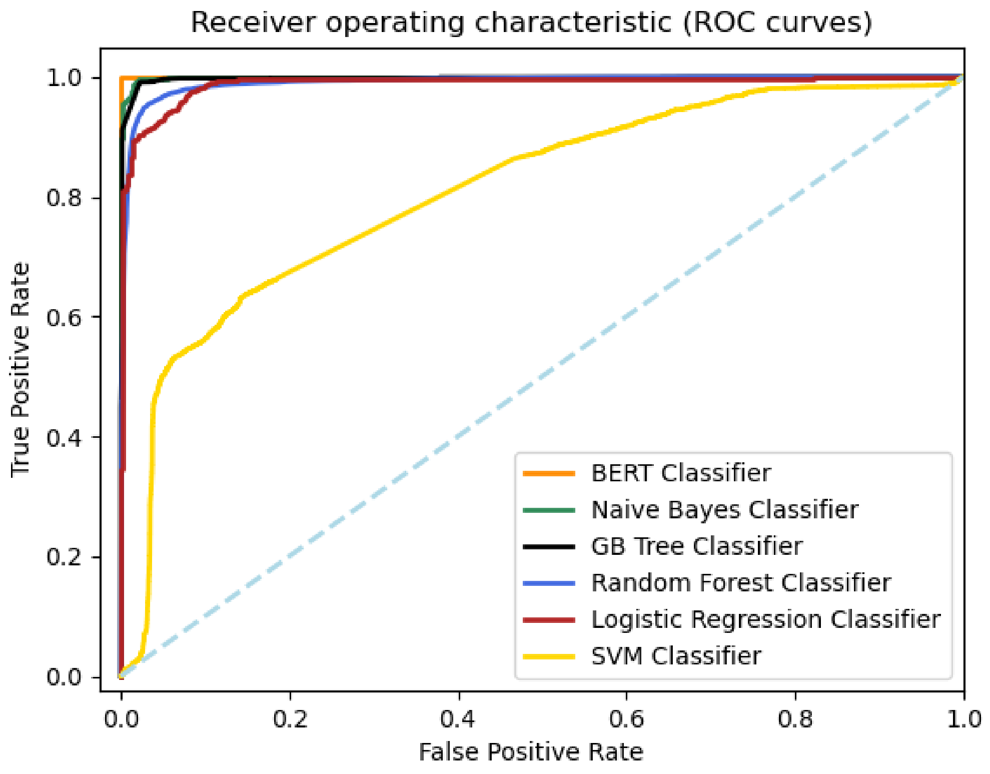


Figure 5.7: ROC curves for the Classifiers

values for Naive Bayes (NB) and Gradient Boosting (GB) Decision Trees were almost equal to 1. This follows a similar trend to the accuracy values and from Figure 5.8, we can notice that, though Naive Bayes model and GB Decision Tree model curves seem similar, the green curve representing Naive Bayes covers more area than the black curve representing GB decision tree. These models

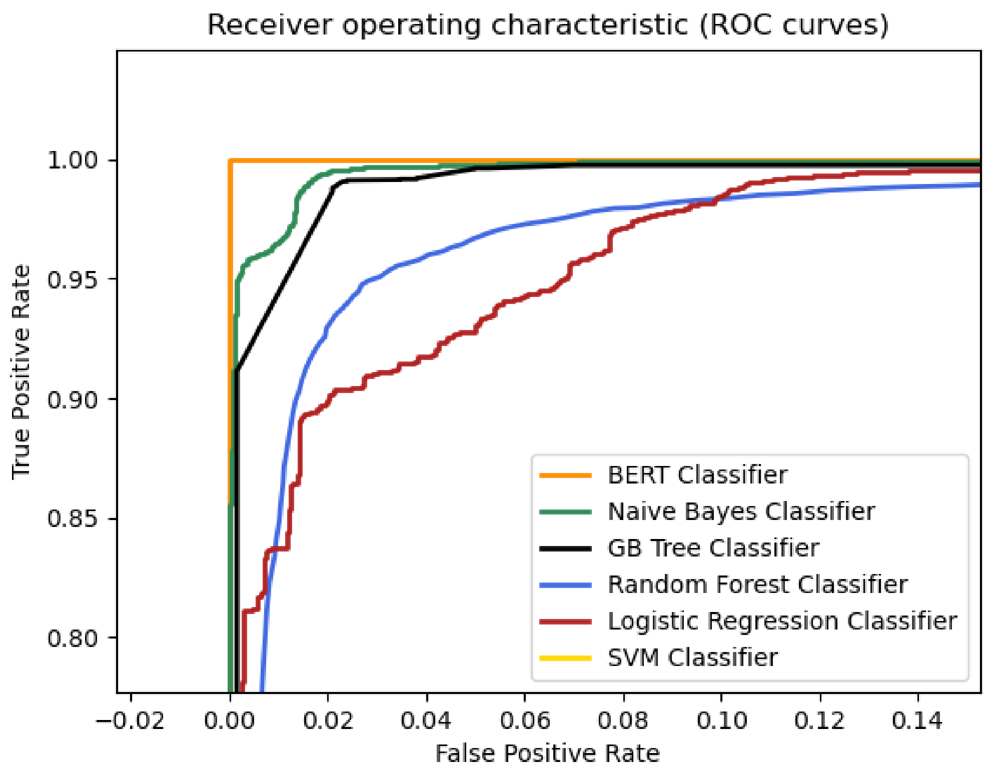


Figure 5.8: Zoomed view of the left Top part of ROC curves

are followed by Random Forest and Logistic Regression models in decreasing values of AUC. The support vector machine classifier represented by the yellow curve in Figure 5.7 had the least area under the curve denoting its poor classification performance on this dataset.

Of all the classification models used, NLP based BERT model had the best performance, being able to predict the labels for the test set with a near perfect accuracy. The next best performing model was Naive Bayes Classifier. This was followed by Gradient Boosting Decision Tree, Random Forest Classifier and Logistic Regression Classifier in the order of decreasing accuracy. The model with the least scores was the SVM classifier. On comparing the models with respect to the training time taken as shown in Table 5.6, we can observe a contrast with

BERT taking the longest time for training. BERT model took over six hours for four epochs of training. This was followed by SVM classifier which took 5,190.30 seconds, Naive Bayes Classifier took 21.92 seconds, Gradient Boosting Decision Tree took 628.84 seconds, Random Forest Classifier took 98.65 seconds and Logistic Regression took 6.7 seconds.

Table 5.6: Classifier Models and their Training time

Classifier	**Training Time**
BERT	6 hrs
Support Vector Machine (SVM)	5190.303s
Naive Bayes Classifier (NBC)	21.92s
Gradient Boosted Decision Tree	628.84s
Random Forest	98.65s
Logistic Regression	6.7s

5.4 Summary

In summary:

- This chapter introduced an HMM based IDS to detect APT type ransomware.
- The idea is to discover the state of the attack. This would help the defender to devise a defense action best suited for that state of attack.
- HMM based IDS not only detects the attack but also characterizes the attacker which helps ins preparedness against future attacks. More details can be found in [85], [86].
- Classifier based IDS was also designed to detect APT type ransomware. For the purpose, ML and NLP based classifiers were used.
- These classifiers detect APT type ransomware from system call data.

- The plan was to differentiate malicious system processes from non-malicious system processes and detect the APT type ransomware.
- Both the IDSes were designed for APT type ransomware but are equally effective against a basic variant of the ransomware.

The IDSes were designed to identify APT type ransomware. But as mentioned in the works of the Roman poet Juvenal, a famous question comes to the front, *Quis custodiet ipsos custodes?* [105]. In English, it means "Who will guard the guards themselves?" [106]. Researchers have worked on techniques to make their frameworks tamper-resistant by adding another layer of security [107],[108]. To make them tamper-resistant, the IDSes needed another layer of protection. Therefore, in order to ensure the security and survivability of the system, a deception architecture and mitigation strategies become essential, which are discussed in the subsequent chapters.

Chapter 6

Deception as a Defense Tool

"All warfare is based on deception. Hence, when we are able to attack, we must seem unable; when using our forces, we must appear inactive; when we are near, we must make the enemy believe we are far away; when far away, we must make him believe we are near."

– Sun Tzu,

Art of War

The detection systems we discussed for APT type ransomware, needed another layer of protection. This layer of protection was ensured through a deception architecture. We designed a deception architecture namely, "Kidemonas." We used commercial-off-the-shelf (CoTS) hardware component called trusted platform module (TPM) for designing Kidemonas. The idea is to run a generic APT detection system in an isolated environment outside the purview of the attacker. The aim of the architecture is to silently detect the intrusion and surreptitiously report the same to the defender. This was for detection of generic APT malware. In order to give a layer of protection to the HMM based IDS, we designed a deception based countermeasure called "Decepticon." Decepticon is a special case of Kidemonas, which runs an HMM based IDS to detect APT type ransomware as its APT detection system in the isolated environment. The distributed nature of the APT detection system ensures that the load of detecting any form of APT type malware doesn’t fall on a single system. Having a distributed APT detection architecture and surreptitious reporting has another advantage. When an attack is unfolding, and an intrusion is detected in one of the nodes of the distributed architecture, a preemptive action in other nodes can be taken to prevent the attack on those nodes. This builds an effective defense strategy when faced with an attack from an APT type malware which possess the capabilities of lateral movement and privilege escalation. This chapter introduces and explains the aforementioned deception architectures in the following sections.

6.1 Kidemonas: The Silent Guardian

Here we put forward a framework to detect generic APT attacks. Deception has been used as a potential weapon to tackle an attack by APT malware. The whole idea is to silently detect any intrusion by an APT malware and surreptitiously report the same to the user and/or the system administrator.

6.1.1 Trusted Computing: Security in Vanilla Form

Trusted computing has been thought of as a probable solution to the defense against various cyber-threats [109]. In [110] the authors tried to leverage the functionalities of trusted computing to detect any attack. They relied heavily on constant monitoring of the system and checking the integrity of the platform regularly. In the event of any intrusion being detected, the system raises an alarm. This alarm is also visible to the attacker. The attacker then can change the mode of attack to take some aggressive measures or can resort to some other form of contingency plan for which the defender may not be prepared and the system can suffer unexpected damage. In [111] the authors present a model wherein the TPM is given an added functionality to communicate via the link layer communication channels to other nodes in a networked system. But it also raises an alarm whenever any form of intrusion is being detected and that alarm is visible to all. Therefore considering the implications of the aforementioned threat detection alarms, we present Kidemonas, an architecture to silently detect any threat and surreptitiously reporting it to the system administrator.

6.1.2 Kidemonas: The Architecture

Deception is an important part of the defense mechanism in cyber-security. The authors of [112] present a software security solution wherein the elements of misdirection and deception are introduced using honey-patches. But our architecture, Kidemonas, uses hardware component, as shown in Figure 3.1 to create the framework as shown in Figure 6.1, in order to introduce the deception scheme. By camouflaging the infiltration detection system, the attacker is deceived into believing that it is successful in maintaining its stealth, and it buys ample time for the defender to come up with countermeasures to thwart the attack. To maintain the cover, the traffic is analyzed in a concealed environment. Therefore, an encrypted copy of the incoming traffic is sent to our hardware based security system, and it is in this unit where the malware detection system analyzes the traffic and surreptitiously reports it to the user or the system administrator. The following subsections discuss the working of each unit as the traffic flows through it.

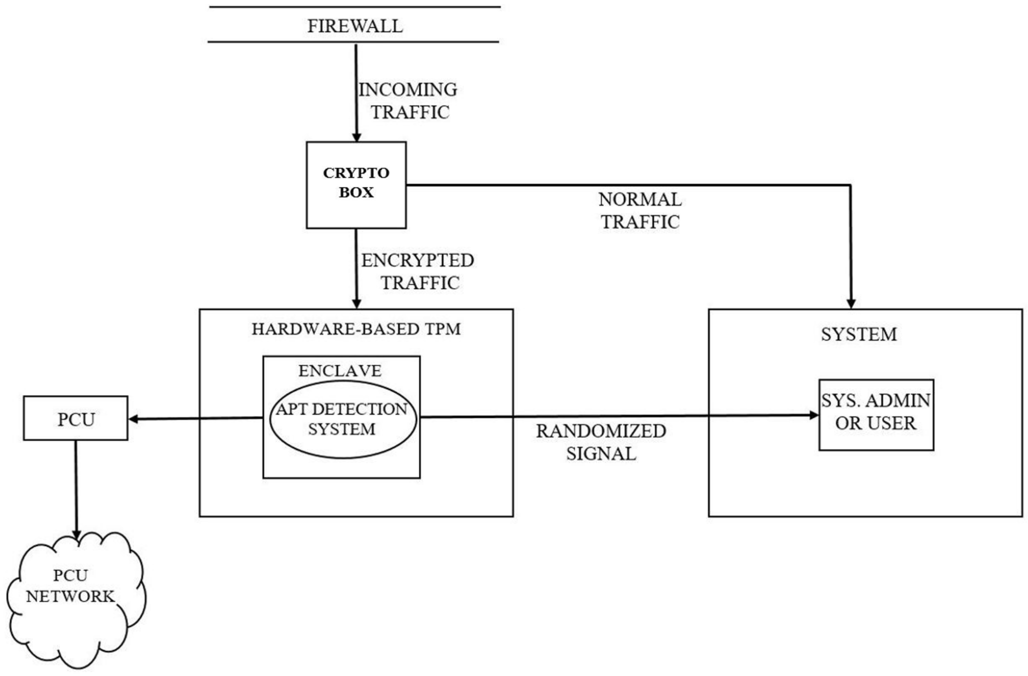


Figure 6.1: Kidemonas Architecture

6.1.2.1 The Firewall

The first line of defense of a system would be the firewall, which acts as a fence and keeps away the malicious actors from the system. But in an event of the breach of the perimeter or in this case breaching the firewall of the system can pose serious threats to the system. APT malware infiltrates the system stealthily, so it can be assumed that the attacker would be able to pass through the firewall.

6.1.2.2 The Crypto-Box

After passing through the firewall, the traffic enters the crypto-box, as shown in Figure 6.1. The crypto-box is used for encrypting the data traffic entering the system. The first task of the crypto-box would be to make a copy of the data traffic entering the system and then to encrypt the copied traffic and sending it to the hardware-based TPM. The crypto-box is configured to encrypt the copied traffic using the RSA-OAEP scheme [113], [114]. The original copy goes to the system as intended.

6.1.2.3 The Hardware-based TPM

The encrypted traffic then goes to the hardware-based security system which comprises of a hardware-based TPM unit. This unit, along with various cryptographic capabilities also provides an isolated and exclusive execution environment or the enclave, which even the high priority OS instructions do not have access without user authentication. The idea is to implement the APT Detection System within this execution environment. Any form of infiltration detected by the system, which is running inside the enclave can be communicated to other nodes in the network using the Peer Communication Units (PCUs) as shown in Figure 6.1. The encrypted traffic can be stored outside the system to further analyze the malware or the attack so that security patches can be produced at the earliest.

Any traffic going inside the TPM doesn't come-out and it behaves like a black-hole. The encrypted traffic entering the TPM, is decrypted using the private key of the TPM. For the hardware-based TPM, the security heavily depends on how safely the private keys are kept hidden. The APT Detection System running inside the TPM analyzes the received traffic and detects any form of infiltration by the APT malware, if it has taken place.

6.1.2.4 The APT Detection System

The architecture has been designed in a way that any state-of-the-art APT malware detection system, which is compatible with the system, can be installed within the concealed environment of the hardware-based TPM. Security Information Event Management (SIEM) and User Behavior Analytics (UBA) are popular APT malware detection schemes [4]. The SIEM scheme collects information from different events occurring in different components of the system, and looks for security flaws. This scheme might blow the cover of the deception mechanism. But any scheme which analyzes the incoming traffic, like big-data analytics [115], can be installed in Kidemonas for APT malware detection.

6.1.2.5 Peer Communication Unit

Peer Communication Units (PCU) are the Link-layer or Layer 2 units, which interact with other nodes in the systems via Layer 2 [111]. The PCU of one node interacts with the PCU of the other, forming a PCU network as shown in Figure 6.2 [111]. The PCU of a given node interacts only with its hardware-based TPM. Once the APT Detection System, running inside the hardwarebased TPM, detects any form of intrusion, it communicates the information to the other nodes in the system through the PCU network. The PCU network actually provides a different communication intra-network link within the system, which is inaccessible to any entity, and is not a part of the system. Even the OS running on each node won't have access to this PCU network.

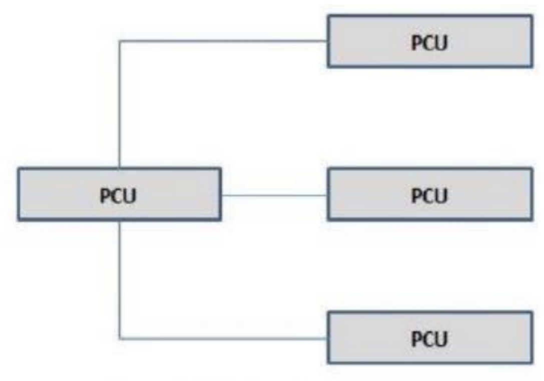


Figure 6.2: Peer Communication Unit

6.1.2.6 The Deception Mechanism

The deception mechanism comprises of an alert, which is covertly raised without letting the attacker know that an alert has been raised. It is achieved by camouflaging the alert as a normal traffic between the TPM and the user or the system administrator. The TPM would send two signals at every interval to the user or the system administrator. The two signals are: Time Stamp and a Random Signal of bit-strings of 0s and 1s. The time stamp will be in the 24-hour format. Since, most of the OS being used today use 64-bit bus, the random bit string will be 64-bits long. At the event of an infiltration being detected, the TPM will generate a time-stamp and use the hour part of the time-stamp as the recognizing bit. For example, if the hour part of the time-stamp is 09, then the bit number 9 of the random string will be set as 1 as well as, (63-9) or the bit number 54 would also be set as 1. In the event if no infiltration being detected and the hour part of the time stamp is 09, then the bit number 9 of the random string will be set as 0 as well as, (63-9) or the bit number 54 would also be set as 0. The user or the system administrator will just check the bit number n and the bit number (63-n), as the bits are numbered from 0 to 63, and would realize if there has been any intrusion or not. The infiltration detection is being camouflaged in the randomness of the bit string.

6.1.3 Analysis of the Security of Kidemonas

6.1.3.1 The Deception Mechanism

The most significant aspect of the security provided by Kidemonas is the deception mechanism. Considering the different stages of the APT life cycle, the security provided by Kidemonas is analyzed in detail. Kidemonas won't be able to deter the attackers in the first three stages. The attacker will be able to perform initial reconnaissance and infiltration, establish a foot-hold and escalate the privileges. But it won't be able to perform internal reconnaissance, and move and compromise internally as far as the Kidemonas unit is concerned. Each time the infiltrated malware receives any command or information from the C&C centers, Kidemonas would receive the same without the attacker knowing about it. Since, the attacker doesn't know about Kidemonas, and no alarms are being raised on infiltration, it would believe that it has successfully and stealthily compromised the system. Therefore, neither the attacker would become aggressive nor would they resort to the contingency plan. This would buy copious time for the defender to come up with a patch or better security features, before the attacker achieves the last stage, i.e., the mission completion stage. This would help the victim to avert any major damage.

6.1.3.2 The Crypto-Box

The Crypto-box uses the RSA-OAEP scheme to encrypt the traffic being sent to the hardware-based security unit. It could use the public-key portion of the endorsement key (EK) of the TPM to encrypt the traffic. Now, RSA-OAEP according to [116] and [114] is chosen cipher-text attack (CCA) secure under the RSA Assumption. This would camouflage the copy of the traffic going in to the security unit. Moreover, as discussed earlier, the private key portion of the endorsement key of the TPM never leaves the TPM and the security heavily depends on how the endorsement key is kept secret. Therefore it would not be possible for the attacker to have a better knowledge of the private key than a random guess.

6.1.3.3 The Trusted Execution Environment

Last but one of the most significant contributions of the Kidemonas architecture along with the surreptitious reporting scheme is that it also provides a trusted and isolated execution environment wherein any state-of-the-art APT malware detection system can be implemented. With time the APT malwares would become more resourceful and might deceive the existing malware detection systems. The APT malware detection system can be updated manually for newer and more advanced threats. This flexibility of system upgradation makes the architecture robust against becoming obsolete in the face of newer and more advanced form of threats.

6.1.3.4 The Security Overview in the Presence of a Threat

Now, taking an example of the threat model, the effectiveness of the Kidemonas architecture can be discussed. The threat model perceived in this paper is a dynamic one, which comes with a contingency plan and has the capability to resort to the contingency plan any time they deem necessary and the trigger for such a scenario is the information being conveyed to the attacker that they have been detected by the system. The attacker, if detected in the very first phase, may abort the mission. This doesn't give time to the defender to learn about the attack in details. Kidemonas, surreptitiously informs the system administrator regarding the attack, and the attacker is deceived into carrying out the attack. This gives the defender an opportunity to study about the attack, during its entire life-cycle and stop the attacker from giving the final blow in the very last stage, as by then the defender would be well armed and ready to thwart the attack. In the second stage, the APT malware installs a backdoor to communicate with the C&C centers of the attack. If discovered in this stage, the attacker can either abort the mission or resort to more aggressive means. Both the scenarios are disadvantageous to the defender. In the former case, they don't get sufficient time to study the attack and in the latter case, they are not ready for a more aggressive form of attack. But, with Kidemonas, camouflaging the infiltration report, the attacker is made to believe that no detection has taken place. This might also help the defender to be prepared not only for the current form of attack but also for any aggressive contingency plan the attacker might have. Similarly, for the later stages of the attack, the attacker is still deceived into believing that the system is unable to detect their threat. The attacker doesn't get any time to resort to the contingency plan, when the defender thwarts the attack in a well prepared manner in the final stage of the attack.

6.1.3.5 The Weakness

But no design is completely secure and each comes with its own share of weaknesses. The biggest concern for the Kidemonas architecture is the insider threat. As of now it is very hard to deal with insider attacks. In a networked system, if the attack happens from one of the nodes, then the entire system is compromised. Insider attack is a scourge that would require special investigation and is outside the scope of this dissertation.

6.2 Decepticon: Deception Based Countermeasure

The Trusted Computing Group (TCG) laid down the specifications for Trusted Platform Module (TPM) with an intention of creating a trusted computing environment [38]. These specifications were capitalized on to create a deception based architecture, Kidemonas [84], which provides isolation to malware detection systems so that the detection can occur outside the purview of the attacker and the intrusion can be surreptitiously reported to the user or the system administrator.

6.2.1 Decepticon: The Architecture

In this research the capabilities of Kidemonas are extended to realize a cost-effective system to detect intrusions from APTs. In a business enterprise, Kidemonas gives the system administrator the capability to run different forms of intrusion detection on different computing units. The information regarding intrusion is shared with the system administrator and the other computing units through a separate channel called the peer communication network. It comprises of a link-layer communicating unit present in each computing unit called the peer communication unit (PCU). A computing unit in this scenario refers to a computer or a server or basically any computing unit which forms a node in the networked system in a corporate network monitored by a single user or a single group of users working collectively for the same purpose. To make the defense strategies cost-effective, we use the smart-box proposed in [17]. The objective here is that whenever a form of intrusion is detected, it is reported to the system administrator silently, who in turn can monitor the attacker's moves and use the smart-box to trigger an appropriate defensive response from the repository. The repository is a storage unit for defense strategies that could be triggered to defend the system at the event of an attack on the system. The defense strategies range from simply blocking certain processes to defending against intricate attacks. The smart-box on learning from the nature of the attack and the status of the malware can trigger an effective response which would be economical in terms of time and resources being used. The smart-box is the decision making unit regarding defense strategies depending upon the characteristics of the malware.

Figure 6.3 represents the hardware based defense architecture called Decepticon whose aim is to deceive APT type ransomware. Kidemonas [84] is a more generic architecture to counter any APT, whereas Decepticon is a customized version to have a HMM based ransomware detection tool (which is subsumed under the Enclave in the figure and discussed in the next section), and a smart-box to trigger defensive actions depending upon the severity of the attack. If the attack is determined to be of a simple nature, the smart-box triggers a simple response to counter it, and if the attack is sophisticated in nature, then it triggers an elaborate response.

The firewall (Fig. 6.3) performs signature based detection. If the malware is able to get past the firewall along with the legitimate traffic, it reaches the crypto-box. The crypto-box makes a copy of the incoming traffic and sends the normal traffic to the system. The copied traffic is encrypted and sent to

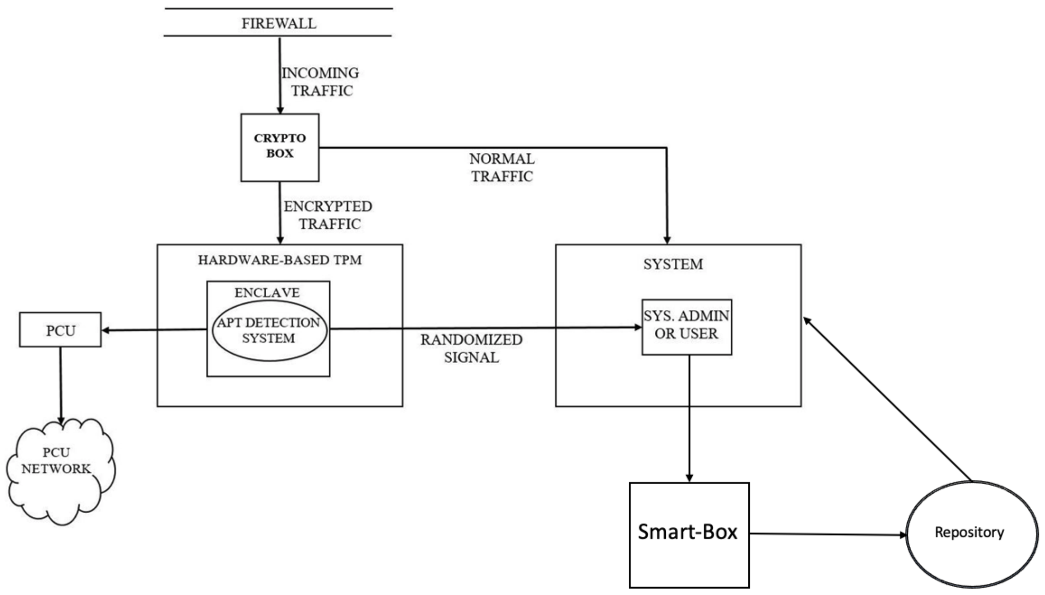


Figure 6.3: Decepticon Architecture

the hardware-based TPM. The encryption is performed using the public-key of the endorsement key of the hardware-based TPM. In the TPM, the ciphertext is decrypted using the private-key component of the endorsement key of the TPM. The analysis of the traffic is done by the HMM based detection tool. Any form of intrusion being detected is sent to the peer communication unit (PCU) and from there to the PCU network, so as to inform every node in the networked system about the form of intrusion. The PCU network is accessible only through the PCU, which in turn is accessible through the hardware-based TPM. At the same time, a surreptitious reporting is done to the user or the system administrator. The system administrator then uses the storage root keys (SRK) to gain access to the TPM to gain knowledge and the nature of the intrusion that has taken place.

The security of the entire system relies on the fact that the private key component of the endorsement key of the TPM, which was created when it was

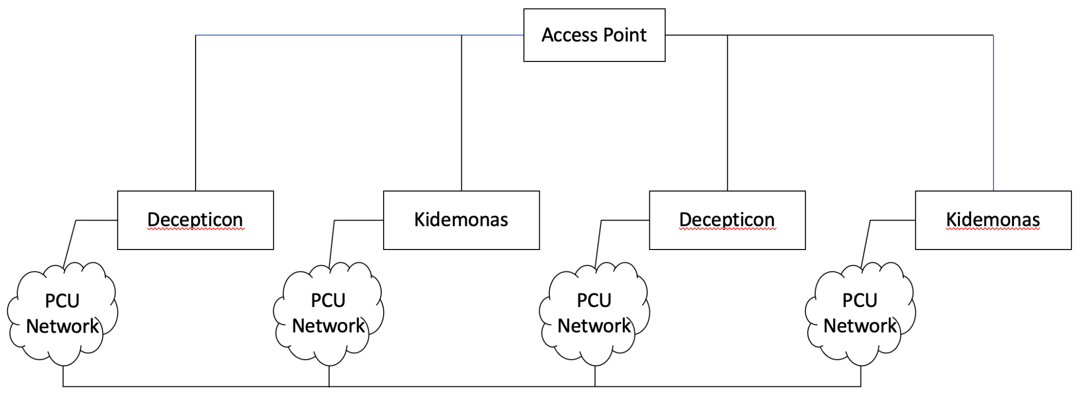


Figure 6.4: The System

manufactured, never leaves the TPM. The security also relies on the fact that the storage root keys (SRK) created by the users, when they took the ownership of the TPM, is kept safely guarded.

Figure 6.4 shows a snapshot of a networked system in a corporate environment. This representation shows multiple computing units connected to a single access point. Each computing unit is connected to other computing units through the PCU network, which is also used to inform each other of any form of intrusion in the system. Figure 6.4 shows different versions of detection tools running on different computing units; some of them running Decepticon while others are running the generic Kidemonas style APT detection tools.

6.2.2 Decepticon - In Action

The Decepticon architecture that is built upon Kidemonas makes it scalable and easy to use due to its reliance on commercial off-the-shelf components (COTS) such as the TPM. This scalability helps in future proofing of the entire system.

The transition and emission probabilities once calculated, would provide the defender with valuable information about the malware that would help the user to trigger a cost-effective response from the repository through a smart-box. The biggest advantage for the defender is *awareness*, *security* and cost-effective *countermeasure*. Once the model is put to application in the real world, it would yield numerical values for the transition and emission probability matrices. This helps the defender to make an informed decision, without compromising the quality of service of the system.

Given the scalability nature of Decepticon as shown in Fig. 6.4, it is safe to assume that there will be multiple nodes in a networked environment and each of them would be running a Kidemonas or a Decepticon type IDS individually. The intrusion detection happens outside the purview of the attacker. However, the framework presented here doesn't guarantee that the APT detection system would be successful at all times. There can be advanced forms of attacks, which might defeat the IDS itself, wherein the IDS fails to identify the attack and gives out false negative and the system becomes defenseless. But once the attack has occurred, a copy of the malware still exists within the Decepticon architecture. That malware can then be analyzed and attacks on systems with similar vulnerabilities can be thwarted. As shown in Fig. 6.5, if one system is under attack, the information is communicated to the other nodes in the networked environment through the PCU network and preventive action can be taken to save the remaining nodes. The probability matrix can be updated for future use. The detection system can be trained on past attacks, extracting features and updating the probability matrices. When the IDS gives out false positives, then also the performance is not affected as it happens outside the system.

Let us now discuss the benefit of using HMM to counter APTs. The purpose

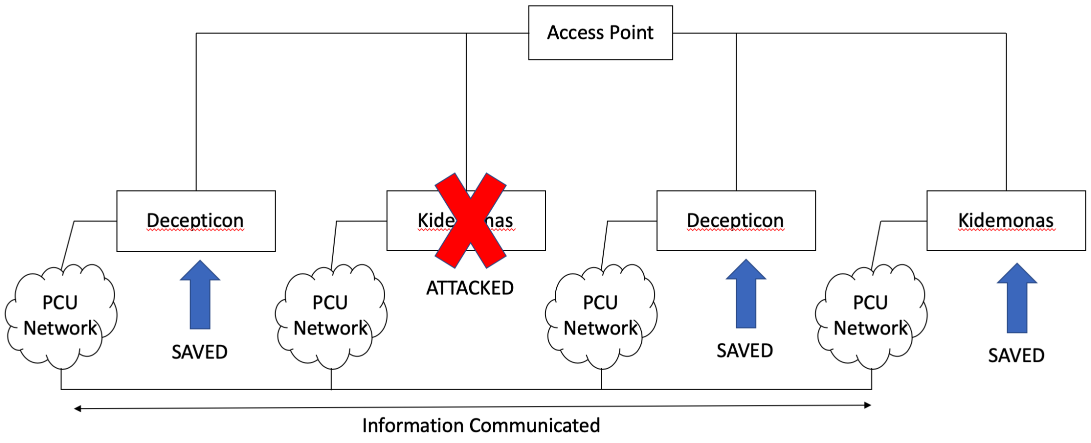


Figure 6.5: A Scenario wherein One System is under Attack

of our framework is to anticipate the state of the malware and take preventive action. For this purpose, one can use a classifier. Using the feature set for a given state, one can do online prediction of the state of the malware. As shown in Fig. 6.6, given the feature set at time $t + 1$ and the behavior observed till time t (the behavior observed through the feature sets manifested for the respective states) and the states observed till time t, the classifier can predict the state of the malware at time $t + 1$. This would be immensely helpful in tailoring a preventive action against the malware. But the HMM based IDS can do more than that. An HMM based IDS would also be able to provide more behavioral data regarding the malware and in case of an APT, the behavioral pattern of the attacker can be logged and analyzed through the probability matrices. It can be trained using similar attacks originating from different APT groups and/or can be trained on different attacks originating from the same APT group. This would not only help the defender to ascertain the state of the malware but also would give an insight regarding the behavior of the malware and/or the attacker.

As illustrated above, our model shows the way the countermeasures become

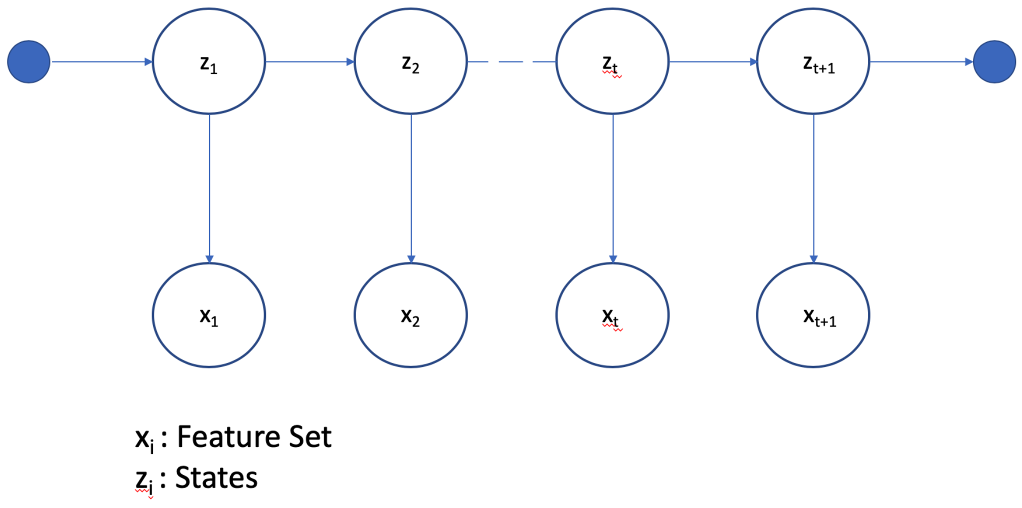


Figure 6.6: Classifier Based On-line Predictive Model

more sophisticated as and when the malware advances to the higher states. The calculation of the transition probability and the emission probability matrices as well as the initial probability distribution is not done in this research due to lack of real world data. The HMM based detection tool and the surreptitious reporting of the intrusion information by the Decepticon architecture pave the way for better security in corporate environments as well as in mission critical systems.

6.3 Summary

In summary:

- We put forward an architecture called Kidemonas as a defense against APT attacks.
- Kidemonas uses commercial-off-the-shelf (COTS) hardware components

to keep the manufacturing cost low.

- This is an open-ended design to accommodate different types of APT detection systems to be put inside Kidemonas.
- The framework paves a path for development of a distributed system to detect generic APT attacks keeping the computational requirements low. More details can be found in the published work [84].
- For our next deception architecture, we concentrated on APT type ransomware.
- We designed an HMM based IDS to detect APT type ransomware as shown in Chapter 5. The IDS would be effective against basic ransomware as well.
- Decepticon is a deception based countermeasure. It is a special case of Kidemonas. The APT detection system inside the framework is an HMM based IDS tailored to detect APT type ransomware.
- This countermeasure is very effective in distributed corporate network wherein APT type ransomware could be detected with lower computational capabilities. More details can be found in the published works [117] and [85].

When faced with an attack, the defender may or may not have the framework to defend themselves against APTs. There can also be a situation wherein the framework is defeated by the APT type malware. Then a few questions arise like “What to do?”, “When to do?”, and “How to do?” To answer these questions, the research work put forward in the next chapter becomes useful.

Chapter 7

Attack Mitigation

"Not having a contingency plan or never performing risk analysis and mitigation activities is like not having an insurance plan for yourself."

–Pooja Agnihotri,

17 Reasons Why Businesses Fail: Unscrew Yourself From Business Failure

The intrusion detection system and the deception architecture were designed as a defense against APT type malware. But the systems which do not have these defense features, and/or the systems in which these defense features are defeated by the APT type malware, are put to great risk. This leads to questions like - "What to do?", "When to do?" and "How to do?" with regard to such sophisticated attacks. To answer these questions, our research presented in this chapter explores solutions using game theoretic analysis. We analyzed the threat scenario of non-APT type ransomware as well as sophisticated APT type ransomware using sequential game models. We introduced two parameters which would help the defender in making an informed decision when under attack. Through a formal analysis of a non-APT type ransomware, our research provides a preliminary treatment of the mitigation strategies to counter advanced threats. We then extended the concept of game theory for more sophisticated APT type ransomware. We designed more elaborate sequential game model for multi-stage advanced ransomware attacks, analyzed the threat scenario and traced the optimal strategies of the attacker for different conditions. We came up with equilibrium conditions to maximize the outcome and minimize the losses for the defender with and/or without the defense features. In the following sections, we elaborate the research for mitigation against ransomware.

7.1 Mitigation Technique for Basic Ransomware Attacks

7.1.1 The Threat

A ransomware can be either a basic variant or a more sophisticated APT variant, as discussed in Chapter 3. An APT type ransomware is generally created by nation state actors. They are highly sophisticated attacks and are mounted through multiple clandestine stages [2]. For such attacks, even though monetary gain is generally the primary goal, they may have other concealed and/or disguised agenda. On the other hand, in a basic ransomware attack, the attacker encrypts the resources under risk and charges a ransom. If the ransom is paid, the attacker releases the encrypted resources, else the victim loses the resources forever. Such attacks generally have only one goal, i.e., to make the resources inaccessible to the victim until the ransom is paid [118]. In this section, we restrict ourselves to the research concerning defense against basic ransomware and consider the APT type ransomware in the next section.

Parameterized attack graphs have been proposed in the literature to model attacks that exploit vulnerabilities. The attack graphs capture attacker's preconditions, system and network vulnerabilities, attacker effects, and the impact of the attack on the network [119]. The attacker precondition component include the attacker's capabilities and the knowledge needed to stage the attacks at an *atomic level*. However, attack graphs were found to be not very useful due to scalability concerns regarding both model specifications and eventual threat analysis [108]. Even with automated tools for attack graph generation [120], such traditional approaches are not feasible in the context of ransomware where the attacker might use social engineering tactics and launch the attack in multiple stages. Game theory can effectively model this type of attacks and capture the interactions between the attacker and the defender. In order to facilitate the development of the game model, we introduce two parameters, that are specific to ransomware type attacks, as described next.

7.1.2 The Game

We now present a game to depict the ransomware attack on a vulnerable and under-prepared system. We assume that the attacker exploited some form of vulnerability, thereby not giving the defender any time for preparedness. Once the attack has occurred, the defender is left with one of two choices. The first choice is to pay ransom and hope the decryption key is released by the attacker, while, the other option involves not paying ransom. The defender can make these choices based on certain conditions. We analyze two conditions which would help the defender make an informed decision on the payment of ransom and decryption of the encrypted resources held for ransom. In accordance

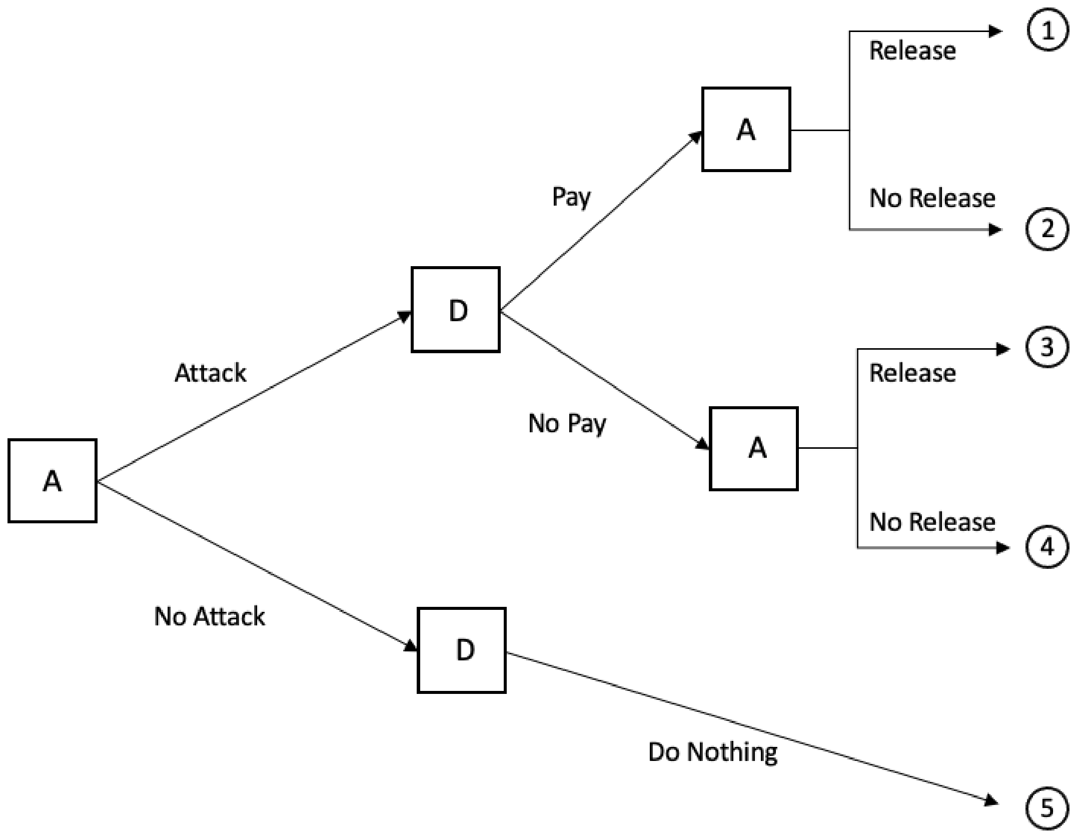


Figure 7.1: Basic Ransomware Attack

with our assumptions, the *willingness* of the defender to pay ransom primarily depends on two factors, the value of recovered resources under siege and the reputation of the attacker.

Figure 7.1 gives a pictorial representation of the game. The game begins with the attacker choosing either of the two strategies “Attack” or “No Attack.” If the attacker chooses the strategy “No Attack”, then the defender has nothing to do in order to respond to the attack. But if the attacker chooses to mount an “Attack”, the vulnerable resources are encrypted and then the defender is left with one of the two choices, “Pay” or “No Pay” of the ransom. Once the defender has made its move, the attacker has two more strategies to choose from, “Release” or "No Release" of the decryption key. If the attacker is a rational player, it will only release the decryption key if a ransom payment is received. If it does not receive the ransom, it will not release the decryption key. But there can be situations where the attacker may choose to do otherwise. That way the attacker chooses not to be rational. The reasons for the attacker not being rational can be many but it is outside the scope of this research. Since the attacker can be rational or irrational, the reputation of the attacker can play an important role in making an informed decision on the part of the defender when under attack.

In eq. (7.1), parameter r_{Rec} gives the ratio of recovered resources after payment of the ransom to the total value of assets of the defender. Variable R is the value of the resources under risk and/or siege. The ransom value charged by the attacker is denoted by β. The value of recovered resources, once the ransom is paid, is given by $R - \beta$. The value of total assets of the defender is denoted by $R_{TotalAssets}$. Therefore, r_{Rec} indicates the importance of the recovered resources for the defender, given the total value of assets, once the ransom is paid and assuming the encrypted resources have been released.

$$r_{Rec} = \frac{R - \beta}{R_{TotalAssets}} \tag{7.1}$$

In eq. (7.2) below, parameter r_{Rep} establishes the reputation of the attacker either as a rational or irrational player in the game. The higher the value of r_{Rep}, the more rational is the attacker and the higher is its trustworthiness. In an incident, if the attacker releases the decryption key on receiving the ransom payment, we assign 1 as the value of reputation for that incident. If the attacker chooses not to release the decryption key when the defender has not made the ransom payment, we assign 1 as the value of reputation for that incident. For

Table 7.1: Attacker Notations

Notation	Description
x_1, x_2	Attacker's first and second strategies, respectively
x^*, $\widehat{x}$	Optimal strategy and best response, respectively
U_A, U_A^*	Expected Utility and Optimal Utility, respectively
Release	Decision to release the encryption keys (value 0 or 1)
(1-Release)	Decision to not release the encryption keys (No Release)

Table 7.2: Defender Notations

Notation	Description
y	Defender's Strategy
y^*, $\widehat{y}$	Optimal strategy and best response, respectively
U_D, U_D^*	Expected Utility and Optimal Utility, respectively
Pay	Decision to pay the ransom (value 0 or 1)
(1-Pay)	Decision to not pay the ransom (No Pay)

other cases we assign 0 as the value of reputation for that particular incident. Then we take a mean of the reputation values of all the last known reported incidents to calculate the overall reputation of the attacker. If the attack is a first time attack, we assign a value of 0.5 to r_{Rep} for the purpose of decision making. The attackers when they act rationally, the r_{Rep} value for them for the next game goes up. If they act irrationally, then they incur penalty and the r_{Rep} value goes down which results in lower willingness to pay the ransom on the part of the defender.

$$r_{Rep} = [Mean\ of\ all\ last\ known\ reported\ incidents] \tag{7.2}$$

Tables 7.1 and 7.2 list the notations used for describing the utility functions and strategies of the attacker and the defender, respectively.

With all the parameters under consideration, we look into the strategies of both the attacker and the defender. Variable x_1 represents the strategy for the attacker which can take up values “Attack” or “No Attack.” We assign the value of the strategy “Attack” as 1 when the attacker decides to attack and 0 otherwise. Similarly for the “No Attack” strategy, the value is 1 when there is no attack, and 0 otherwise. The strategy variable x_2 for the attacker can take up values “Release” or “No Release.” The value of “Release” is 1 when the attacker decides to release the decryption key, and 0 otherwise. Similarly, the value of “No Release” is 1 when the attacker decides against releasing the decryption key, and 0 otherwise. The decision variable for the defender is denoted by y. It takes up either of the two strategies as its value, viz. “Pay” and “No Pay.” After an attack has taken place, if the defender decides to pay the ransom then the value of the strategy “Pay” is 1, and 0 otherwise. Similarly, if the defender decides against payment of the ransom, the value of “No Pay” strategy is 1, and 0 otherwise. Now, with $x_1 = Attack$, $y = Pay$, and $x_2 = Release$, we define the utility functions of the attacker and the defender. Equations (7.3) and (7.4) show the utility functions of the defender and the attacker, respectively.

$$\begin{aligned} U_D \equiv{} & (x_1) * [(y) * \{(x_2) * (R - \beta) \\ & + (1 - x_2) * (-R - \beta)\} \\ & + (1 - y) * \{(x_2) * (R) + (1 - x_2) * (-R)\}] \\ & + (1 - x_1) * (0) \end{aligned} \tag{7.3}$$

$$\begin{aligned} U_A \equiv{} & (x_1) * [(y) * \{(x_2) * (\beta) \\ & + (1 - x_2) * (\beta)\} + (1 - y) * \{(x_2) * (0) \\ & + (1 - x_2) * (0)\}] \\ & + (1 - x_1) * (\lambda) \end{aligned} \tag{7.4}$$

Table 7.3: Recovered Resources and Reputation Value for Defender

		Value of Affected Resources (r_{Rec})	
		High (H)	Low (L)
Reputation (r_{Rep})	High (H)	H, H	H, L
	Low (L)	L, H	L, L

The above equations show all strategies and all scenarios including the ones which generated a pay-off of 0 for the player. The simplified equations are:

$$\begin{aligned} U_D \equiv (x_1) * [(y) * \{(x_2) * (R - \beta) + (1 - x_2) * (-R - \beta)\} \\ +(1 - y) * \{(x_2) * (R) + (1 - x_2) * (-R)\}] \end{aligned} \tag{7.5}$$

$$\begin{aligned} U_A \equiv (x_1) * [(y) * \{(x_2) * (\beta) + (1 - x_2) * (\beta)\}] \\ +(1 - x_1) * (\lambda) \end{aligned} \tag{7.6}$$

The defender makes a decision of paying the ransom based on the value of resource which it might get back on payment of the ransom, and the reputation of the malware. If the values of r_{Rec} and r_{Rep} are "significantly high" then the defender should pay the ransom. If the values of r_{Rec} and r_{Rep} are "significantly low", then the defender should decide not to pay the ransom. Table 7.3 shows the four main scenarios for different values of r_{Rec} and r_{Rep}. The high and low values of r_{Rec} and r_{Rep} are set by a threshold defined by the defender. The threshold for r_{Rec} is t_{Rec}, value of which is decided by the defender based on the total value of assets it owns. If $r_{Rec} \geq t_{Rec}$ it is said to have a "High (H)" value. Otherwise r_{Rec} is said to have a "Low (L)" value. Similarly, If $r_{Rep} \geq t_{Rep}$ it is said to have a "High (H)" value. Otherwise r_{Rep} is said to have a "Low (L)" value. Predetermining the threshold values helps the defender in attack preparedness. This also helps in making economic decisions with contingency plans in place.

Table 7.4: Pay-off table for the ransomware attack game

Outcome	Attacker	Defender
1	β	$R - \beta$
2	β	$-R - \beta$
3	0	R
4	0	$-R$
5	λ	0

With the strategies and utility functions in place, we now describe how the players make strategic decisions and how the game proceeds. Then we present equilibrium solutions depending upon the various conditions. Thereafter we conduct a sensitivity analysis so that the defender can visualize the expected change in the decision from the change in the parameters. The change in the value of parameter r_{Rec} signifies a change in the importance of the value of encrypted resources to the defender. Change in r_{Rec} is caused by a change in any of the three parameters, viz. R, β and $R_{TotalAssets}$. This would help defender in attack preparedness, and as may be seen through the sensitivity analysis, the effects of change in the value of the parameters on the decision making process would be vivid.

7.1.3 Decision Making Conditions

Table 7.4 shows the pay-off for the defender and the attacker for each outcome. When the attacker decides to attack, the maximum pay-off for the attacker is the ransom amount it receives, as represented by $U_A(x_1 = Attack) = \beta$. For this ransomware, the main goal is monetary gain from the ransom received from the victims. If the attacker decides not to attack, then its pay-off is the savings by avoiding the cost of attack as represented by $U_A(x = No\ Attack) = \lambda$.

For $x_1 = Attack$ the following condition must hold,

$$F_1 \equiv U_A(x = Attack) \geq U_A(x = No\ Attack) \equiv \beta \geq \lambda$$

For $x_1 = No\ Attack$ the following condition must hold,

$$F_2 \equiv U_A(x = Attack) < U_A(x = No\ Attack) \equiv \beta < \lambda$$

From the conditions F_1 and F_2 we get,

$$x_1^* = \begin{cases} \text{“Attack”} & \beta \geq \lambda \\ \text{“No Attack”} & \text{Otherwise} \end{cases} \tag{7.7}$$

For $x_2 = Release$ or $x_2 = No\ Release$ the following condition should hold so that it is in the best interest of the attacker,

$$x_2^* = \begin{cases} \text{“Release”} & \text{y=“Pay”} \\ \text{“No Release”} & \text{Otherwise} \end{cases} \tag{7.8}$$

The attacker can make decisions based on the pay-off table. The attacker starts the game by making the first move. The first mover's advantage goes to them. When the attacker attacks, the defender is left with the choice of paying or not paying the ransom. Once the defender has made the decision, the attacker decides to release or not release the decryption key. With this decision, the attacker ends the game. The decision for the defender cannot be made easily in a similar fashion. The pay-off table does not quantify the importance of the value of resources for the defender. Moreover, the pay-off table does not guarantee or

Table 7.5: Best Response of the Attacker and the Defender given the conditions (Equilibrium Strategies)

Conditions		**Strategies**	
Attacker	**Defender**	**Attacker($\widehat{x_1}, \widehat{x_2}$)**	**Defender($\widehat{y}$)**
$\beta < \lambda$	N/A	No Attack, Nothing	Do Nothing
$\beta \geq \lambda$	$r_{Rep} \geq t_{Rep}$ AND $r_{Rec} \geq t_{Rec}$	Attack, Release	Pay
$\beta \geq \lambda$	$r_{Rep} < t_{Rep}$ OR $r_{Rec} < t_{Rec}$	Attack, No Release	No Pay

give insight into the rationality and reputation of the attacker. Consequently, the defender needs to depend on other parameters. Considering this aspect, in this paper we introduced two parameters to help make the defender an informed decision, viz. r_{Rec} and r_{Rep}.

The defender decides the threshold for both the parameters. If the value of the parameter is above the threshold, then it is quantified to have a "High (H)" value, else "Low (L)" value. Once the defender has the values for both parameters, they need to refer to Table 7.3 in order to make the decision. Therefore, the optimal strategy is

$$y^* = \begin{cases} \text{"Pay"} & r_{Rec} \geq t_{Rec} \text{ AND } r_{Rep} \geq t_{Rep} \\ \text{"No Pay"} & r_{Rec} < t_{Rec} \text{ OR } r_{Rep} < t_{Rep} \end{cases} \tag{7.9}$$

7.1.4 Equilibrium Solutions

The game presents the conditions, the strategies, and the pay-offs for each strategy. Considering these factors, Table 7.5 presents the best responses for both the attacker and the defender. Given the conditions, these best responses translate to equilibrium solutions for the game.

Parameter λ denotes the cost of attack on the part of the attacker. When they decide not to mount an attack, i.e., the strategy is "No Attack", the pay-off is

λ. This is the financial saving they make by avoiding the cost of attack. If the system is harder to infiltrate, then the value of λ is higher. For the defender it means, if the system they build is more secure against infiltration, the value of λ increases which discourages the attacker from mounting the attack. This information is important because this encourages to use a stronger encryption system to secure the database and use security best practices to lower the number of vulnerabilities that might exist in the system.

In the event of an attack, the defender is left with either of the two choices, viz. "Pay" and "No Pay." This is where the threshold values for r_{Rec} and r_{Rep} comes handy in the decision making process. The defender needs to decide a value for t_{Rec} based on the value of resources under siege and the total value of resources owned by them. This helps to decide the limit at which the defender is comfortable in paying the ransom. For different values of the resources, ransom, and total assets, as r_{Rec} goes below t_{Rec}, the willingness to pay the ransom decreases. The reason being the value of recovered resources becomes less important to the defender.

The defender shouldn't rely on the r_{Rec} parameter alone. The reputation of the malware is also important. If the value of r_{Rec} is low, i.e., less than t_{Rec}, the resources are less important to the defender and it can decide not to pay the ransom. But if it is high, i.e., $r_{Rec} \geq t_{Rec}$, the next action the defender should take is to check the value of r_{Rep}. If $r_{Rep} < t_{Rep}$, the reputation of the malware is low. This signifies that if the defender pays the ransom, there is a very high possibility that the attacker wouldn't release the decryption key either. But if $r_{Rep} \geq t_{Rep}$, the reputation of the malware is high and it can be trusted with the payment of the ransom. The best strategy for the defender therefore would be to "Pay" the ransom when $r_{Rec} \geq t_{Rec}$ and $r_{Rep} \geq t_{Rep}$ and "No Pay" otherwise.

The attacker, if feels that the pay-off is higher when $x_1 = Attack$ as compared to $x_1 = No\ Attack$, then mounts the attack. For the basic ransomware considered in this paper, with attacker being rational, it is in its best interest to release the decryption key on receiving the ransom payment and not releasing decryption key, otherwise. Therefore, the attacker's best strategy would be $\widehat{x_1},\ \widehat{x_2} = Attack,\ Release$ on receiving the ransom payment and $\widehat{x_1},\ \widehat{x_2} = Attack,\ No\ Release$ if the ransom payment is not made.

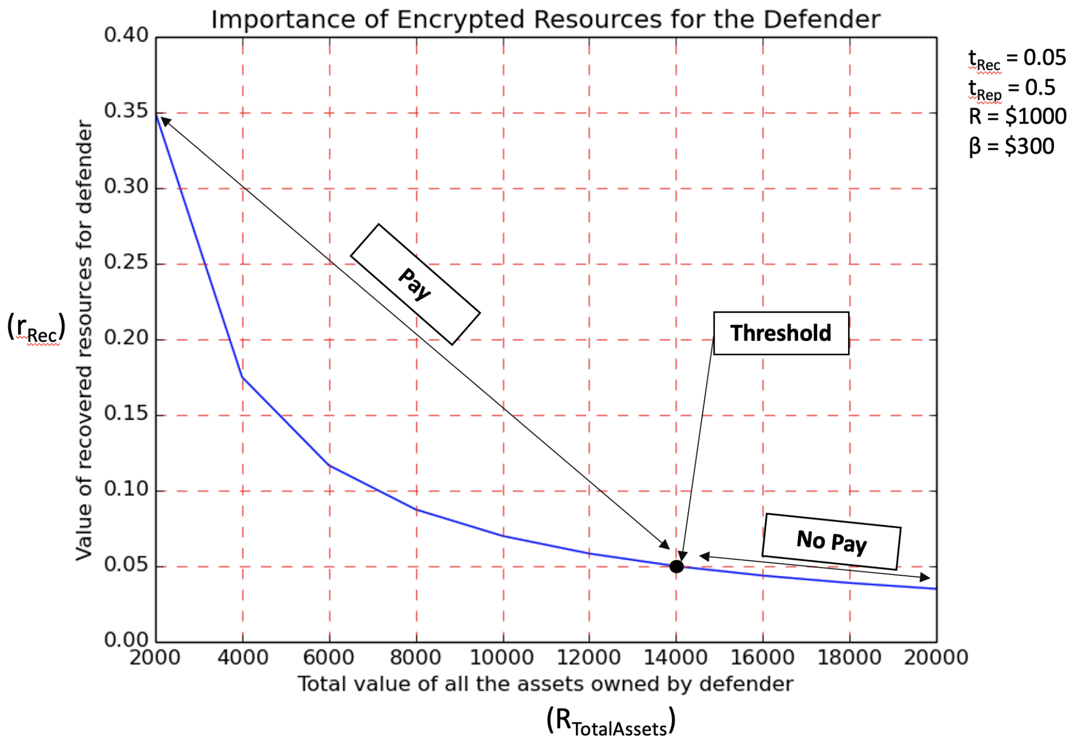
7.1.5 Sensitivity Analysis

We now perform a sensitivity analysis to determine how the values of ransom and total assets affect the decision making of the defender.

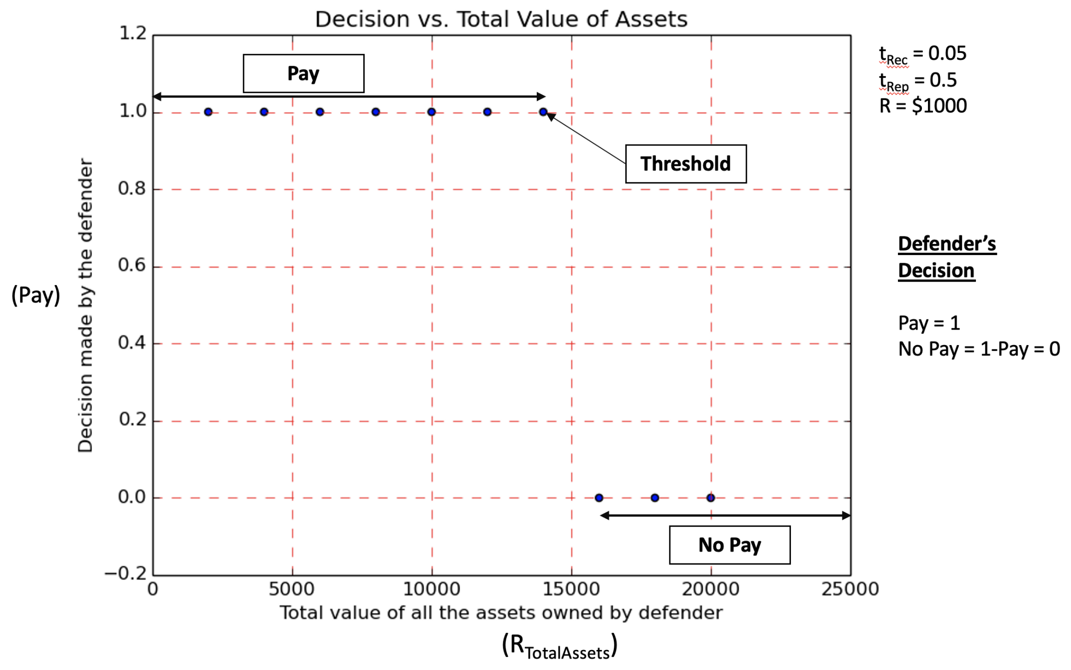
To begin with, we assume an organization with a value of total assets ($R_{TotalAssets}$) of \$10,000. The value of resources (R) under siege is \$1,000. The threshold values for the recovered resources parameter (t_{Rec}) and the reputation parameter (t_{Rep}) are assigned 0.05 and 0.5, respectively. For the purpose of analysis, we use the value of r_{Rep} to be 0.618 (this is obtained by randomly generating a few reputations for past incidents and taking the mean). The ransom value (β) was set at \$300. Now in order to understand how much the total value of all assets affect the decision making process, we vary $R_{TotalAssets}$ while keeping the values of other parameters unchanged. When the value of $R_{TotalAssets}$ is \$14,000, the r_{Rec} value is at the threshold. Figure 7.2a shows how the value of r_{Rec} changes as we increase the value of total assets owned by the defender. The idea is to visualize the change in the importance of the value of resources under siege for the defender when the value of total assets owned by them changes. The higher the value of r_{Rec}, the more important is the encrypted resources to the defender. Figure 7.2b shows how the decision of the defender changes as the importance of the value of encrypted resources changes. The decision “Pay” is denoted by the value 1. The decision “No Pay” or not to pay the ransom is denoted by the value 0, i.e., $No\ Pay = 1 - Pay = 0$. The plot shows that when the importance of the encrypted resources diminishes, the willingness to pay the ransom decreases. From equation (7.1) and Figure 7.2a, it is apparent that r_{Rec} is inversely related to $R_{Total Assets}$. Figure 7.2b shows how increasing the $R_{Total Assets}$ value affects the decision of the defender.

Now, we vary the ransom value from \$100 to \$1,000. We keep the value of R at \$1,000, t_{Rec} 0.05, t_{Rep} 0.5, and r_{Rep} 0.618. Figure 7.3a shows how increasing the ransom value affects the r_{Rec} value. When the ransom value is \$500, it is the threshold value for r_{Rec}. The figure shows that as the value of ransom increases, the effective value of the recovered resources decreases. Therefore, the importance of the same to the defender decreases and so does the willingness to pay the ransom. Figure 7.3b shows how increasing the ransom value affects the decision of the defender. The decision “Pay” is denoted by value 1 for a ransom value and the decision “No Pay” is denoted by 0. With an increase in the value of the ransom, the willingness to pay decreases owing to the fact that the effective value of the recovered resources diminishes. From equation (7.1) and Figure 7.3a, it is evident that the relationship between r_{Rec} and β is linear with a negative slope.

The sensitivity analysis shows how one can visualize the importance of each parameter in the decision making process. In this research, we presented an example with synthetic data. But this is applicable in the real world if the defender wants to plug-in real values. This decision making process helps to make an informed decision when faced with an attack. The sensitivity analysis helps

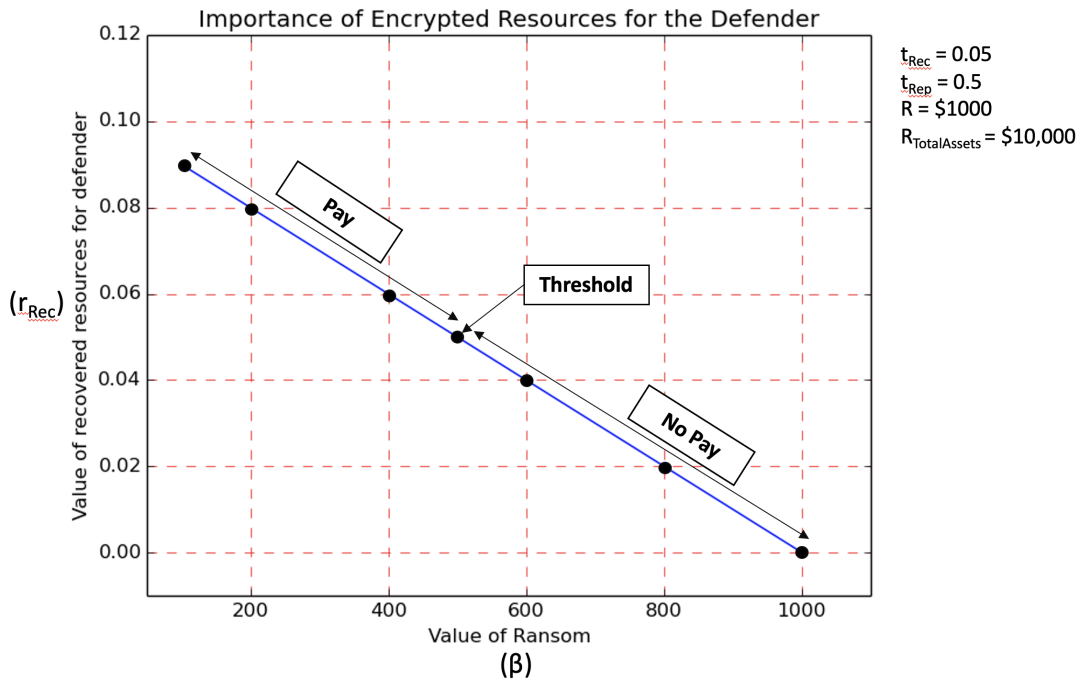


(a) Importance of the resources for the defender

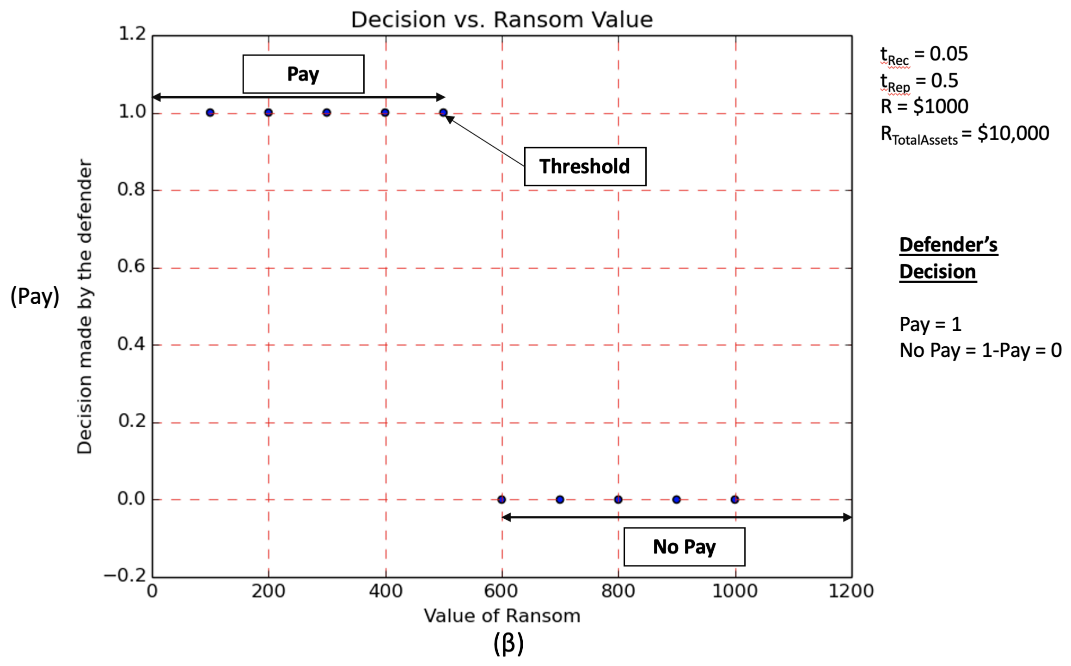


(b) Decision made by the defender

Figure 7.2: Sensitivity Analysis by varying $R_{Total\,Assets}$



(a) Importance of the resources for the defender



(b) Decision made by the defender

Figure 7.3: Sensitivity Analysis by varying Ransom Value

to visualize the effect of the attack and helps in attack preparedness on the part of the defender.

7.1.6 Prescriptive Solution

The sensitivity analysis in Section 7.1.5 shows how the change in the value of ransom or change in the total value of assets owned by the defender affects the decision made by them. The equilibrium strategies for the attacker and the defender are shown in Table 7.5. The game is designed without any data and the equilibrium strategies were obtained through backward induction. Even though the sensitivity analyses were performed on an example set of data, the game would work fine for any range of data for a basic ransomware attack. Data on ransomware attacks are hard to come by as institutions and organizations often do not report the details fearing the leakage of sensitive information in the public domain and/or adverse effects on their reputation. Therefore, there can be an argument here about the incomplete information game [121]. To begin with, one can argue that λ value is an unknown entity to the defender. But through penetration testing and/or employment of ethical hackers that value can be known with quite precision [122], [123]. Another value r_{Rep} may seem to be unknown. But through media reports and browsing through historical attacks by the same malware or the same attackers, it can be calculated. If no information is available whatsoever, then the value of r_{Rep} is assumed to be 0.5. This way the game no longer becomes an incomplete information game. With the values under consideration and assumption, Table 7.5 and Algorithm 1 present a prescriptive solution to a basic ransomware attack. The equilibrium strategies and the algorithm present an opportunity for the defender to prepare in advance and/or make an informed decision when under attack from ransomware.

Algorithm 1 Choosing Defender's strategy based on the Optimized Strategy of Attacker from Table 7.5

```
$t_{Rec}$ = # Set by Defender based on system config.
$t_{Rep}$ = # Set by Defender based on info collected
if $\beta \geq \lambda$ then
    if $r_{Rec} \geq t_{Rec}$ AND $r_{Rep} \geq t_{Rep}$ then
        return Pay
    else
        return No Pay
    end if
else if $\beta < \lambda$ AND $\widehat{x_1} = No\ Attack$ then
    return Do Nothing
end if
```

The basic ransomware may come with few more features. An important feature being an early deadline for ransom payment. After this early deadline, often the value of ransom demanded is doubled. If the defender wishes to pay the ransom, then they will have to pay double the amount after the early deadline. In this scenario, the defender can simply update the values of the parameters wherever applicable, including the value of the ransom. Another feature in the game can be the existence of a bargaining stage between the attacker and the defender. After the bargaining process, if the ransom value changes and/or value of resources under siege changes, then the defender can update the value of the parameters in the game. The value of r_{Rep} is calculated by observing the ransomware attacks which have been known to the defender and/or reported publicly as shown in the eq. (7.2). If the attack is happening for the first time and/or there exists no reports of historical occurrence of the same, then the defender can proceed with the value of 0.5 for r_{Rep}. But through proper investigation, and if any further clue can be found that links the ongoing attack to some other attack and/or attacker, then the defender can update eq. (7.2) using the values from those attacks. Thereafter, the defender can update the values of the parameters and tables in the game.

The defender, with the updated game can refer to Table 7.5 and Algorithm 2 to make an informed decision. Having a strong security system, effective intrusion detection system, strong encryption system and following proper security practices, the defender effectively increases the value of λ. This acts as a deterrent against probable attacks on the system.

7.2 Mitigation Technique for APT type Ransomware Attacks

7.2.1 The Threat

Malware created by the APT groups do not typically carry out the attacks in a single stage. The Cyber Kill Chain framework developed by Lockheed Martin describes an APT through a seven stage life cycle [18]. APT groups are generally nation state actors [124]. They perform highly targeted attacks and do not stop until the goal is achieved [125]. Researchers are always working toward developing a system and process to create an environment safe from APT type attacks [126]. The following characteristics make them a true APT: 1) exploiting zero-day vulnerabilities to achieve their goal, 2) non-stop campaign until goals are achieved, 3) adaptive and having the ability to attack high value targets through multiple modes of attack [2], [117], and 4) using stealth to quietly invade in a series of steps [1]. In this research, the threat considered is ransomware which are developed by APT groups. WannaCry, is an example of a highly sophisticated ransomware created by an APT group called the Lazurus group of North Korea and its level of sophistication is evident from the existence of contingency plan of attack on being discovered [2], [127]. In the following subsections we demonstrate the development of mitigation strategies against APT type ransomware with use of Game Theory.

7.2.2 Notations and Assumptions

An APT type ransomware is equipped with sophisticated capabilities and resources. An APT type ransomware while manifesting the characteristics of a ransomware, it often performs targeted attacks with ulterior motive and interest in the encrypted resources. In these situations, the attacker may not want to release the encrypted files even after receiving the ransom. The attacker might do so when it deems that it has an advantage of keeping the resources to itself. Being an APT, it has huge amount of resources at its disposal. Table 7.6 denotes the possible strategies and notations of the attacker from the following assumptions we make to describe them:

- Ransomware is the primary form of attack, and therefore, the highest payoff is obtained by encrypting the target resources.
- The attacker being an APT type malware, has the capability to perform a contingency form of attack when it perceives that it has been discovered or at risk of being discovered as described in [2]. A contingency attack could mean a different form of attack than the on-going type of attack and that includes both aborting the current attack (campaign abort strategy), and a decoy attack to confuse the defender.

- The attacker is a rational player in game theoretic terms and thus, the strategies opted by it are motivated by the maximization of the payoff.

Table 7.7 denotes the possible strategies and notations of the defender from the following assumptions we make to describe them to address the APT type ransomware attacks.

- The defender is a rational player.
- The defender may have an intrusion detection system (IDS) which employs deception as a defense strategy [117]. The deception architecture in our research means existence of an architecture which deceives the attacker in believing in their success while silently detects the intrusion, surreptitiously reports it to the defender and making an intelligent back-up, all outside the purview of the attacker. The IDS is not perfect. A successful deception and discovery mechanism implies that it was successful in discovering the malware and the defender was surreptitiously reported of the intrusion.
- If the defender is successful in detection and discovery of the ransomware, then it implies that critical data was backed-up through a defender's back-up strategy.

These assumptions will define the premises of the game between the attacker and the defender. Table 7.8 summarizes the pay-off parameters for the defender which are defined below.

- 'g' denotes the damage suffered by the defender when the strategy "No Surrender" is chosen. "No Surrender" strategy means not complying with

Table 7.6: Strategies & Notations of the Attacker in the Game

Strategies & Notations	**Definition**
Ransomware (RW)	Ransomware type attack
Contingency Attack (CT)	Mounts a different form of attack
Campaign Abort (CA)	Aborts the ongoing attack
Campaign Complete (CC)	Completes the ongoing attack
Release (R)	Releases the encrypted resources
No Release (NR)	Does not release the encrypted resources
Attack (AT)	Mounts an attack on the targeted system
No Attack (NAT)	Abstains from mounting an attack (1-AT)
$x^*, \widehat{x}$	Optimal strategy and best response respectively
U_A, U_A^*	Expected Utility and Optimal Utility respectively

Table 7.7: Strategies & Notations of the Defender in the game

Strategies & Notations	**Definition**
Pay (P)	Pays the ransom
No Pay (NP)	Doesn't pay the ransom (1-P)
Surrender (S)	Surrenders to the contingency attack
No Surrender (NS)	Doesn't surrender to contingency attack (1-S)
Do Nothing (NOT)	When there is no attack, the defender does nothing
$y^*, \widehat{y}$	Optimal strategy and best response respectively
U_D, U_D^*	Expected Utility and Optimal Utility respectively

or not giving up when faced with a "Contingency Attack" and taking some or the other defense action.

- 'g_A' denotes the damage suffered by the defender when the attacker chooses the strategy "Campaign Abort" given that the defender has already chosen the strategy "No Surrender."
- 'g_C' denotes the damage suffered by the defender when the attacker chooses the strategy "Campaign Complete" given that the defender has already chosen the strategy "No Surrender."

- 'c' denotes the damage suffered by the defender when the strategy "Surrender" is chosen. "Surrender" strategy means complying with or giving up when faced with a "Contingency Attack."

  - 'c_A' denotes the damage suffered by the defender when the attacker chooses the strategy "Campaign Abort" given that the defender has already chosen the strategy "Surrender."
  - 'c_C' denotes the damage suffered by the defender when the attacker chooses the strategy "Campaign Complete" given that the defender has already chosen the strategy "Surrender."

- 'R' denotes the value of resources that are held hostage for a ransom by the attacker.
- 'd' denotes the cost incurred by the defender to implement a deception, detection, and a back-up mechanism in place.
- P_D denotes the probability that if a deception, discovery, and back-up mechanism is present then whether it is successful or not.

Table 7.9 summarizes the pay-off parameters for the attacker which are defined below.

- 'β' denotes the value of the ransom charged by the attacker when a successful ransomware attack has been mounted.
- 'γ' denotes the value of the encrypted resources for the attacker. Depending on the value of 'γ', the attacker may choose to not release the decryption key even after receiving the ransom payment.
- 'λ' denotes the cost incurred by the attacker to mount a successful attack. There can be a '$-\lambda$' term for every outcome. Then for the "No Attack"

Table 7.8: Model Parameters - Defender

Parameters	**Definition**
g	Damage suffered due to "No Surrender" strategy
g_A	Damage due to "Campaign Abort" strategy
g_C	Damage suffered due to "Campaign Complete" strategy
c	Damage suffered due to "Surrender" strategy
c_A	Damage due to "Campaign Abort" strategy
c_C	Damage suffered due to "Campaign Complete" strategy
R	Value of the resources under risk
d	Cost of implementing deception and back-up mechanism
P_D	Probability of discovery and having deception mechanism

scenario the outcome for the attacker is 0. The equations, the math and the equilibrium solution would still be the same. For a costly attack, the outcome for the attacker could be negative. We just took out the '$-\lambda$' term from every outcome and kept the "No Attack" outcome to be 'λ', to show the monetary savings.

- 'δ' denotes the value of gains for the attacker when the "Campaign Abort" strategy is opted.
- 'ω' denotes the value of the gains for the attacker when the "Campaign Complete" strategy is opted.
- P_P denotes the probability with which the attacker perceives whether a sophisticated deception and detection mechanism to detect ransomware attack exists in the system or not.

7.2.3 Game Model: Model Description

The game begins when the attacker creates a malware and mounts an attack on the system. If there is no attack, the defender does nothing and no attack-

Table 7.9: Model Parameters - Attacker

Parameter	**Definition**
β	Ransom charged for the encrypted resources
γ	Advantage parameter for *"Not Releasing"* the key
λ	Cost of *attack*
δ	Gain from "Campaign Abort" strategy
ω	Gain from "Campaign Complete" strategy
P_P	Perception probability regarding deception mechanism

defense situation arises. The attack begins with initial reconnaissance of the system, followed by initial compromise. If the attacker perceives that it has not been discovered and/or chances of being discovered is low, then the ransomware type attack is executed. If the attacker perceives that it has been discovered and/or there is a deception and discovery mechanism in the system which might lead to its discovery, then the contingency plan of attack is executed. We introduce a *chance node* in the game tree instead of a *decision node* to represent this situation. Chance nodes in a game model are the ones wherein a player makes one of the possible decisions with certain probability. Decision nodes in a game model are the ones wherein a player makes one of the possible decisions with certainty. By introducing chance nodes to represent the attacker's perception, we can assume that with certain probability the malware perceives its discovery and executes the contingency plan of attack. In response to the contingency attack, the defender can choose to either surrender or not surrender. The attacker then can proceed to either the campaign complete or the campaign abort option, owing to its favorable outcome.

If there is no deception and discovery mechanism, or the mechanism failed to detect the ransomware attack, the attacker would have correctly perceived a low probability of discovery. Under such condition, the attacker mounts ran-

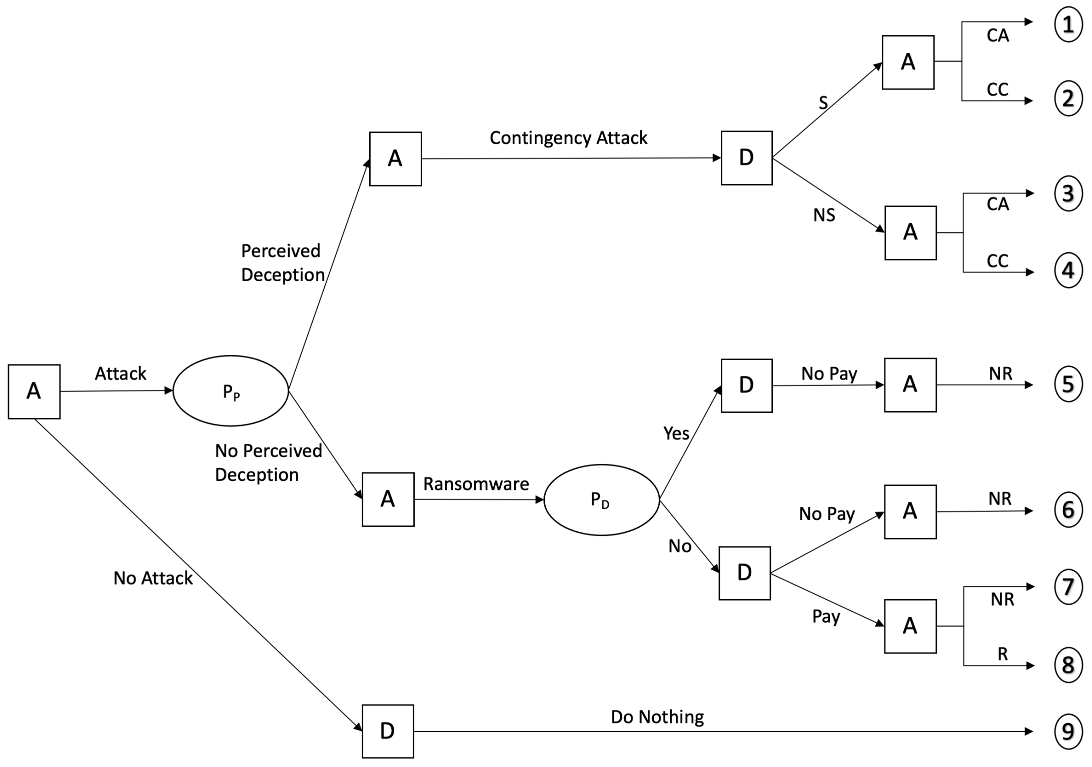


Figure 7.4: APT Type Ransomware Game

somware attack and the defender may decide to pay. Once the attacker has received the ransom, it may choose to release or not release the decryption key, depending upon the favorable outcomes defined by the pay-offs. If the defender decides not to pay the ransom assuming that ransom value is too high, then the attacker chooses not to release the decryption key. But if the attacker's perception of lack of a defense mechanism is false and it has been silently discovered by the IDS and surreptitiously been reported to the defender chooses to not pay the ransom. The attacker being a rational player in the game, then chooses to not release the decryption key.

Table 7.10: Pay-off for each outcome in the Game

Scenario	Outcome	Attacker	Defender
Contingency Attack	1	δ	$-c_A$
	2	ω	$-c_C$
	3	δ	$-g_A$
	4	ω	$-g_C$
Ransomware Attack	5	0	$R-d$
	6	0	$-R$
	7	$\beta+\gamma$	$-\beta-R$
	8	β	$-\beta+R$
No Attack	9	λ	0

7.2.4 Pay-off for the Players

The attacker makes the first move and the defender responds. Then depending upon the defender's response, the attacker makes the next move. Therefore, we have a sequential game as shown in Fig. 7.4. The strategies opted by the attacker are shown in Table 7.6. The strategies undertaken by the defender are listed in Table 7.7. The model parameters for the defender and the attacker are listed in tables 7.8 and 7.9, respectively. The model parameters help to construct the pay-off table for each outcome in the game as shown in Table 7.10. When the attacker perceives that it has been discovered or there is a risk of discovery by the defender, it proceeds with the contingency attack with probability P_P. With probability $1-P_P$, the attacker proceeds with the ransomware attack, as the perception is that the defender has failed to discover. When a ransomware type attack is ongoing, the probability of the defender discovering the attacker without the latter's knowledge is given by P_D. The probability of the attacker being not discovered by the defender during a ransomware type attack is given by $1-P_D$.

7.2.5 Decision Making Conditions

7.2.5.1 Best Response of the Attacker

When the attacker decides to attack, they can either choose to mount a ransomware type attack or a contingency mode of attack depending upon the perception. During the contingency mode of attack, the attacker can respond to the defender by choosing either of the two strategies, Campaign Abort (CA) or Campaign Complete (CC). Expected utility of the attacker in case they choose the strategy CA is given by eq. (7.10) and the expected utility when they choose CC is given by eq. (7.11).

$$U_A(x = CA) = \delta \tag{7.10}$$

$$U_A(x = CC) = \omega \tag{7.11}$$

For $\widehat{x_{CT}} = CA$ the condition is given by eq. (7.12). Similarly, for $\widehat{x_{CT}} = CC$ the condition is given by eq. (7.13).

$$F_1 \equiv U_A(x = CA) \geq U_A(x = CC) \equiv \delta \geq \omega \tag{7.12}$$

$$F_2 \equiv U_A(x = CA) < U_A(x = CC) \equiv \delta < \omega \tag{7.13}$$

When mounting the contingency attack, the best response of the attacker can be derived from the equations (7.10) - (7.13). The best response is given by eq. (7.14).

$$\widehat{x_{CT}} = \begin{cases} \text{CA} & \text{if } \delta \geq \omega \\ \text{CC} & \text{Otherwise} \end{cases} \tag{7.14}$$

If the defender has a deception architecture in place against a ransomware attack, then they decide not to pay the ransom, and thereafter the attacker decides not to release the decryption key. If the defender has no deception architecture during the ransomware attack, and they choose to pay the ransom, the attacker can respond by either releasing (R) the decryption key or not releasing (NR) the decryption key. The expected utilities of the attacker for release of the decryption key is given by eq. (7.15) and similarly for the no release of the decryption key is given by eq. (7.16).

$$U_A(x = R) = \beta \tag{7.15}$$

$$U_A(x = NR) = \beta + \gamma \tag{7.16}$$

For $\widehat{x_{RW}} = R$ the condition is given by eq. (7.17). Similarly, for $\widehat{x_{RW}} = NR$ the condition is given by eq. (7.18).

$$F_3 \equiv U_A(x = R) \geq U_A(x = NR) \equiv \gamma \leq 0 \tag{7.17}$$

$$F_4 \equiv U_A(x = R) < U_A(x = NR) \equiv \gamma > 0 \tag{7.18}$$

When mounting the ransomware attack, the best response of the attacker can be derived from the equations (7.15) - (7.18). The best response is given by eq. (7.19).

$$\widehat{x_{RW}} = \begin{cases} \text{R} & \text{if } \gamma \leq 0 \\ \text{NR} & \text{Otherwise} \end{cases} \tag{7.19}$$

7.2.5.2 Best response of the Defender

The defender is the second mover in the game. The defender responds to either the contingency attack or the ransomware attack. During the contingency attack, the defender can respond to the attacker with either surrendering (S) or not surrendering (NS) to the attack. Expected utilities of the defender in case they choose to surrender when the attacker chooses to abort the campaign or complete the campaign are given by the equations (7.20) and (7.21), respectively.

$$U_D(y = S,\ x = CA) = -c_A \tag{7.20}$$

$$U_D(y = S,\ x = CC) = -c_C \tag{7.21}$$

Similarly, expected utilities of the defender in case they choose not to surrender when the attacker chooses to abort the campaign or complete the campaign are given by the equations (7.22) and (7.23), respectively.

$$U_D(y = NS,\ x = CA) = -g_A \tag{7.22}$$

$$U_D(y = NS,\ x = CC) = -g_C \tag{7.23}$$

When condition F_1 holds given by eq. (7.12), the condition given by eq. (7.24) must hold for $\widehat{y_{CT}} = S$

$$\begin{aligned} F_5 &\equiv U_D(y = S,\ x = CA) > U_D(y = NS,\ x = CA) \\ &\equiv c_A < g_A \end{aligned} \tag{7.24}$$

When condition F_2 holds given by eq. (7.13), the condition given by eq. (7.25) must hold for $\widehat{y_{CT}} = S$

$$\begin{aligned} F_6 &\equiv U_D(y = S,\ x = CA) > U_D(y = NS,\ x = CA) \\ &\equiv c_C < g_C \end{aligned} \tag{7.25}$$

When condition F_1 holds given by eq. (7.12), the condition given by eq. (7.26) must hold for $\widehat{y_{CT}} = NS$

$$\begin{aligned} F_7 &\equiv U_D(y = S,\ x = CA) \leq U_D(y = NS,\ x = CA) \\ &\equiv c_A \geq g_A \end{aligned} \tag{7.26}$$

When condition F_2 holds given by eq. (7.13), the condition given by eq. (7.27) must hold for $\widehat{y_{CT}} = NS$

$$\begin{aligned} F_8 &\equiv U_D(y = S,\ x = CA) \leq U_D(y = NS,\ x = CA) \\ &\equiv c_C \geq g_C \end{aligned} \tag{7.27}$$

From the equations (7.22) - (7.27), the best response function of the defender can be defined as shown in equation (7.28) when the attacker decides to launch a contingency attack.

$$\widehat{y_{CT}} = \begin{cases} \text{S} & \text{if } [\{(\delta \geq \omega) \cap (F_5)\} \cup \{(\delta < \omega) \cap (F_6)\}] \\ \text{NS} & \text{Otherwise} \end{cases} \tag{7.28}$$

During a ransomware attack, the defender with a deception architecture can respond to the attacker by not paying the ransom and the attacker responds to that by not releasing the decryption key. This infers that the defender is not in need of the decryption key and primarily being a ransomware, the attacker decides not to release the decryption key. In the situation when the defender doesn't have a deception architecture, they can respond by either of the two strategies, Pay or No Pay. The expected utilities of the defender are given by the equations (7.29) - (7.31) when the attacker decides to mount a ransomware attack.

$$U_D(y = NP,\ x = NR) = -R \tag{7.29}$$

$$U_D(y = P,\ x = R) = -\beta + R \tag{7.30}$$

$$U_D(y = P,\ x = NR) = -\beta - R \tag{7.31}$$

When the condition F_3 holds as shown by the eq. (7.17), the best response by the defender $\widehat{y_{RW}} = NP$ is given by eq. (7.32). Similarly, the best response by the defender $\widehat{y_{RW}} = P$ is given by eq. (7.33).

$$\begin{aligned} F_9 \equiv U_D(y = NP,\ x = NR) > U_D(y = Pay,\ x = R) \\ \equiv \beta > 2R \end{aligned} \tag{7.32}$$

$$\begin{aligned} F_{10} \equiv U_D(y = NP,\ x = NR) \leq U_D(y = Pay,\ x = R) \\ \equiv \beta \leq 2R \end{aligned} \tag{7.33}$$

When the condition F_4 holds as shown by eq. (7.18), the decision not to pay strictly dominates the decision to pay. Hence, $\widehat{y_{RW}} = NP$.

From the equations (7.29) - (7.33), we can write the best response function of the defender for a ransomware type attack as shown in eq. (7.34).

$$\widehat{y_{RW}} = \begin{cases} \text{P} & \text{If } [(\gamma \leq 0) \cap (\beta \leq 2R)] \\ \text{NP} & \text{Otherwise} \end{cases} \tag{7.34}$$

When the attacker chooses not to attack, the only option for the defender is to do nothing. Hence, $\widehat{y} = NOT$.

7.2.6 Equilibrium Solutions

The best responses of the attacker and the defender under different conditions were derived in Section 7.2.5. From those conditions and respective best responses of the players, we obtain the equilibrium solutions as shown in Table 7.11. The equilibrium solutions in Table 7.11 were obtained through backward induction from the game as shown in Figure 7.4 and the pay-offs for each player as shown in Table 7.10. One key result which is directly evident from the equilibrium solutions is that if the value of λ can be made considerably high, i.e., if the system is secure enough so that the cost of attack is really high, then it acts as a deterrent against an attack. This can be achieved through strong encryption, proper authorization and authentication protocols, and by creating an awareness of the security of the system amongst the user through proper training programs.

7.2.7 Sensitivity Analysis

As noted before, data on a ransomware attack is hard to acquire. Organizations often refrain from reporting owing to effects on their reputation and/or

Table 7.11: Equilibrium Solution

Case	Condition	x^*	y^*	U_A^*	U_D^*
1	$[\delta \geq max(\omega, \frac{\lambda}{P_P})] \cap [\{\gamma > 0\} \cup \{(\gamma \leq 0) \cap (\beta > 2R)\}] \cap [c_A < g_A]$	CA, NR	S, NP	δP_P	$(1 - P_P)(2RP_D - dP_D - R) - c_A P_P$
2	$[\delta \geq max(\omega, \frac{\lambda}{P_P})] \cap [\{\gamma > 0\} \cup \{(\gamma \leq 0) \cap (\beta > 2R)\}] \cap [c_A \geq g_A]$	CA, NR	NS, NP	δP_P	$(1 - P_P)(2RP_D - dP_D - R) - g_A P_P$
3	$[\delta \geq max(\omega, \frac{\lambda - \beta(1-P_P)(1-P_D)}{P_P})] \cap [\{(\gamma \leq 0) \cap (\beta \leq 2R)\}] \cap [c_A < g_A]$	CA, R	S, P	$\delta P_P + \beta(1 - P_P)(1 - P_D)$	$(1 - P_P)(-\beta + R - dP_D + \beta P_D) - c_A P_P$
4	$[\delta \geq max(\omega, \frac{\lambda - \beta(1-P_P)(1-P_D)}{P_P})] \cap [\{(\gamma \leq 0) \cap (\beta \leq 2R)\}] \cap [c_A \geq g_A]$	CA, R	NS, P	$\delta P_P + \beta(1 - P_P)(1 - P_D)$	$(1 - P_P)(-\beta + R - dP_D + \beta P_D) - g_A P_P$
5	$[\omega \geq max(\delta, \frac{\lambda}{P_P})] \cap [\{\gamma > 0\} \cup \{(\gamma \leq 0) \cap (\beta > 2R)\}] \cap [c_C < g_C]$	CC, NR	S, NP	ωP_P	$(1 - P_P)(2RP_D - dP_D - R) - c_C P_P$
6	$[\omega \geq max(\delta, \frac{\lambda}{P_P})] \cap [\{\gamma > 0\} \cup \{(\gamma \leq 0) \cap (\beta > 2R)\}] \cap [c_C \geq g_C]$	CC, NR	NS, NP	ωP_P	$(1 - P_P)(2RP_D - dP_D - R) - g_C P_P$
7	$[\omega \geq max(\delta, \frac{\lambda - \beta(1-P_P)(1-P_D)}{P_P})] \cap [\{(\gamma \leq 0) \cap (\beta \leq 2R)\}] \cap [c_C < g_C]$	CC, R	S, P	$\omega P_P + \beta(1 - P_P)(1 - P_D)$	$(1 - P_P)(-\beta + R - dP_D + \beta P_D) - c_C P_P$
8	$[\omega \geq max(\delta, \frac{\lambda - \beta(1-P_P)(1-P_D)}{P_P})] \cap [\{(\gamma \leq 0) \cap (\beta \leq 2R)\}] \cap [c_C \geq g_C]$	CC, R	NS, P	$\omega P_P + \beta(1 - P_P)(1 - P_D)$	$(1 - P_P)(-\beta + R - dP_D + \beta P_D) - g_C P_P$
9	$[\lambda > max\{\delta P_P,\ \delta P_P + \beta(1 - P_P)(1 - P_D),\ \omega P_P,\ \omega P_P + \beta(1 - P_P)(1 - P_D)\}]$	NAT	NOT	λ	0

leakage of information regarding the intellectual properties of the organization. A thorough validation of our model is thus not possible due to the paucity of real world data. Alternatively, we perform a sensitivity analysis to show the working of our model and to gain some insight of the attack-defense scenario of ransomware. For the purpose of the sensitivity analysis we use synthetic data with a small example organization or a start-up as a use case. The value of the resources under siege by an APT type ransomware is assumed to be \$10,000. The values of the different parameters were varied to observe the equilibrium solution pertaining to the changing conditions. The low and high values of the parameters for different conditions for the sensitivity analyses are shown in Table 7.12. Ransomware is the primary form of attack, so it is expected that ransom should have a high value. Therefore, in the sensitivity analysis for the perception probability (P_P) and discovery probability (P_D), the value of ransom was kept at \$10,000 whenever $\beta \leq 2R$ and was kept at \$21,000 whenever $\beta > 2R$. Contingency attack is a fall back option for the attacker, therefore we kept the outcome for δ at \$6,000 and ω at \$3,000 whenever the condition was $\delta \geq \omega$ and the values were reversed when the condition was otherwise. This makes the intended outcome for the attacker lesser in case of a contingency attack. The cost of a deception mechanism was kept at 10% and therefore the value of d was set at \$1,000. Similarly, the losses for the defender if the attack is purely a contingency attack should be lesser. Therefore, the value of c_A was kept at \$5,000 and the value of g_A at \$2,500 whenever $c_A \geq g_A$ and the values were reversed when the condition was otherwise. The values for c_C and g_C were chosen in a similar manner. For the purpose of the sensitivity analysis, we chose to keep the value for λ at \$1,000 which is again 10% of the value of resources under siege. The value of γ was kept at 0. Even though we chose certain values for the parameters, the game model was not designed for any range of values. The model should work fine for any range and set of data. The equilibrium conditions triggered are sensitive to the conditions satisfied and would work for any value.

The strategies of the defender are mostly reactions to the strategies of the attacker. Moreover, the attacker is the one who begins and ends the game. Consequently, the strategies of the attacker play a dominant role in deciding the equilibrium condition case being triggered pertaining to a particular set of conditions. The attacker being an APT type ransomware, we are more interested in the parameters pertaining to the ransomware type attack. Contingency attack can be any form of attack which the attacker is capable of, when the perception is that a ransomware type attack is not much effective. So, herein we report the effect of changing the values of the ransom, the perception probability and the discovery probability from the sensitivity analysis so as to understand the attack-defense scenario of an APT type ransomware.

Table 7.12: Values of parameters for sensitivity analyses

Parameter	Value(s)
δ	\$3,000, \$6,000
ω	\$3,000, \$6,000
c_A	\$2,500, \$5,000
c_C	\$2,500, \$5,000
g_A	\$2,500, \$5,000
g_C	\$2,500, \$5,000
λ	\$1,000
d	\$1,000
γ	\$0

7.2.7.1 Varying the Ransom Value (β)

We varied the ransom value demanded by the attacker from \$1,000 to \$41,000. The values of P_P and P_D were kept at 0.5 and 0.25, respectively. Figure 7.5 shows the equilibrium condition cases being triggered under different conditions. The number on y-axis refers to the equilibrium condition case number as shown in Table 7.11. Figure 7.6 shows the outcome of the players for the conditions $\delta > \omega$, $c_A < g_A$ and equilibrium condition cases 1 and 3.

7.2.7.2 Varying the Perception Probability (P_P)

The perception probability plays an important role in the attacker's decision making regarding ransomware attack or contingency attack. But through sensitivity analysis we find that if the conditions remain the same, the probability of perception of the attacker of a probable discovery by the defender doesn't have much effect on the change of equilibrium condition when the value of P_P changes. The conditions wherein the attacker's outcome is less than the cost of attack, the attacker decides not to attack and in other cases it is motivated by the maximization of their outcome. Therefore, changing conditions plays a

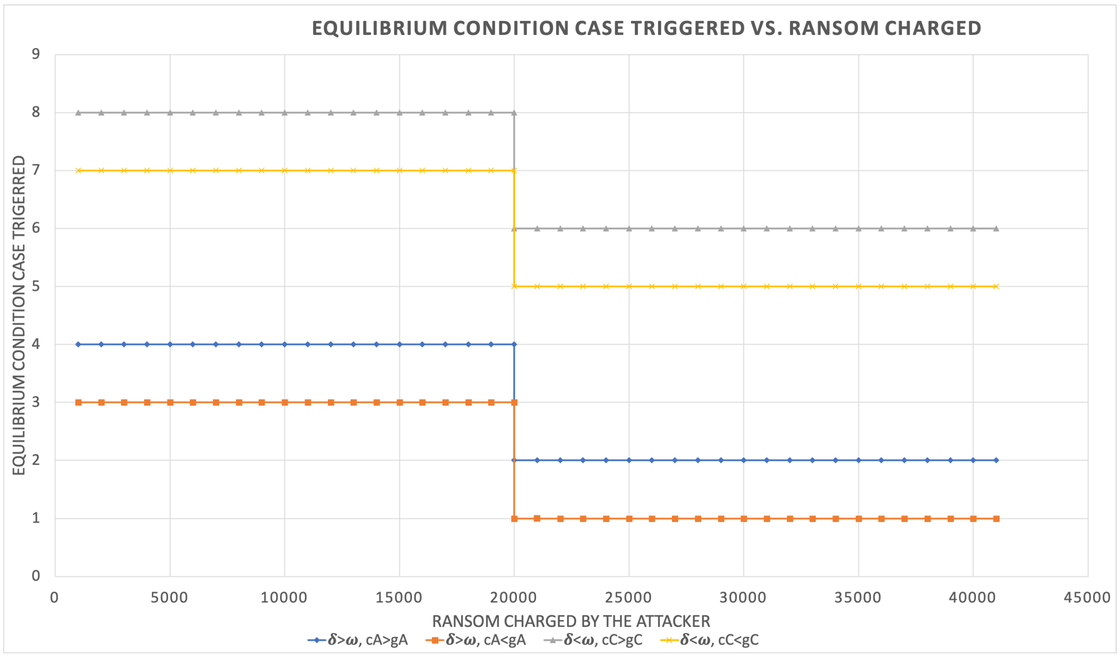


Figure 7.5: Equilibrium Condition Case Triggered vs. Ransom Value for different Conditions

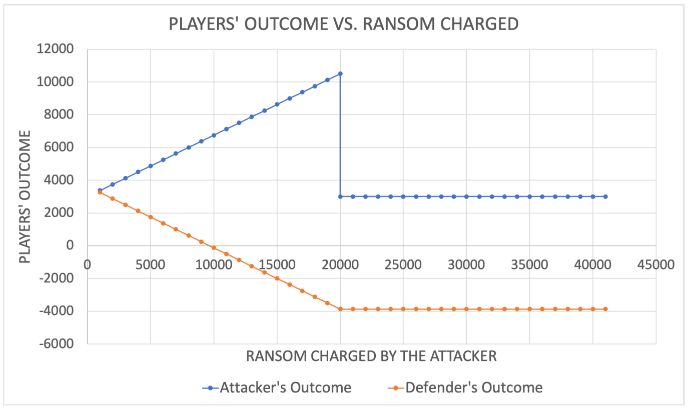


Figure 7.6: Players' Outcome vs. Ransom Value for $\delta > \omega$, $c_A < g_A$ (Equilibrium Condition Cases 1 and 3)

more important role than changing the perception probability of the attacker. Figure 7.7 shows the equilibrium condition case triggered when the perception probability of the attacker on their discovery changes. The number on y-axis refers to the equilibrium condition case number as shown in Table 7.11. Figure 7.8 shows the players' outcome for varying perception probability of the

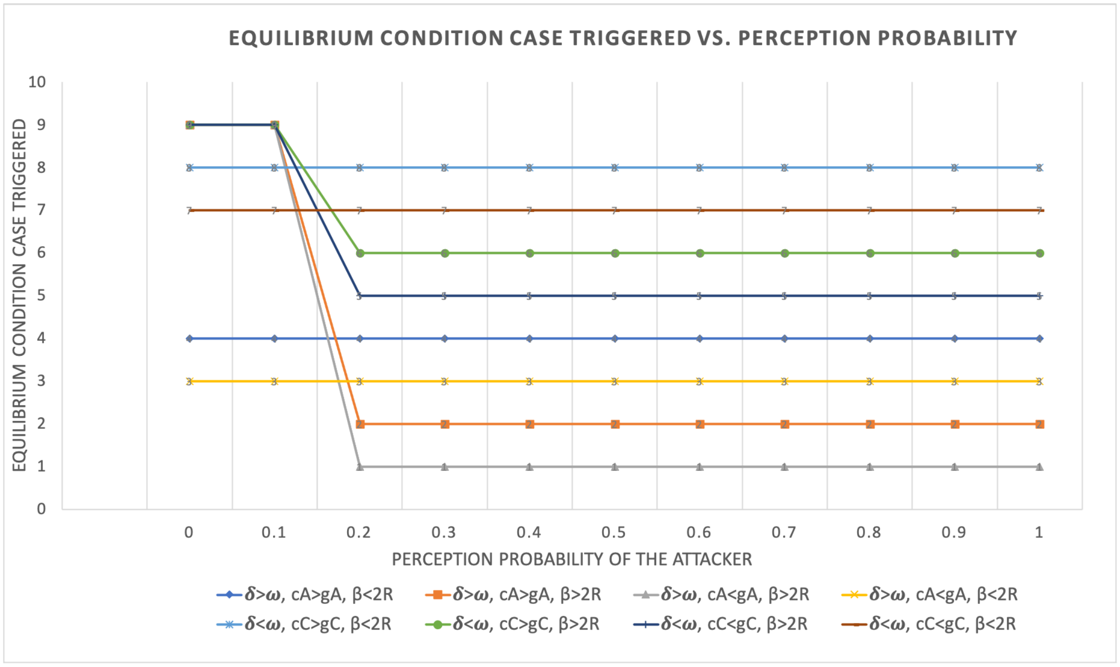
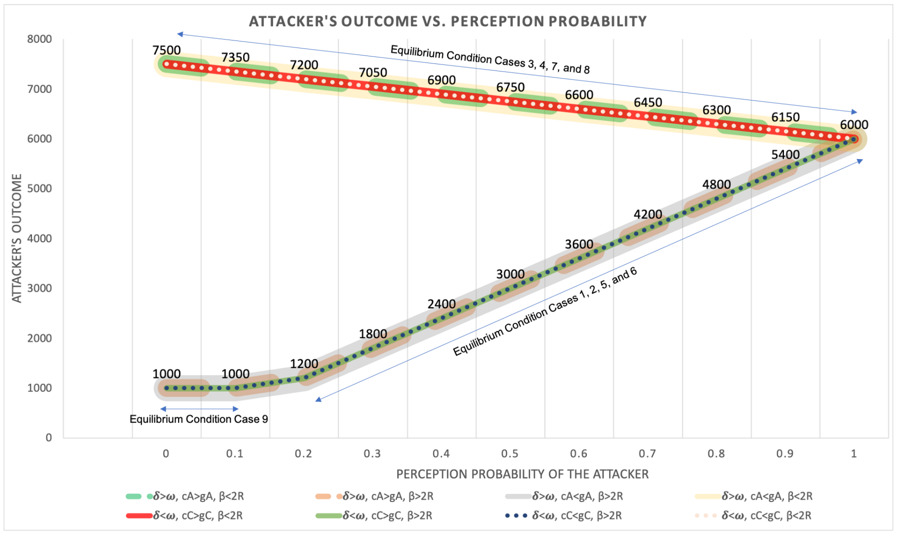
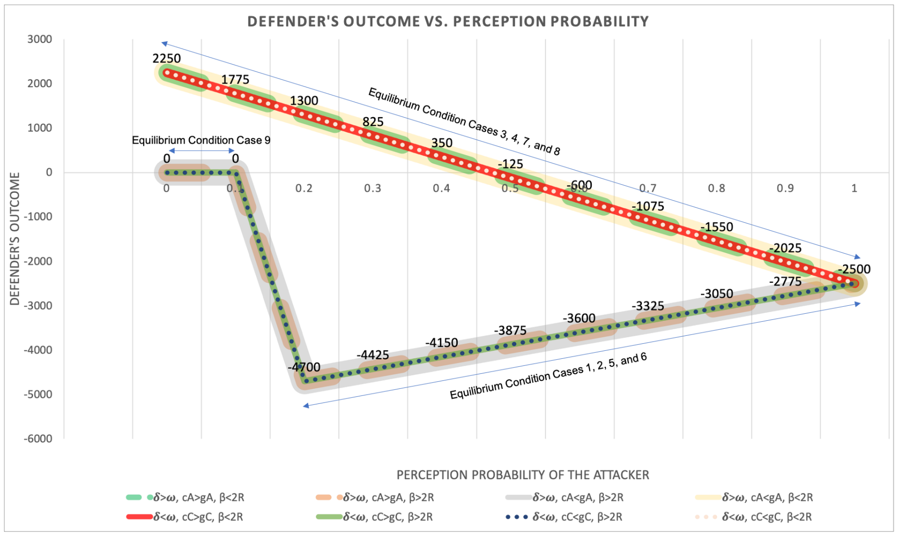


Figure 7.7: Equilibrium Condition Case Triggered vs. Perception Probability for different conditions

attacker. Figure 7.8a shows that the attacker's outcome varies between \$6,000 and \$7,500 for equilibrium condition cases 3, 4, 7, and 8. It also shows that the outcome varies between \$1,000 and \$6,000 for equilibrium condition cases 1, 2, 5, 6, and 9. Figure 7.8b shows that the defender's outcome varies between -\$2,500 and \$2,250 for equilibrium condition cases 3, 4, 7, and 8. It also shows that the outcome varies between \$0 and -\$4,700 for equilibrium condition cases 1, 2, 5, 6, and 9. The negative outcome implies net loss for the defender in the game. The players' outcomes do change with the variation in the perception probability but the equilibrium condition case changes with an overall change in the conditions as shown in Table 7.11.



(a) Attacker's Outcome



(b) Defender's Outcome

Figure 7.8: Players' Outcome vs. Perception Probability for different conditions

7.2.7.3 Varying the Discovery Probability (P_D)

We observed similar results for the discovery probability (P_D) variation-based sensitivity analysis compared to the perception probability (P_P) variation-based sensitivity analysis of Sec. 7.2.7.2. For reasons of brevity, the graphs are not included in the paper.

7.2.8 Prescriptive Solutions

The sensitivity analysis shows how the decisions would be affected by the various parameters. The solutions were not data dependent and would work for any sets and any range of data given the conditions are satisfied. Therefore, this would give the defender a definite set of strategies when a particular set of conditions are met, which we demonstrated through a set of values for the parameters with reasonable estimations. The equilibrium solutions are not very sensitive to the changing value of the parameters, and thereby helping the defender to make conclusive decisions given the conditions. As in the case of basic ransomware analysis, there can be an the incomplete information game [121]. The information regarding certain parameters are generally unknown, which includes P_P, P_D, δ, ω, c_A, c_C, g_A, g_C and γ. The game is designed as such that the equilibrium solutions are not much sensitive to the change in the values of P_P and P_D. The values of the outcomes for the attacker and the defender would change owing to the changes in the values of P_P and P_D, but the equilibrium strategies for a given set of conditions would mostly remain the same. Thus, the defender can prepare for every condition when the values of the rest of the parameters are known. The values of δ and ω denote the gains from the Campaign Abort and Campaign Complete strategies on the part of the attacker if they decide to choose the contingency plan of attack. If the defender employs ethical hackers and/or perform penetration testing [122], [123] regularly on their system, they would be known values. Moreover, they will become aware of the vulnerabilities in the system and the effects of Campaign Abort and Campaign Complete strategies of the attacker and thereby gain the knowledge of the other parameters in the system like c_A, c_C, g_A, and g_C. For the value of γ, the defender can assume that it is of the same value to the attacker as it is to them given the nature of the resources. All these would help the defender to prepare in advance as the APT type ransomware attacks are on the rise [2], [15], [128]. But in the event when the defender is caught off-guard and the attacker has managed to mount a successful attack, the defender still can use Algorithm 2 below to figure out the equilibrium condition given the attack and the resources under siege as shown in Table 7.11. In the algorithm, the outcomes of the attacker for different conditions are denoted by $U_{A1} = \delta P_P$, $U_{A2} = \delta P_P + \beta(1 - P_P)(1 - P_D)$, $U_{A3} = \omega P_P$, and $U_{A4} = \omega P_P + \beta(1 - P_P)(1 - P_D)$. The game is designed as such that each of the players is rational in nature and would play to optimize their outcomes and in the event of an unfavorable condition, they would play to minimize the losses. The prescriptive solutions given by Algorithm 2 and the equilibrium solutions in Table 7.11 not only help the attacker in the preparedness but also in the mitigation if a successful attack has occurred.

The game is designed for an APT type ransomware attack. The algorithm presented here is parameterized and therefore, a defender can plug in the values of the parameters to get the equilibrium strategies for a variety of ransomware attacks. There can be a basic ransomware attacks which doesn't have any contingency plan of attack and doesn't have any perception of the system under attack. This game model works fine for those attacks as well. The parameters for the contingency plan of attack would have a value 0, as the attacker has no contingency attack plan. This game theoretic analysis is an approach to the solutions of the problem faced from a large variety of ransomware attacks.

Algorithm 2 Choosing Defender's strategy based on the Optimized Strategy of Attacker from Table 7.11

```
if $\delta \geq \omega$ then
  if $\gamma > 0$ then
    if $U_{A1} < \lambda$ then
      return 9
    else if $c_A < g_A$ then
      return 1
    else
      return 2
    end if
  else
    if $\beta > 2R$ then
      if $U_{A1} < \lambda$ then
        return 9
      else if $c_A < g_A$ then
        return 1
      else
        return 2
      end if
    else
      if $U_{A2} < \lambda$ then
        return 9
      else if $c_A < g_A$ then
        return 3
      else
        return 4
      end if
    end if
  end if
else
  if $\gamma > 0$ then
    if $U_{A3} < \lambda$ then
      return 9
    else if $c_C < g_C$ then
      return 5
    else
      return 6
    end if
  else
    if $\beta > 2R$ then
      if $U_{A3} < \lambda$ then
        return 9
      else if $c_C < g_C$ then
        return 5
      else
        return 6
```

7.2.9 Summary

The research presented in this chapter is summarized as follows:

- The intrusion detection system and the deception architecture were designed as a defense against APT type malware. But the systems which do not have these defense features, and/or the systems in which these defense features are defeated by the APT type malware, are put to great risk. This leads to questions like - “What to do?”, “When to do?” and “How to do?” with regard to such sophisticated attacks.

- To answer these questions, our research explored solutions using game theoretic analysis. We analyzed the threat scenario of non-APT type ransomware using a sequential game model. We introduced two parameters which would help the defender in making an informed decision when under attack.

- The r_{Rec} parameter is the ratio of the value of recovered resources after payment of ransom to the total value of all the assets owned by the defender. This helps the defender to quantify the importance of the resources under siege. The higher the value of r_{Rec}, the more willing the defender is to pay ransom.

- The second parameter, r_{Rep} helps the defender to guess how “trustworthy” the malware is. The higher the reputation, the more willing is the defender to pay the ransom. More details can be found in the published work [129].

- Through a formal analysis of a non-APT type ransomware, our research provided a preliminary treatment of the mitigation strategies to counter advanced threats.

- We then extended the concept of game theory for more sophisticated APT type ransomware.
- We designed more elaborate sequential game model for multi-stage advanced ransomware attacks, analyzed the threat scenario and traced the optimal strategies of the attacker for different conditions. We came up with equilibrium conditions to maximize the outcome and minimize the losses of the defender with and/or without the defense features. More details can be found in the published work [130].
- Our finding is that the equilibrium is not much sensitive to the changing values of the parameters but are quite sensitive to the changing values of the conditions. If the conditions change owing to change in the values, then the strategies and therefore, the equilibrium would change. But if the change in values of the parameters doesn't change the conditions, the equilibrium strategies wouldn't change.
- The game is designed for rational players who would play to optimize their outcomes and if the conditions are unfavorable, they would play to minimize the losses. We demonstrated the same through a sensitivity analysis. This helps in both preparedness and mitigation of attacks from APT type ransomware.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

The problem addressed in this dissertation is the modeling, detection, deception, and mitigation of Advanced Persistent Threats. The definition of APT had become outdated and wouldn't account for the most recent state-of-the-art sophisticated attacks. Through our research, we took a holistic approach to identify an APT and updated the literature defining an APT. With the newer features being attributed to the APT attacks, a system is needed to detect an intrusion by an APT type malware. We designed AI based IDSes for APT type ransomware using Hidden Markov Model (HMM), ML, and NLP models. We used ML models like NBC, RF, SVM, GB decision trees, and LR, and NLP model BERT to design classifier based IDS. We created a dataset using system call logs by running ransomware in a sandboxed environment. We used Picus Security's simulator, a commercial tool, to collect data from APT type ransomware, viz. Darkside, BlackByte, BlackMatter, REvil and Diavol. We also used HMM to design an IDS. The advantage of using HMM over classifier based IDS is, it not only identifies the attack but also characterizes the attacker. It also helps the defender to identify the state of the attacker so that a defense strategy could be tailored to be best suited to that state of attack.

The IDSes we designed for APT type ransomware needed another layer of protection. This layer of protection was ensured through a deception architecture. We designed a deception architecture called Kidemonas. We used a commercial-off-the-shelf (CoTS) hardware component called trusted platform module (TPM) for designing Kidemonas. The idea is to run a generic APT detection system in an isolated environment outside the purview of the attacker. The aim of the architecture is to silently detect the intrusion and surreptitiously report the same to the defender. This was for detection of generic APT malware. In order to give a layer of protection to the HMM based IDS, we designed a deception based countermeasure called Decepticon. Decepticon is a special case of Kidemonas, which runs an HMM based IDS to detect APT type ransomware as its APT detection system in the isolated environment. The distributed nature of the APT detection system ensures that the burden of detecting any form of APT type malware doesn't fall on a single system. A distributed APT detection architecture and surreptitious reporting has another advantage. When an attack is unfolding, and an intrusion is detected in one of the nodes of the distributed architecture, a preemptive action in other nodes can be taken to prevent the attack on those nodes. This builds an effective defense strategy when faced with an attack from an APT type malware which possess the capabilities of lateral movement and privilege escalation.

The IDSes and the deception architectures were designed as a defense against APT type malware. But the systems which do not have these defense features, and/or the systems in which these defense features are defeated by the APT type malware, are put to great risk. This leads to questions like - “What to do?”, “When to do?” and “How to do?” with regard to such sophisticated attacks. To answer these questions, our research is explored solutions using game theoretic analysis. We analyzed the threat scenario of non-APT type ransomware using a sequential game model. We introduced two parameters which would help the defender in making an informed decision when under attack. The r_{Rec} parameter is the ratio of the value of recovered resources after payment of ransom to the total value of all the assets owned by the defender. This helps the defender to quantify the importance of the resources under siege. The higher the value of r_{Rec}, the more willing the defender is to pay ransom. The second parameter r_{Rep}, helps the defender to guess how “trustworthy” the malware is. The higher the reputation, the more willing is the defender to pay the ransom. Through a formal analysis of a non-APT type ransomware, our research provided a preliminary treatment of the mitigation strategies to counter advanced threats. We then extended the concept of game theory for more sophisticated APT type ransomware. We designed a more elaborate sequential game model for multi-stage advanced ransomware attacks, analyzed the threat scenario, and traced the optimal strategies of the attacker for different conditions. We came up with equilibrium conditions to maximize the outcome and minimize the losses of the defender with and/or without the defense features. Our finding is that the equilibrium is not much sensitive to the changing values of the parameters but are quite sensitive to the changing values of the conditions. If the conditions change owing to change in the values, then the strategies and therefore, the equilibrium would change. But if the change in values of the parameters doesn’t change the conditions, the equilibrium strategies wouldn’t change. The game is designed for rational players who would play to optimize their outcomes and if the conditions are unfavorable, they would play to minimize the losses. We demonstrated the same through a sensitivity analysis. This helps in both preparedness and mitigation of attacks from APT type ransomware.

Organizations can utilize the new results from our research regarding the novel features that more sophisticated APT attacks may come armed with. They can use our research on AI based detection system and the hardware assisted deception mechanism to detect and defend themselves against attacks by APT groups. They can also prepare themselves in advance by utilizing our research on Game Theory based mitigation strategies and also use the same to make informed decisions when under attack. Our research puts forward a holistic approach in defense against attacks from APT groups through detection of attacks, deceiving the attackers, and providing mitigation strategies to the defender when under attack.

8.2 Future Work

Our research is designed around attacks from APT type malware, but it is equally, if not more effective, applicable against basic malware which mostly mount single stage attacks. Even though our current work focuses on APT type ransomware, we would like to extend our scope to other forms of APT malware, and non-APT type sophisticated malware in the future. We would like to study the security and privacy implications on the performance of the entire system. We would also like to design strategies for defense against other types of malware using game theory. We also want to explore the areas of AI for security and security for AI. Preparedness is the key. A cyber attack in today's world is not a question of "if" but "when" [131].

Chapter 9

Publications

9.1 Papers

1. Baksi, Rudra Prasad and Shambhu J. Upadhyaya, 2022. "Game Theoretic Analysis of Ransomware: A Preliminary Study." *In Proceedings of the 8th International Conference on Information Systems Security and Privacy*, ISBN 978-989-758-553-1, ISSN 2184-4356, pages 242-251.

2. Baksi, Rudra Prasad, and Shambhu J. Upadhyaya. "Decepticon: a Theoretical Framework to Counter Advanced Persistent Threats." *Information Systems Frontiers* (2020): 1-17.

3. Baksi, Rudra Prasad, and Shambhu J. Upadhyaya, "Decepticon: A Hidden Markov Model Approach to Counter Advanced Persistent Threats." *International Conference On Secure Knowledge Management In Artificial Intelligence Era.* Springer, Singapore, 2019.

4. Baksi, Rudra Prasad, and Shambhu J. Upadhyaya, "A Comprehensive Model for Elucidating Advanced Persistent Threats", *International Confer-*

ence on Security and Management (SAM '18), Las Vegas, NV, July 2018.

5. Baksi, Rudra Prasad, and Shambhu J. Upadhyaya, "Kidemonas: The Silent Guardian", *Secure Knowledge Management (SKM '17)*, Tampa, FL, October 2017.

9.2 Other Publications

1. Baksi, Rudra Prasad. "Pay or Not Pay? A Game-Theoretical Analysis of Ransomware Interactions Considering a Defender's Deception Architecture." *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S), 2022, pp. 53-54, doi: 10.1109/DSN-S54099.2022.00030.*

Bibliography

[1] Ruchika Mehresh. Schemes for surviving advanced persistent threats. *Faculty of the Graduate School of the University at Buffalo, State University of New York*, 2013.

[2] Rudra Prasad Baksi and Shambhu J Upadhyaya. A comprehensive model for elucidating advanced persistent threats (APT). In *Proceedings of the International Conference on Security and Management (SAM)*, pages 245–251. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2018.

[3] Carl Leonard. 2015 threat report. *Websense Security Labs*, 2015.

[4] Brahim ID Messaoud, Karim Guennoun, Mohamed Wahbi, and Mohamed Sadik. Advanced persistent threat: New analysis driven by life cycle phases and their challenges. In *2016 International Conference on Advanced Communication Systems and Information Security (ACOSIS)*, pages 1–6. IEEE, 2016.

[5] James T Bennett, Ned Moran, and Nart Villeneuve. Poison ivy: Assessing damage and extracting intelligence. *FireEye Threat Research Blog*, 2013.

[6] J Vukalović and D Delija. Advanced persistent threats-detection and defense. In *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1324–1330. IEEE, 2015.

[7] Ponemon Institute LLC. The state of advanced persistent threats. *Ponemon Institute Research Report*, December 2013.

[8] Z Clark. The worm that spreads wanacrypt0r. *Malwarebytes Labs*, May 2017.

[9] Secureworks. Wcry ransomware campaign. *Secureworks Inc.*, May 2017.

[10] Stuart E. Madnick and John J. Donovan. Application and analysis of the virtual machine approach to information system security and isolation. In *Proceedings of the Workshop on Virtual Computer Systems*, pages 210–224, New York, NY, USA, 1973. ACM.

[11] Jacob R Lorch, Yi-Min Wang, Chad Verbowski, Helen J Wang, and Samuel King. Isolation environment-based information access, September 20 2011. US Patent 8,024,815.

[12] Boldizsár Bencsáth, Gábor Pék, Levente Buttyán, and Mark Felegyhazi. The cousins of stuxnet: Duqu, flame, and gauss. *Future Internet*, 4(4):971–1003, 2012.

[13] Nicolas Falliere, Liam O Murchu, and Eric Chien. W32. stuxnet dossier. *White paper, Symantec Corp., Security Response*, 5(6):29, 2011.

[14] Ralph Langner. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy*, 9(3):49–51, 2011.

[15] Wira Zanoramy A Zakaria, Mohd Faizal Abdollah, Othman Mohd, and Aswami Fadillah Mohd Ariffin. The rise of ransomware. In *Proceedings of the 2017 International Conference on Software and e-Business*, pages 66–70. ACM, 2017.

[16] Adel Alshamrani, Sowmya Myneni, Ankur Chowdhary, and Dijiang Huang. A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities. *IEEE Communications Surveys & Tutorials*, 21(2):1851–1877, 2019.

[17] Ruchika Mehresh and Shambhu Upadhyaya. A deception framework for survivability against next generation cyber attacks. In *Proceedings of the International Conference on Security and Management (SAM)*, page 1. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2012.

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

[19] LogRhythm. The apt lifecycle and its log trail. *Tech. Rep.*, July 2013.

[20] Awais Rashid, Rajiv Ramdhany, Matthew Edwards, Sarah Kibirige Mukisa, Muhammad Ali Babar, David Hutchison, and Ruzanna Chitchyan. Detecting and preventing data exfiltration, 2014.

[21] Alina Bizga. Ransomware attack confirmed by australia-based beverage manufacturer. *Security Boulevard*, 06 2020.

[22] TOI Correspondent. Fact check: Iit madras servers were under ransomware attack? *Times of India*, 02 2020.

[23] T E Raja Simhan. Iit-m's email servers shut down, raises malware concerns. *Business Line*, 02 2020.

[24] Tomas Meskauskas. How to uninstall wannacash ncov ransomware? *PC Risk*, 04 2020.

[25] Melissa Robbins. Cyberattack hits indian nuclear plant. *Arms Control Association*, 12 2019.

[26] David Strom. Beware of blackbyte ransomware. *Avast Blog*, February 2022.

[27] Hüseyin Can YÜCEEL. Ttps used by blackbyte ransomware targeting critical infrastructure. *Picus Blog*, February 2022.

[28] Trend Micro Research. What we know about the darkside ransomware and the us pipeline attack. *Trend Micro*, 05 2021.

[29] Trend Micro Research. Revil. *Trend Micro*, 12 2021.

[30] Ken Dilanian. Code in huge ransomware attack written to avoid computers that use russian, says new report. *NBC News*, 07 2021.

[31] CISA Original Release. Blackmatter ransomware. *Cybersecurity & Infrastructure Security Agency*, October 2021.

[32] Wayne Labs. Blackmatter ransomware targets food/agriculture sector. *Food Engineering Magazine*, 10 2021.

[33] THE DFIR REPORT. Real intrusions by real attackers, the truth behind the intrusion: Diavol ransomware. *THE DFIR REPORT*, 12 2021.

[34] Dor Neemani and Asaf Rubinfeld. Diavol - a new ransomware used by wizard spider? *FORTINET Blog*, 07 2021.

[35] Lawrence Abrams. Fbi links diavol ransomware to the trickbot cybercrime group. *BleepingComputer*, 01 2022.

[36] Kevin Poulsen Robert McMillan and Dustin Volz. Secret world of pro-russia hacking group exposed in leak. *The Wall Street Journal*, 03 2022.

[37] Oluwasegun Ishaya Adelaiye, Aminat Showole, and Silas Ageebee Faki. Evaluating advanced persistent threats mitigation effects: A review. *International Journal of Information Security Science*, 7(4):159–171, 2018.

[38] TCG. Tpm main specification. *Trusted Computing Group*, 03 2011.

[39] Eusebius(Guillaume Piolle). Simplified schema of a trusted platform module (tpm). *Own Work*, September 2008.

[40] Victor Costan and Srinivas Devadas. Intel sgx explained. *IACR Cryptology ePrint Archive*, 2016(086):1–118, 2016.

[41] Jinsoo Jang, Changho Choi, Jaehyuk Lee, Nohyun Kwak, Seongman Lee, Yeseul Choi, and Brent Byunghoon Kang. Privatezone: Providing a private execution environment using arm trustzone. *IEEE Transactions on Dependable and Secure Computing*, 15(5):797–810, 2016.

[42] Carlton Shepherd, Ghada Arfaoui, Iakovos Gurulian, Robert P Lee, Konstantinos Markantonakis, Raja Naeem Akram, Damien Sauveron, and Emmanuel Conchon. Secure and trusted execution: Past, present, and future-a critical review in the context of the internet of things and cyber-physical systems. In *2016 IEEE Trustcom/BigDataSE/ISPA*, pages 168–177. IEEE, 2016.

[43] ChongChong Zhao, Daniyaer Saifuding, Hongliang Tian, Yong Zhang, and ChunXiao Xing. On the performance of intel sgx. In *2016 13th Web Information Systems and Applications Conference (WISA)*, pages 184–187. IEEE, 2016.

[44] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[45] Andrej Ljolje and Stephen E Levinson. Development of an acoustic-phonetic hidden markov model for continuous speech recognition. *IEEE Transactions on signal processing*, 39(1):29–39, 1991.

[46] Mou-Yen Chen, Amlan Kundu, and Jian Zhou. Off-line handwritten word recognition using a hidden markov model type stochastic network. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 16(5):481–496, 1994.

[47] Satheesh Kumar Sasidharan and Ciza Thomas. A survey on metamorphic malware detection based on hidden markov model. In *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 357–362. IEEE, 2018.

[48] Irina Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.

[49] Theodoros Evgeniou and Massimiliano Pontil. Support vector machines: Theory and applications. volume 2049, pages 249–257, 01 2001.

[50] Chao-Ying Joanne Peng, Kuk Lida Lee, and Gary M Ingersoll. An introduction to logistic regression analysis and reporting. *The journal of educational research*, 96(1):3–14, 2002.

[51] Alexey Natekin and Alois Knoll. Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, 7:21, 2013.

[52] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[53] Stephen D Bay, Dennis Kibler, Michael J Pazzani, and Padhraic Smyth. The uci kdd archive of large data sets for data mining research and experimentation. *ACM SIGKDD explorations newsletter*, 2(2):81–85, 2000.

[54] Zeeshan Ahmad, Adnan Shahid Khan, Cheah Wai Shiang, Johari Abdullah, and Farhan Ahmad. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, 32(1):e4150, 2021.

[55] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*, pages 1–6. Ieee, 2009.

[56] L Dhanabal and SP Shantharajah. A study on nsl-kdd dataset for intrusion detection system based on classification algorithms. *International journal of advanced research in computer and communication engineering*, 4(6):446–452, 2015.

[57] Mouhammd Alkasassbeh, Ghazi Al-Naymat, Ahmad Hassanat, and Mohammad Almseidin. Detecting distributed denial of service attacks using data mining techniques. *International Journal of Advanced Computer Science and Applications*, 7(1):436–445, 2016.

[58] Mohammad Almseidin, Maen Alzubi, Szilveszter Kovacs, and Mouhammd Alkasassbeh. Evaluation of machine learning algorithms for intrusion detection system. In *2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY)*, pages 000277–000282, 2017.

[59] Anish Halimaa and K Sundarakantham. Machine learning based intrusion detection system. In *2019 3rd International conference on trends in electronics and informatics (ICOEI)*, pages 916–920. IEEE, 2019.

[60] Pankaj Kumar Keserwani, Mahesh Chandra Govil, Emmanuel S Pilli, and Prajjval Govil. A smart anomaly-based intrusion detection system for the internet of things (iot) network using gwo–pso–rf model. *Journal of Reliable Intelligent Environments*, 7(1):3–21, 2021.

[61] Ediga Sathyanarayana Phalguna Krishna and Thangavelu Arunkumar. Hybrid particle swarm and gray wolf optimization algorithm for iot intrusion detection system. *International Journal of Intelligent Engineering & Systems*, 2021.

[62] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[63] Trung Kien Tran and Hiroshi Sato. Nlp-based approaches for malware classification from api sequences. In *2017 21st Asia Pacific Symposium on Intelligent and Evolutionary Systems (IES)*, pages 101–105. IEEE, 2017.

[64] Pejman Najafi, Daniel Koehler, Feng Cheng, and Christoph Meinel. Nlp-based entity behavior analytics for malware detection. In *2021 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pages 1–5. IEEE, 2021.

[65] Qinqin Wang, Hanbing Yan, and Zhihui Han. Explainable apt attribution for malware using nlp techniques. In *2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)*, pages 70–80. IEEE, 2021.

[66] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[67] Abir Rahali and Moulay A Akhloufi. Malbert: Using transformers for cybersecurity and malicious software detection. *arXiv preprint arXiv:2103.03806*, 2021.

[68] Nicoló Andronio, Stefano Zanero, and Federico Maggi. Heldroid: Dissecting and detecting mobile ransomware. In *international symposium on recent advances in intrusion detection*, pages 382–404. Springer, 2015.

[69] Andrea Continella, Alessandro Guagnelli, Giovanni Zingaro, Giulio De Pasquale, Alessandro Barenghi, Stefano Zanero, and Federico Maggi. Shieldfs: a self-healing, ransomware-aware filesystem. In *Proceedings of the 32nd annual conference on computer security applications*, pages 336–347, 2016.

[70] Nolen Scaife, Henry Carter, Patrick Traynor, and Kevin RB Butler. Cryptolock (and drop it): stopping ransomware attacks on user data. In *2016 IEEE 36th international conference on distributed computing systems (ICDCS)*, pages 303–312. IEEE, 2016.

[71] Eugene Kolodenker, William Koch, Gianluca Stringhini, and Manuel Egele. Paybreak: Defense against cryptographic ransomware. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 599–611, 2017.

[72] Edward Cartwright, Julio Hernandez Castro, and Anna Cartwright. To pay or not: game theoretic models of ransomware. *Journal of Cybersecurity*, 5(1):tyz009, 2019.

[73] Avinash K Dixit and Susan Skeath. *Games of strategy: Fourth international student edition*. WW Norton & Company, 2015.

[74] Reinhard Selten. A simple game model of kidnapping. In *Mathematical Economics and Game Theory*, pages 139–155. Springer, 1977.

[75] Theodoros Spyridopoulos, George Oikonomou, Theo Tryfonas, and Mengmeng Ge. Game theoretic approach for cost-benefit analysis of malware proliferation prevention. In *IFIP International Information Security Conference*, pages 28–41. Springer, 2013.

[76] MHR Khouzani, Saswati Sarkar, and Eitan Altman. A dynamic game solution to malware attack. In *2011 Proceedings IEEE INFOCOM*, pages 2138–2146. IEEE, 2011.

[77] Weiwei Zhou and Bin Yu. A cloud-assisted malware detection and suppression framework for wireless multimedia system in iot based on dynamic differential game. *China Communications*, 15(2):209–223, 2018.

[78] Hayreddin Çeker, Jun Zhuang, Shambhu Upadhyaya, Quang Duy La, and Boon-Hee Soong. Deception-based game theoretical approach to mitigate dos attacks. In *International Conference on Decision and Game Theory for Security*, pages 18–38. Springer, 2016.

[79] Adrian Pauna. Improved self adaptive honeypots capable of detecting rootkit malware. In *2012 9th International Conference on Communications (COMM)*, pages 281–284. IEEE, 2012.

[80] Jun Zhuang and Vicki M Bier. Reasons for secrecy and deception in homeland-security resource allocation. *Risk Analysis: An International Journal*, 30(12):1737–1743, 2010.

[81] Jun Zhuang and Vicki M Bier. Secrecy and deception at equilibrium, with applications to anti-terrorism resource allocation. *Defence and Peace Economics*, 22(1):43–61, 2011.

[82] Jun Zhuang, Vicki M Bier, and Oguzhan Alagoz. Modeling secrecy and deception in a multiple-period attacker–defender signaling game. *European Journal of Operational Research*, 203(2):409–418, 2010.

[83] Quanyan Zhu and Stefan Rass. On multi-phase and multi-stage game-theoretic modeling of advanced persistent threats. *IEEE Access*, 6:13958–13971, 2018.

[84] Rudra Prasad Baksi and Shambhu J Upadhyaya. Kidemonas: The silent guardian. *arXiv preprint arXiv:1712.00841*, 2017.

[85] Rudra Prasad Baksi and Shambhu J Upadhyaya. Decepticon: A hidden markov model approach to counter advanced persistent threats. In *International Conference On Secure Knowledge Management In Artificial Intelligence Era*, pages 38–54. Springer, 2019.

[86] Rudra P Baksi and Shambhu J Upadhyaya. Decepticon: a theoretical framework to counter advanced persistent threats. *Information Systems Frontiers*, 23(4):897–913, 2021.

[87] Mohammad Reza Faghani and Uyen Trang Nugyen. Modeling the propagation of trojan malware in online social networks. *arXiv preprint arXiv:1708.00969*, 2017.

[88] Grant Sanderson. Animated math. *3blue1brown*, 2020.

[89] DJ Daley and JM Gani. Cambridge studies in mathematical biology. *Epidemic modelling: an introduction*, 1999.

[90] Pradip De, Yonghe Liu, and Sajal K Das. An epidemic theoretic framework for vulnerability analysis of broadcast protocols in wireless sensor networks. *IEEE Transactions on mobile Computing*, 8(3):413–425, 2008.

[91] Roberto Di Pietro and Nino Vincenzo Verde. Introducing epidemic models for data survivability in unattended wireless sensor networks. In *2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pages 1–6. IEEE, 2011.

[92] Frank Spitzer. *Principles of random walk*, volume 34. Springer Science & Business Media, 2013.

[93] Gjergji Zyba, Geoffrey M Voelker, Michael Liljenstam, András Méhes, and Per Johansson. Defending mobile phones from proximity malware. In *IEEE INFOCOM 2009*, pages 1503–1511. IEEE, 2009.

[94] Steve Mansfield-Devine. Ddos goes mainstream: how headline-grabbing attacks could make this threat an organisation's biggest nightmare. *Network Security*, 2016(11):7–13, 2016.

[95] Anas AlMajali and Waleed Dweik. Analysing and modelling worm propagation speed in the smart grid communication infrastructure. *International Journal of Embedded Systems*, 11(1):11–21, 2019.

[96] Endgame. Wcry/wannacry technical analysis. *elastic*, 05 2017.

[97] Joan Daemen and Vincent Rijmen. The rijndael block cipher: Aes proposal. In *First candidate conference (AeS1)*, pages 343–348, 1999.

[98] Prerna Mahajan and Abhishek Sachdeva. A study of encryption algorithms AES, DES and RSA for security. *Global Journal of Computer Science and Technology*, 2013.

[99] Andy Greenberg. Hackers are trying to reignite wannacry with nonstop botnet attacks. *Wired Security*, 05 2017.

[100] G. D. Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[101] Lawrence Rabiner and B Juang. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16, 1986.

[102] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[103] Picus Security the complete security control validation platform. `https://www.picussecurity.com/`.

[104] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[105] Satire VI Juvenal. lines 347–348.“. *Sed quis custodiet ipsos custodes*, 2nd Century CE.

[106] quis custodiet ipsos custodes? https://www.merriam-webster.com/dictionary/quis%20custodiet%20ipsos%20custodes%3F. Merriam-Webster.com Dictionary accessed July 15, 2021.

[107] Ramkumar Chinchani, Shambhu Upadhyaya, and Kevin Kwiat. A tamper-resistant framework for unambiguous detection of attacks in user space using process monitors. In *First IEEE International Workshop on Information Assurance, 2003. IWIAS 2003. Proceedings.*, pages 25–34. IEEE, 2003.

[108] R. Chinchani, A. Iyer, H.Q. Ngo, and S. Upadhyaya. Towards a theory of insider threat assessment. In *2005 International Conference on Dependable Systems and Networks (DSN'05)*, pages 108–117, 2005.

[109] Rolf Oppliger and Ruedi Rytz. Does trusted computing remedy computer security problems? *IEEE Security & Privacy*, 3(2):16–19, 2005.

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

[111] Markus Maybaum and Jens Toelle. Arming the trusted platform module pro-active system integrity monitoring focussing on peer system notification. In *MILCOM 2015-2015 IEEE Military Communications Conference*, pages 1584–1589. IEEE, 2015.

[112] Frederico Araujo, Kevin W Hamlen, Sebastian Biedermann, and Stefan Katzenbeisser. From patches to honey-patches: Lightweight attacker misdirection, deception, and disinformation. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 942–953, 2014.

[113] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. Rsa-oaep is secure under the rsa assumption. In *Annual International Cryptology Conference*, pages 260–274. Springer, 2001.

[114] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. Rsa-oaep is secure under the rsa assumption. *Journal of Cryptology*, 17(2):81–104, 2004.

[115] Ping Chen, Lieven Desmet, and Christophe Huygens. A study on advanced persistent threats. In *IFIP International Conference on Communications and Multimedia Security*, pages 63–72. Springer, 2014.

[116] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 92–111. Springer, 1994.

[117] Rudra P Baksi and Shambhu J Upadhyaya. Decepticon: a theoretical framework to counter advanced persistent threats. *Information Systems Frontiers*, pages 1–17, 2020.

[118] Aaron Zimba and Mumbi Chishimba. Understanding the evolution of ransomware: paradigm shifts in attack structures. *International Journal of computer network and information security*, 11(1):26, 2019.

[119] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J.M. Wing. Automated generation and analysis of attack graphs. In *Proceedings 2002 IEEE Symposium on Security and Privacy*, pages 273–284, 2002.

[120] Oleg Sheyner and Jeannette Wing. Tools for generating and analyzing attack graphs. volume 3188, pages 344–372, 11 2003.

[121] John C Harsanyi. Games with incomplete information. In *Evolution and Progress in Democracies*, pages 43–55. Springer, 1994.

[122] Sundar Krishnan and Mingkui Wei. Scada testbed for vulnerability assessments, penetration testing and incident forensics. In *2019 7th International Symposium on Digital Forensics and Security (ISDFS)*, pages 1–6. IEEE, 2019.

[123] Morgan A Zantua, Viatcheslav Popovsky, Barbara Endicott-Popovsky, and Fred B Holt. Discovering a profile for protect and defend: Penetration testing. In *International Conference on Learning and Collaboration Technologies*, pages 530–540. Springer, 2018.

[124] Mike Auty. Anatomy of an advanced persistent threat. *Network Security*, 2015(4):13–16, 2015.

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

[126] Eric Baize. Developing secure products in the age of advanced persistent threats. *IEEE Security Privacy*, 10(3):88–92, 2012.

[127] Yu-Kyung Kim, Jemin Justin Lee, Myong-Hyun Go, and Kyungho Lee. Analysis of the asymmetrical relationships between state actors and apt threat groups. In *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 695–700, 2020.

[128] Ibrar Yaqoob, Ejaz Ahmed, Muhammad Habib ur Rehman, Abdelmuttlib Ibrahim Abdalla Ahmed, Mohammed Ali Al-garadi, Muhammad Imran, and Mohsen Guizani. The rise of ransomware and emerging security challenges in the internet of things. *Computer Networks*, 129:444–458, 2017.

[129] Rudra Prasad Baksi and Shambhu J Upadhyaya. Game theoretic analysis of ransomware: A preliminary study. In *ICISSP*, pages 242–251, 2022.

[130] Rudra Prasad Baksi. Pay or Not Pay? A game-theoretical analysis of ransomware interactions considering a defender's deception architecture. In *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S)*, pages 53–54, 2022.

[131] Not "If" But "When"—The not "if" but "when"—the ever increasing threat of a data breach in 2021. https://www.jdsupra.com/legalnews/not-if-but-when-the-ever-increasing-8569092. July 15, 2021.