Information Systems Frontiers
https://doi.org/10.1007/s10796-020-10087-4

Decepticon: a Theoretical Framework to Counter Advanced Persistent Threats

Rudra P. Baksi[1] · **Shambhu J. Upadhyaya**[1]

Accepted: 27 October 2020

Abstract

Deception has been proposed in the literature as an effective defense mechanism to address Advanced Persistent Threats (APT). However, administering deception in a cost-effective manner requires a good understanding of the attack landscape. The attacks mounted by APT groups are highly diverse and sophisticated in nature and can render traditional signature based intrusion detection systems useless. This necessitates the development of behavior oriented defense mechanisms. In this paper, we develop Decepticon (Deception-based countermeasure), a Hidden Markov Model based framework where the indicators of compromise (IoC) are used as the observable features to aid in detection. This theoretical framework also includes several models to represent the spread of APTs in a computer system. The presented framework can be used to select an appropriate deception script when faced with APTs or other similar malware and trigger an appropriate defensive response. The effectiveness of the models in a networked system is illustrated by considering a real APT type ransomware.

Keywords Advanced Persistent Threats (APT) · Computer security · Cyber-security · Hidden Markov Model (HMM) · Ransomware

1 Introduction

Advanced Persistent Threats (APT) are a form of quiet invaders (Mehresh 2013b) and are a lingering nuisance 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 (Baksi and Upadhyaya 2018). 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 (Leonard 2015; Messaoud et al. 2016). 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 (Baksi and Upadhyaya 2018).

According to FireEye, 4,192 attacks were detected in 2013, which were mounted by groups that can confidently be classified as APT groups (Bennett et al. 2013). 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 (Vukalović and Delija 2015). 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 (LLC 2013). 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 (Baksi and Upadhyaya 2018). 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 (Clark 2017; Secureworks 2017).

✉ Rudra P. Baksi
rudrapra@buffalo.edu

Shambhu J. Upadhyaya
shambhu@buffalo.edu

[1] Department of Computer Science and Engineering, University at Buffalo, SUNY, Buffalo, NY 14260, USA

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. Lesser 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 (Madnick and Donovan 1973). Isolation can be achieved through software or hardware (Lorch et al. 2011). 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 (Bencsáth et al. 2012; Falliere et al. 2011; Langner 2011). 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.

In this paper, the aforementioned research ideas, viz. isolation and deception, are used to confront intricate attacks arising from APT groups. The paper puts forward a basic architecture, which is aimed at deceiving the attacker into believing in its success, while surreptitiously triggering a fix to thwart the attack. To make the defense-system cost-effective, the defender must have knowledge about the attack scenario. The information about the spread and the status of a malware helps a defender to develop an efficient attack averting strategy. This paper presents Decepticon, a Hidden Markov Model (HMM) based deceptive countermeasure which uses certain indicators of compromise (IoC) for detection and mitigation of APTs. With the inclusion of several malware spread models for the understanding of the IoCs, this HMM-based detection system serves largely as a theoretical framework with a ransomware as use case. The major contributions of the paper are the design of a hardware-based defense framework and the development of a model for a HMM-based APT type ransomware detection tool. The framework is a special case of the Kidemonas architecture (Baksi and Upadhyaya 2017) and uses the concept of smart-box from Mehresh and Upadhyaya (2012) for surreptitious reporting and triggering of defensive scripts on being attacked. The paper is organized as follows. Section 2 discusses related work in this area. Section 3 discusses three malware spread models and analyzes their suitability for the understanding of the indicators of compromise (IoC). Section 4 presents the new deception architecture. Section 5 describes the HMM based detection system. Section 6 shows the working of the Decepticon architecture. Section 7 discusses the applicability of the system in the presence of an APT type ransomware, viz. WannaCry. Finally, Section 8 concludes the paper and paves way for future work.

2 Preliminaries and Related Work

In this section, some preliminaries are given on malware, APT, the Trusted Platform Module (TPM) hardware, deception, and HMM, which are used to develop the Decepticon architecture in Section 4. Related work on these topics is also reviewed.

Malware created by the APT groups 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 (Hutchins et al. 2011). 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 installation of back-doors to maintain a 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 (LogRhythm 2013). Lancaster University describes APT through a three stage life cycle (Rashid et al. 2014). Baksi and Upadhyaya (2018) describe APT through a set of five characteristics exhibited by a sophisticated malware.

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* (Zakaria et al. 2017). 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 the ransomware under consideration in this paper is a more advanced form of malware. In addition to possessing the features of a primary ransomware, they are more sophisticated by having a contingency plan of attack on being discovered (Baksi and Upadhyaya 2018). 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 (Bizga 2020; Correspondent 2020; Simhan 2020). Off late, the malware WannaCash is also causing trouble to the cyber-world (Meskauskas 2020). Another example of a recent attack is the one on Indian nuclear power plants causing significant data breaches (Robbins 2019).

The the Trusted Platform Module (TPM) is a hardware component designed following the guidelines of the security consortium, the Trusted Computing Group (TCG) (TCG 2011). The TPM comes with essential cryptographic potential. It can generate cryptographic keys, both symmetric and asymmetric keys. It also 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 1 gives a simplified schematic of the TPM version 1.2, the specifications of which are laid down by the TCG. 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 components which come with certain functionalities in addition to the ones already mentioned for TPMs (Costan and Devadas 2016; Jang et al. 2016; Shepherd et al. 2016; Zhao et al. 2016). They provide a *Trusted Execution Environment (TEE)*, which are 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.

Deception can often be considered as a potential weapon against sophisticated attacks and it is an important area of research. In Ċeker et al. (2016), the authors use deception to fight against denial of service (DoS) attacks. The authors have analyzed the deceptive strategy using a game-theoretic model based on the signaling game with perfect Bayesian equilibrium (PBE) to investigate the implications of deception to counter the attacks. Deception as a defensive strategy has been used in Pauna (2012), wherein the authors have used deceptive measures to lure the attackers to high-interaction honeypots for designing a malware detection system.

Hidden Markov Models (HMM) have been historically used for speech recognition (Rabiner 1989a; Ljolje and Levinson 1991). It has also been applied for handwritten character and word recognition (Chen et al. 1994). 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

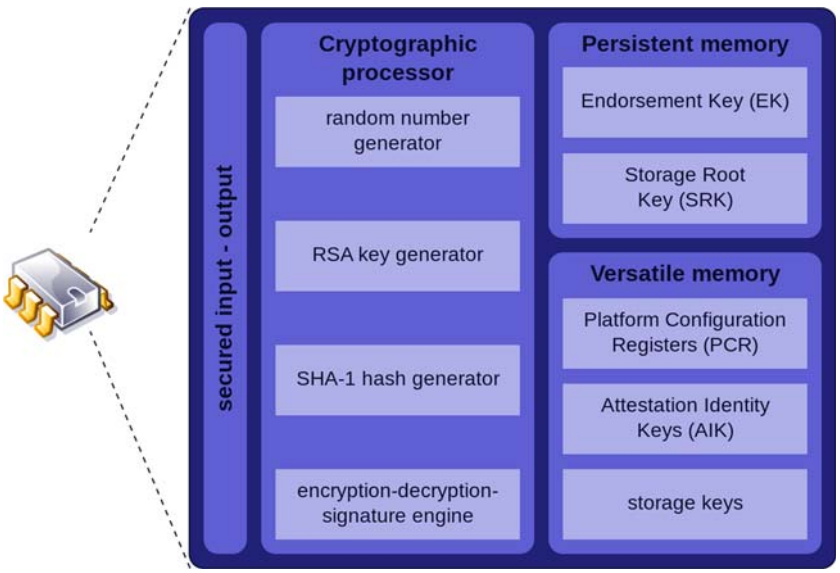


Fig. 1 A Simplified Schema of TPM (Piolle 2008)

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 (Kumar Sasidharan and Thomas 2018).

3 The Malware Spread Model

The threat being considered in this paper 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 being discussed in this section would help the defender in detecting the malware in its early stages. We consider exponential growth, epidemic spread model, and random walk for the spread of the ransomware and discuss their applicability in attack detection.

3.1 Model 1 - Exponential Growth

The exponential growth model (Faghani and Nugyen 2017; Sanderson) can be used to model those malware which infect 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

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

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

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

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 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.

3.2 Model 2 - Epidemic Spread Model

The spread of diseases which become epidemic are perceived mathematically using the epidemic spread model (Daley and Gani 1999). 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 (De et al. 2008; Di Pietro and Verde 2011). It is similar to the exponential growth model but more detailed. It takes into account both the susceptible systems and/or devices as well as the recovered ones. It gives a more comprehensive insight to 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 assumptions, we move forward with the SIR model.

For the model we use $N(t)$, $S(t)$, $I(t)$, and $R(t)$ to denote the total number of nodes (that have the exploitable vulnerabilities), number of susceptible nodes, number of infected nodes, and number of recovered nodes, respectively in a networked system. This give us $N(t) = S(t) + I(t) + R(t)$. In the same network, there can be nodes which do not have the vulnerabilities that can be exploited by the malware and are not relevant to our model. 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 (De et al. 2008):

$$\frac{dS(t)}{dt} = -\delta \cdot S(t) \cdot I(t)$$

$$\frac{dI(t)}{dt} = \delta \cdot S(t) \cdot I(t) - \gamma \cdot I(t)$$

$$\frac{dR(t)}{dt} = \gamma \cdot I(t)$$

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 (Baksi and Upadhyaya 2018). The Gamma value (γ) gives a sense of the weaknesses of the malware and subsequently might help in discovering the kill-switch to thwart the attack (Baksi and Upadhyaya 2018). Both δ and γ values, if correctly estimated, then become important elements in designing a behavioral based intrusion detection and/or intrusion prevention system.

3.3 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 (Spitzer 2013; Zyba et al. 2009). The APT type ransomware, which is being considered as the malware for this paper, 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 manifests 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.

3.4 Model Comparison and Applicability

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 the defender comprehend the systems at risk and the timing constraints regarding the exploitation of the vulnerabilities (Baksi and Upadhyaya 2018). The exponential growth model gives the growth rate, the inflection point and the extent of the infection. The limitation of this model is its applicability to systems only of large population size. It means that small or medium-sized systems of networked nodes may not be suitably modeled with the exponential growth characteristic. 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. This gives the defender a valuable information regarding the time for preparedness. But again, the drawback is, it would be difficult to apply this model for a system with a very large population size which is at risk. 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. The model can be

used for large as well moderately sized population. The applicability of the SIR model for systems to defend against APT type malware is discussed in Section 7.

4 Decepticon - The Architecture

The Trusted Computing Group (TCG) laid down the specifications for Trusted Platform Module (TPM) with an intention of creating a trusted computing environment (TCG 2011). These specifications were capitalized on to create a deception based architecture, Kidemonas (Baksi and Upadhyaya 2017), 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.

In this paper the capabilities of Kidemonas are extended to realize a cost-effective system to detect intrusions from APTs. In a business enterprise type environment, 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 (Mehresh and Upadhyaya 2012). 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 defensive strategies depending upon the characteristics of the malware.

Figure 2 represents the hardware based defense architecture called Decepticon whose aim is to deceive APT type ransomware. Kidemonas (Baksi and Upadhyaya 2017) 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.

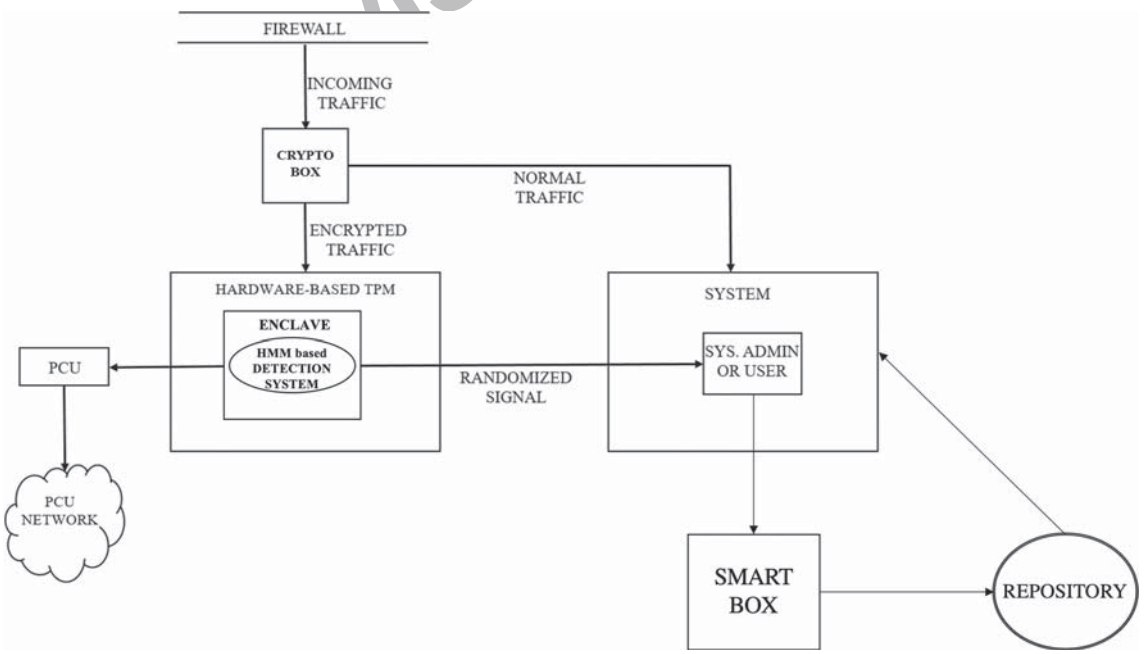


Fig. 2 Decepticon Architecture

The firewall (Fig. 2) 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 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 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.

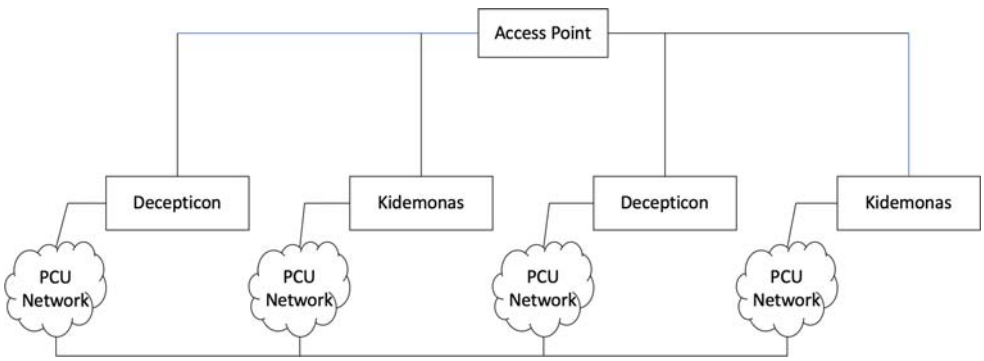
Figure 3 shows a snapshot of a networked system in a corporate environment. This representation shows multiple computing units connected to a single access point. Each computing unit is connected to other computing units through the PCU network, which is also used to inform each other of any form of intrusion in the system. Figure 3 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.

5 A Hidden Markov Model Based Detection Scheme

The threat model under consideration in this paper 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 useless. 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 (Baksi and Upadhyaya 2018). 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 the attacker might resort to, if it believes 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, it 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 (Mehresh 2013b) 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.

Fig. 3 The System



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 3) 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 4 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:

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

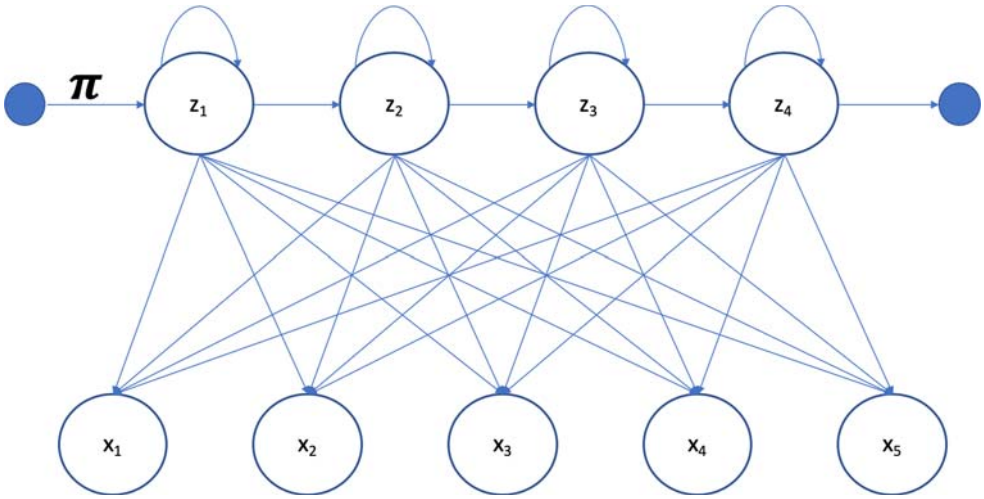


Fig. 4 HMM based Ransomware Detection Model

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

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 paper 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 (Baksi and Upadhyaya 2018).

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.

6 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

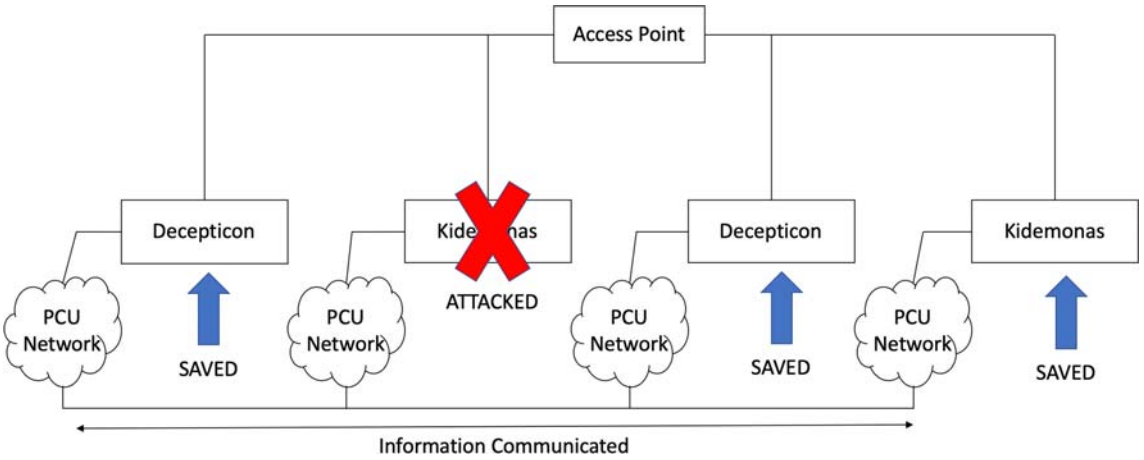


Fig. 5 A Scenario wherein One System is under Attack

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. 3, 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. 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 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, given the feature set

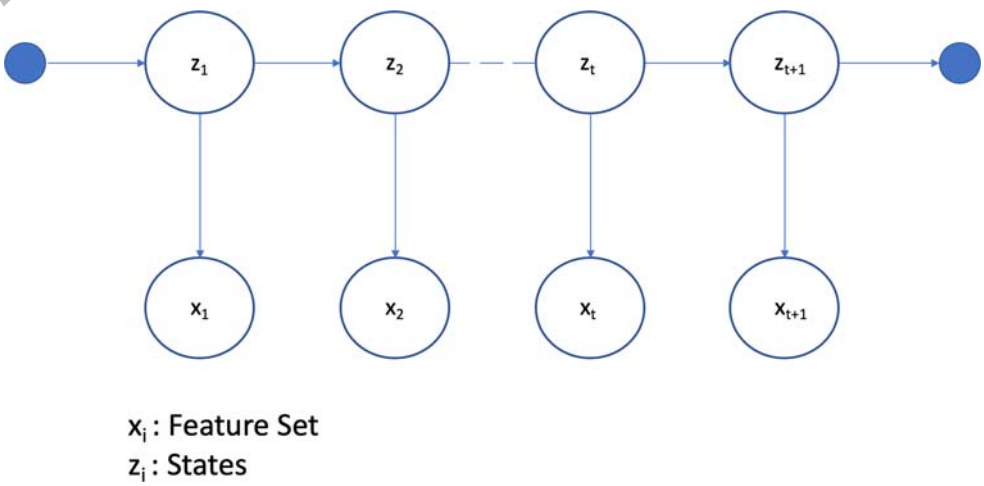


Fig. 6 Classifier Based On-line Predictive Model

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 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 paper 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. We now discuss a use case to illustrate how the proposed model can be used in the real world.

7 WannaCry: a Use Case

We consider WannaCry (Endgame 2017; Secureworks 2017) to illustrate the effectiveness of the models developed in this paper and the associated detection framework. It was created by the APT group named Lazarus from North Korea (Endgame 2017). We first look 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 7 shows the execution flow of WannaCry (Endgame 2017). 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 (Secureworks 2017). 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. 7. 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 7 shows that after infiltration, the main task of the malware is to encrypt the system using the AES encryption scheme (Daemen and Rijmen 1999; Mahajan and Sachdeva 2013a). 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. 7) has been triggered, are the ones that formed the population of *recovered* systems.

The spread model used for WannaCry in this paper is the SIR model. The important data we need are 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 are then used to compute the rates of change of susceptible, infected and recovered nodes as described in Section 3.2.

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 numbe r infected nodes. Through further investigation, one can approximate the type of resources under attack.

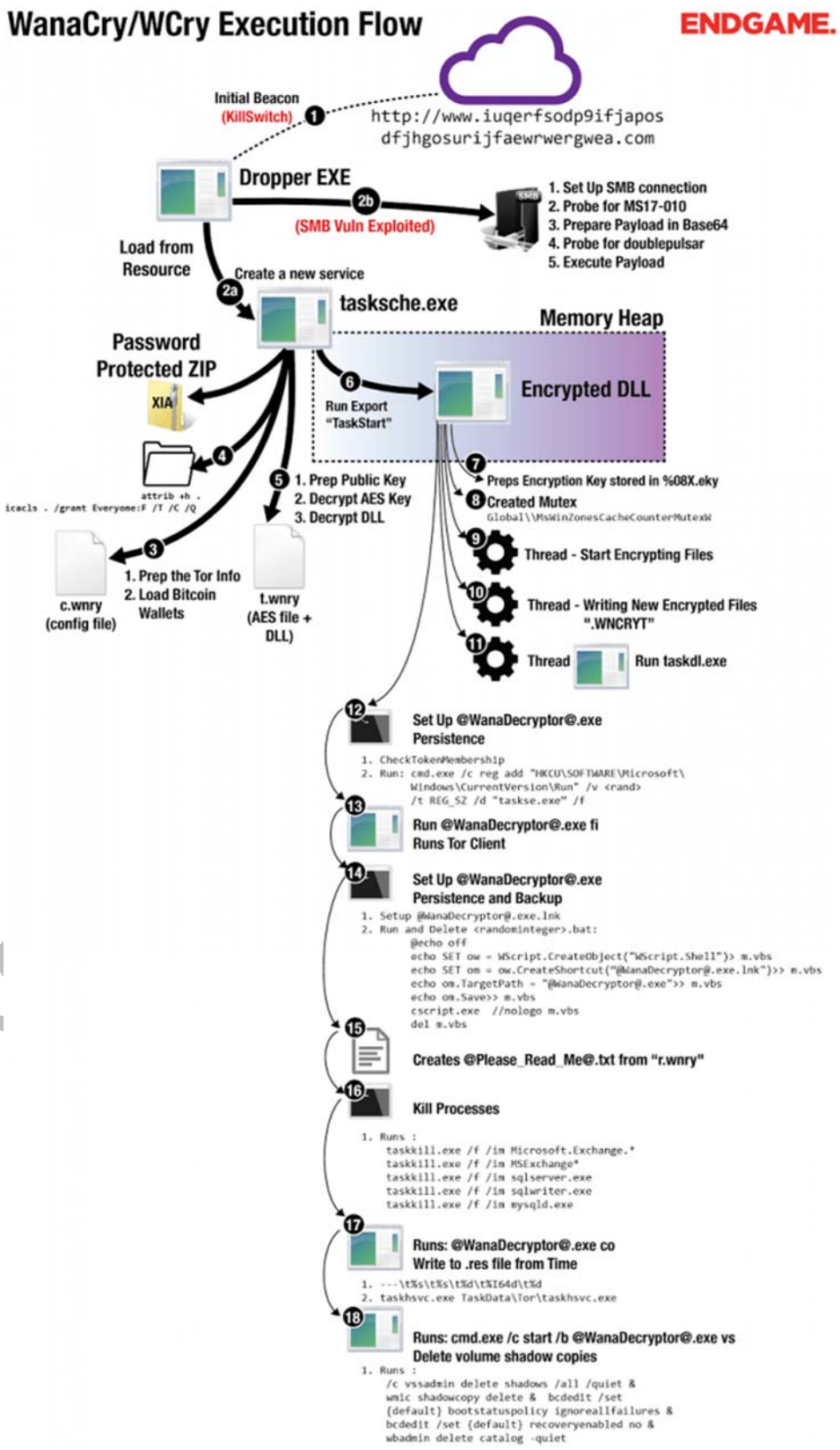


Fig. 7 WannaCry Execution Flow (Endgame 2017)

- 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 1 gives us the parameters required for both the spread model and the HMM based detection model.

The first step now is the identification of the observable features/states (as discussed in Section 5).

- 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 (Secureworks 2017). 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" (Greenberg 2017).

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,

Table 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)

Table 2 Illustrative transition probabilities for each hidden state

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

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 and the transition probabilities will be given as stated in matrix A in Section 5.

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 2.

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. 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 3.

Table 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

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 random 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 4.

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 (Forney 1973; Rabiner and Juang 1986, 1989a, b).

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

Table 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

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".

8 Conclusion and Future Work

The paper describes a new architecture which incorporates the idea of isolation to secure a networked system against advanced persistent threats (APT). It employs deception as a defense technique through the use of a hardware-based TPM. Deception is used to surreptitiously report the detection of an attack to the system administrator. This dupes the attacker into believing in its silent invasion while giving the defender valuable time to prepare for preventive strategy to thwart the attack. In this paper, we also described the development of an HMM based APT type ransomware detection tool.

Prior to the development of a prototype of the HMM based detection tool, a test-bench needs to be created to analyze and validate the model. Initially, the experiments can be conducted via simulation using customized software. Currently, commercially available TPMs have limited memory and processing capabilities. This would make running of process heavy detection models inside a TPM a difficult proposition, and hence, the choice of software simulation tools is a preferred option for initial experiments. One of the drawbacks for the security in this framework is the existence of insider threat (Bishop and Gates 2008; Greitzer et al. 2008). An insider threat can defeat the presented solution and addressing this threat will be a part of future work.

Acknowledgments This research is supported in part by the National Science Foundation under Grant No. DGE –1754085. Usual disclaimers apply.

References

Baksi, R.P., & Upadhyaya, S.J. (2017). Kidemonas: The silent guardian. arXiv:1712.00841.

Baksi, R.P., & Upadhyaya, S.J. (2018). A comprehensive model for elucidating advanced persistent threats (apt). In *Proceedings of the International Conference on Security and Management (SAM)* (pp. 245–251): The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

Bencsáth, B., Pék, G., Buttyán, L., Felegyhazi, M. (2012). The cousins of stuxnet: duqu, flame, and gauss. *Future Internet*, *4*(4), 971–1003.

Bennett, J.T., Moran, N., Villeneuve, N. (2013). *Poison ivy: Assessing damage and extracting intelligence*. FireEye Threat Research Blog.

Bishop, M., & Gates, C. (2008). Defining the insider threat. In *Proceedings of the 4th annual workshop on Cyber security and information intelligence research: developing strategies to meet the cyber security and information intelligence challenges ahead* (pp. 1–3).

Bizga, A. (2020). Ransomware attack confirmed by australia-based beverage manufacturer. Security Boulevard. https://securityboulevard.com/2020/06/ransomware-attack-confirmed-by-australia-based-beverage-manufacturer/.

Ćeker, H., Zhuang, J., Upadhyaya, S., La, Q.D., Soong, B.H. (2016). Deception-based game theoretical approach to mitigate dos attacks. In *International conference on decision and game theory for security* (pp. 18–38): Springer.

Chen, M.Y., Kundu, A., Zhou, J. (1994). Off-line handwritten word recognition using a hidden markov model type stochastic network. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, *16*(5), 481–496.

Clark, Z. (2017). The worm that spreads wanacrypt0r. Malwarebytes Labs. https://blog.malwarebytes.com/threat-analysis/2017/05/the-worm-that-spreadswanacrypt0r/.

Correspondent, T. (2020). Fact check: Iit madras servers were under ransomware attack? Times of India. https://timesofindia.indiatimes.com/times-fact-check/news/fact-check-iit-madras-servers-were-under-ransomware-attack/articleshow/74319280.cms.

Costan, V., & Devadas, S. (2016). Intel sgx explained. *IACR Cryptology ePrint Archive*, *2016*(086), 1–118.

Daemen, J., & Rijmen, V. (1999). Aes proposal: Rijndael.

Daley, D., & Gani, J. (1999). Cambridge studies in mathematical biology Epidemic modelling: an introduction.

De, P., Liu, Y., Das, S.K. (2008). An epidemic theoretic framework for vulnerability analysis of broadcast protocols in wireless sensor networks. *IEEE Transactions on Mobile Computing*, *8*(3), 413–425.

Di Pietro, R., & Verde, N.V. (2011). 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* (pp. 1–6): IEEE.

Endgame (2017). Wcry/wannacry technical analysis. elastic. https://www.elastic.co/blog/wcrywanacry-ransomware-technical-analysis.

Faghani, M.R., & Nugyen, U.T. (2017). Modeling the propagation of trojan malware in online social networks. arXiv:1708.00969.

Falliere, N., Murchu, L.O., Chien, E. (2011). W32. stuxnet dossier. White paper, Symantec Corp. *Security Response*, *5*(6), 29.

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

Greenberg, A. (2017). Hackers are trying to reignite wannacry with nonstop botnet attacks. Wired Security. https://www.wired.com/2017/05/wannacry-ransomware-ddos-attack/.

Greitzer, F.L., Moore, A.P., Cappelli, D.M., Andrews, D.H., Carroll, L.A., Hull, T.D. (2008). Combating the insider cyber threat. *IEEE Security Privacy*, *6*(1), 61–64.

Hutchins, E.M., Cloppert, M.J., Amin, R.M. (2011). 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.

Jang, J., Choi, C., Lee, J., Kwak, N., Lee, S., Choi, Y., Kang, B.B. (2016). Privatezone: Providing a private execution environment using arm trustzone. *IEEE Transactions on Dependable and Secure Computing*, *15*(5), 797–810.

Kumar Sasidharan, S., & Thomas, C. (2018). A survey on metamorphic malware detection based on hidden markov model. In *2018 International conference on advances in computing, communications and informatics (ICACCI)* (pp. 357–362): IEEE.

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

Leonard, C. (2015). 2015 threat report. Websense Security Labs.

Ljolje, A., & Levinson, S.E. (1991). Development of an acoustic-phonetic hidden markov model for continuous speech recognition. *IEEE Transactions on signal processing*, *39*(1), 29–39.

LLC, P.I. (2013). *The state of advanced persistent threats*. Ponemon Institute Research Report.

LogRhythm (2013). *The apt lifecycle and its log trail*. Technical Report.

Lorch, J.R., Wang, Y.M., Verbowski, C., Wang, H.J., King, S. (2011). Isolation environment-based information access. US Patent 8,024,815.

Madnick, S.E., & Donovan, J.J. (1973). Application and analysis of the virtual machine approach to information system security and isolation. In *Proceedings of the Workshop on Virtual Computer Systems* (pp. 210–224). New York: ACM. https://doi.org/10.1145/800122.803961.

Mehresh, R., & Upadhyaya, S. (2012). A deception framework for survivability against next generation cyber attacks. In *Proceedings of the International Conference on Security and Management (SAM)* (p. 1): The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

Mahajan, P., & Sachdeva, A. (2013a). A study of encryption algorithms AES, DES and RSA for security Global Journal of Computer Science and Technology.

Mehresh, R. (2013b). *Schemes for surviving advanced persistent threats*. PhD Dissertation, Faculty of the Graduate School of the University at Buffalo State University of New York.

Meskauskas, T. (2020). How to uninstall wannacash ncov ransomware? PC Risk. https://www.pcrisk.com/removal-guides/17477-wannacash-ncov-ransomware.

Messaoud, B.I., Guennoun, K., Wahbi, M., Sadik, M. (2016). 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)* (pp. 1–6): IEEE.

Pauna, A. (2012). Improved self adaptive honeypots capable of detecting rootkit malware. In *2012 9Th international conference on communications (COMM)* (pp. 281–284): IEEE.

Piolle, E. (2008). Simplified schema of a trusted platform module (tpm). Wikipedia. https://commons.wikimedia.org/wiki/File:TPM.svg.

Rabiner, L., & Juang, B. (1986). An introduction to hidden markov models. *IEEE Assp Magazine 3*(1), 4–16.

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

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

Rashid, A., Ramdhany, R., Edwards, M., Kibirige Mukisa, S., Ali Babar, M., Hutchison, D., Chitchyan, R. (2014). Detecting and preventing data exfiltration.

Robbins, M. (2019). Cyberattack hits indian nuclear plant. Arms Control Association. https://www.armscontrol.org/act/2019-12/news/cyberattack-hits-indian-nuclear-plant.

Sanderson, G. Animated math. 3blue1brown https://www.3blue1brown.com/.

Secureworks (2017). Wcry ransomware campaign. Secureworks Inc. https://www.secureworks.com/blog/wcry-ransomware-campaign.

Shepherd, C., Arfaoui, G., Gurulian, I., Lee, R.P., Markantonakis, K., Akram, R.N., Sauveron, D., Conchon, E. (2016). 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* (pp. 168–177): IEEE.

Simhan, T.E.R. (2020). Iit-m's email servers shut down, raises malware concerns. Business Line. https://www.thehindubusinessline.com/info-tech/cyber-attack-shuts-iit-madras-email-system/article30861902.ece.

Spitzer, F. (2013). *Principles of random walk* Vol. 34. Berlin: Springer Science & Business Media.

TCG (2011). Tpm main specification. Trusted Computing Group. https://trustedcomputinggroup.org/tpm-main-specification/.

Vukalović, J., & Delija, D. (2015). Advanced persistent threats-detection and defense. In *2015 38Th international convention on information and communication technology, electronics and microelectronics (MIPRO)* (pp. 1324–1330): IEEE.

Zakaria, W.Z.A., Abdollah, M.F., Mohd, O., Ariffin, A.F.M. (2017). The rise of ransomware. In *Proceedings of the 2017 International Conference on Software and e-Business* (pp. 66–70): ACM.

Zhao, C., Saifuding, D., Tian, H., Zhang, Y., Xing, C. (2016). On the performance of intel sgx. In *2016 13Th web information systems and applications conference (WISA)* (pp. 184–187): IEEE.

Zyba, G., Voelker, G.M., Liljenstam, M., Méhes, A., Johansson, P. (2009). Defending mobile phones from proximity malware. In *IEEE INFOCOM 2009* (pp. 1503–1511): IEEE.

Rudra Prasad Baksi is a PhD candidate in the department of Computer Science and Engineering at University at Buffalo, SUNY, USA. He currently works on systems security with a focus on hardware-assisted security to counter advanced persistent threats (APT). Prior to joining UB, he completed his Masters degree at University of Florida, Gainesville, Florida, USA. His research interests span areas of systems security, hardware-assisted security, computer security, cryptography, and game-theory.

Shambhu J. Upadhyaya is a Professor of Computer Science and Engineering with the State University of New York at Buffalo, Buffalo, NY, USA, where he also directs the Center of Excellence in Information Systems Assurance Research and Education (CEISARE), designated by the National Security Agency and the Department of Homeland Security. Prior to July 1998, he was a faculty member with the Electrical and Computer Engineering Department. He has authored or coauthored more than 295 articles in refereed journals and conferences. His research has been supported by the National Science Foundation, U.S. Air Force Research Laboratory, the U.S. Air Force Office of Scientific Research, DARPA, and National Security Agency. His research interests include broad areas of information assurance, computer security, and fault-tolerant computing.