Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Design of Intrusion Detection Honeypot using Social Leopard Algorithm to detect IoT ransomware attacks

[1]Sibi Chakkaravarthy S, [2]D. Sangeetha, [2]Meenalosini Vimal Cruz, [2]V. Vaidehi, [3]Balasubramanian R

[1]VIT-AP University, Andhra Pradesh, India
[2]Madras Institute of Technology, Anna University, India.
[2]Keene State College, Keene, USNH, New Hampshire, USA
[2]Mother Teresa Women's University, Kodaikanal, India.
[3]Indian Institute of Technology, Roorkee, India.

Corresponding author: Sibi Chakkaravarthy S (e-mail: chakkaravarthy.sibi@vitap.ac.in).

ABSTRACT In recent times, ransomware has become the most significant cyber-attack targeting individuals, enterprises, healthcare industries, and the Internet of Things (IoT). Existing security systems like Intrusion Detection and Prevention System (IDPS) and Anti-virus (AV) as a single monitoring agent is complicated and time-consuming, thus fails in ransomware detection. A robust Intrusion Detection Honeypot (IDH) is proposed to address the issues mentioned above. IDH consists of i) Honeyfolder, ii) Audit Watch, and iii) Complex Event Processing (CEP). Honeyfolder is a decoy folder modeled using Social Leopard Algorithm (SoLA), especially for getting attacked and acting as an early warning system to alert the user during the suspicious file activities. AuditWatch is an Entropy module that verifies the entropy of the files and folders. CEP engine is used to aggregate data from different security systems to confirm the ransomware behavior, attack pattern, and promptly respond to them. The proposed IDH is experimentally tested in a secured testbed using more than 20 variants of recent ransomware of all types. The experimental result confirms that the proposed IDH significantly improves the ransomware detection time, rate, and accuracy compared with the existing state of the art ransomware detection model.

INDEX TERMS Complex Event Processing, CEP, Honeypot, Honeyfolder, SoLA, Intrusion Detection Honeypot, Ransomware

I. INTRODUCTION

IoT is a system of built-in-sensors interconnected to collect and transfer the data automatically without any human interventions. The sensors in the IoT devices interact with the internal and external state of environments and makes a decision autonomously. The tremendous era of IoT is the dawn of the digital world, which has seen massive inventions almost in every facet of our lives. The performance of today's IoT devices extends from managing the extreme amount of data to computing through synthetic intelligence. However, any disruption or malfunction of any devices in an IoT infrastructure may incur devastating threats to the process and trustworthiness. The ransomware attack is known to be quite effective in disrupting and paralyzing the IoT devices [1]. IoT Ransomware is malicious software that restricts access to the IoT devices until an amount of ransom is paid. There are two types of ransomware: Locker [24] and Crypto-ransomware [2][15]. Locker ransomware locks the entire system and denies access to the system, whereas the crypto-ransomware encrypts the user files and denies access to the user files in the system [25]. Locker is a common variant used in most IoT ransomware, locking the entire device and denying access. However, the ransomware attacks that occurred from the late of 2014 witnessed the CryptoLocker ransomware attacks [26], a hybrid variant which combines locker and crypto-ransomware [11]. In general, the IoT ransomware attack model is different from the generic computer and smartphone model. IoT ransomware attack is considered a severe threat because it is initiated against the target IoT device at a time and place where the devices will not be able to reset or counter the effects of the ransomware and will be more willing to pay the ransom. For example, hackers can lock up any thermostat in a smart home at a high temperature or hackers can lock smart car's control panel with no means to access or hackers can hack industrial IoT and lock smart power grids or hackers can hack the drug infusion pump and lock the drug blends to an irregular proportion, thus demanding a huge ransom to release the lock.

These IoT ransomwares are explicitly designed for IoT devices. IoT infrastructure is entirely dependent on other embedded systems such as Wi-Fi Access Points (AP), external adapters, dongles. However, there is zero assurance that these devices are fully secured and updated [11]. According to the report given by Forbes [22], most of the IoT gadgets are still functional with out-of-date firmware. Deployment of such devices possess an inherent risk to the enterprises and leads the attacker to find the different attack surface and vulnerabilities. Moreover, a determined attacker can easily guess how to penetrate the attack surface and re-purpose the devices [10].

Malware writers use many social engineering attacks to compromise targets [7]. There are many ways through which the ransomware enters into a system. For instance, i) malicious advertisements lure the users into the hijacked or compromised websites which, when visited leads to the ransomware infection, ii) Spear Phishing emails can be used for invoking ransomware; the mail comes along with the malicious link or attachments containing the payloads for affecting the system. Hence, most attackers use social engineering attacks such as Spear Phishing emails to attract the victim by sending fake news or messages that appear to come from a legal source (company or person) [7]. Such emails are designed to have malicious links that, when clicked, download the ransomware. Further, the malicious links are posted on social media, legal websites, and when the users view the particular web page, the malicious attachments embedded in the link get downloaded. Ransomware downloaded through such links will break into the system by exploiting the vulnerabilities and start the encryption process. iii) Downloaders such as drive-by-download, water holing, malvertising, removable media, and botnets are also used to propagate the ransomware. Malware developers are frequently finding their attack vector to infect the victim. There are various ways in which the attackers take control of the user systems and encrypt all files. The attacker mainly exploits the human weakness to take down the system [12].

Based on the facts mentioned above, this paper focuses on the cryptographic techniques used by contemporary ransomwares, and the experimental analysis confirms that most of the ransomwares are utilizing asymmetric cryptographic techniques [10]. This cryptographic technique utilizes two different keys for encryption and decryption. Hence, the ransomware utilizing asymmetric schemes residing on a victim's machine needs an external connection with the C&C server to register for the keys to start the encryption process. Blocking the communication process can make the ransomware inactive due to the missing keys, and without the valid keys, the encryption process cannot be initiated. Taking this into account, deploying Software Defined Networking (SDN) infrastructure can provide an additional security measure to block the ransomware communication by identifying the network traffic flow and applying the control rules in real-time to block such suspicious traffic.

SDN's main functionality was with a Layer 2 switch, which forwards all the DNS traffic to the controller. The flow was initially crafted and added so that the controller starts monitoring from the first packet in the flow. This feature results in the inspection, evaluation, and comparison of all the DNS queries (request) with the blacklisted and whitelisted DNS database used by the ransomware command and control (C&C). Any query that matches the blacklisted DNS leads to the rejection of the response, and the response never reaches the infected host. Thus, the encryption will not be started. Furthermore, the flow is also verified with the whitelisted DNS for further processing. If the domain is not listed in both the Blacklist and whitelist, no actions are performed. However, this is an exceptional case, and it is not near possible due to the frequent DNS update [14].

Further, a Honeypot agent (Honeyfolder) modeled using SoLA is installed as a monitoring agent in all the hosts in the network to monitor all the user activities and file system activities to detect ransomware.

The main objective of this paper is to propose a novel IDH using CEP, which collects an enormous amount of data from various sources such as Honeyfolder, SDN network and hosts, Audit Watch, Firewall (IP Tables). Then convert the data into event instance and process the events in the CEP engine by applying aggregation rules to determine the malware behavior, attack pattern, and respond promptly. In general, distributed security systems in enterprises produce a massive amount of data. Predicting abnormal events from data streams is very complex and requires enormous processing power. In order to analyze, predict, and detect the abnormal events from the complex data streams, CEP is used. Previous work (Ranjan et al. [8]) on CEP based Hybrid IDS discusses how CEP based IDS can be used for real-time intrusion detection. However, issues like malware propagation, privilege escalation, ransomware post attacks are not addressed in the previous work. In this paper, a complete solution to handle multiple event streams from different scenarios such as user activities, ransomware activities, user privilege escalation, suspicious internal activities is modeled using event expressions and validated in real-time. This paper proposes a robust IDH for ransomware detection and mitigation with the following contributions.

- Implementation of Honeyfolder using SoLA to detect ransomware activities with no loss of user data in post-attack conditions.
- SoLA is the first algorithm that utilizes the behavior of a leopard.
- Implementation of an efficient rule engine for processing data into event streams and aggregating the events to determine ransomware behavior, attack patterns, and decision makings on time.
- Deploying SDN infrastructure for providing an additional security measure to block the

ransomware communication by identifying the suspicious network traffic flow.

- Design and implementation of multi-security systems (IDS (Snort), Firewall (Pfsense), Honeypot (Dionaea), Honeyfolder, Audit watch, SDN Controller)), and effectively utilizing CEP engine to make decisions.
- Consideration of multiple data sources (Honeyfolder, AuditWatch, SDN controller, firewall) to detect and distinguish the ransomware activities from user activities.

The organization of this paper is as follows: Section 2 details the background information about Honeypot, CEP, Existing Social behavior algorithms. Section 3 explains the social behavior of the leopard, which is modeled and utilized in this paper. Section 4 presents the proposed Intrusion Detection Honeypot. In Section 5, the experimental setup, configuration, and data set collection are explained. Section 6 exhibits the experimental results and analyze the performance of the proposed IDH with the state of the art Honeypot models. Finally, the paper is concluded in section 7.

II. Literature Survey

This section gives a clear insight into the state of the art technologies such as Honeypot and CEP along with their essential background. Further, this paper gives a brief discussion on the nature-inspired computing and Lion Optimization algorithm.

A. Honeypot

Honeypot is a mechanism to capture the attack strategy used by the attacker. It does not prevent or mitigate the attack. The honey pot's job is to remain silent and pretend itself as a real environment and trap the attacker. Honeypots are deployed in such a way that the attackers consider it a productive system and attack it. It collects information about the attacker by their movements and giving the details required by the attacker. The sole purpose of Honeypot is to collect information about the attack and attackers rather than preventing it. There are two types of Honeypots, research, and production honey pots. Production Honeypot is used to collect information about the attacker and mitigate the organization's risks, whereas the research Honeypots are used to gather information as much as possible. This information is exploited to understand the vulnerabilities in the existing environment and build a better defense system. Any data to and from Honeypot is suspicious; the reason is the Honeypot has no productive resources, i.e., if the honey pot is accessed, it is considered unauthorized access. The information from Honeypot is of high value, which can be used to develop a high-security system.

Eliot et al. [3] discussed Honeypot's real-time deployment in an internal organization's network. In their work, they presented the various types of Honeypots and their advantages. They claimed that the Honeypot deployment behind a firewall could be more successful in the production environment. The installation of a Honeypot in an external peripheral mode to defend against the advanced persistent threats proposed by Tian et al. [4].

Fan et al. [5] proposed a novel honeypot architecture called HoneyDOC to support all-round honeypot design and implementation. HoneyDOC consists of collaborative modules such as decoy, captor, and orchestrator. Shi et al. [6] proposed a blockchain-based Honeypot, which uses a dynamic mechanism of combining the functionalities of Low and High interaction Honeypot. Their work consists of a real system in time architecture that is widely used to capture real-time attacks. Further, their Honeypot possesses dynamic property, which shows up the present location and other services run in the Honeypot. Each port-level access is logged using blockchain, and independent analysis on various attacks was also carried out. However, their work focuses only on the network type attacks, and they do not address the real problem of malware-based attacks. To address the above-discussed issues, an efficient Honeypot with IDS is desirable. This paper proposes a robust IDH, and the results confirm that the proposed IDH is effective in terms of accuracy in detection and attracting the attacker with less false positives.

B. Nature Inspired Computing

The problem statements tackled by engineering solutions involve many moving variables. The goal of the engineer solutions is to optimize the results by loosely generalizing a few of such variables and appropriately apply heuristics to make the solution simpler in the approach to achieve high precision and efficiency. A standard way of doing it is to break it down into subproblems and arrive at the required solution from a bottom-up approach. This approach requires a great deal of experience in multidisciplinary techniques that can otherwise be complex and optimize the way the solutions are compiled. For instance, if we are building a car, the job is not just to build something that moves but also to facilitate other criteria such as safety, comfort, optimal space, ease of usage, which involves several trade-offs that need to be optimized depending on the design and the requirements that need to be met, which in itself can be quite complicated. An important point to note here is that no design is perfect, and optimization is a general ongoing process which is out in motion anytime a new technology or sub-branch of science is learned and attempted to be put to use.

Many engineering applications have optimization problems that are complex to solve. These problems have exponentially growing search space that increases with time. Therefore, the meta-heuristic algorithms must be used rather than the traditional optimization methods to solve these problems. The meta-heuristic algorithms have excellent performance in recognizing patterns, tuning neural networks, clustering, processing image and videos, scheduling, etc.

Nature has been the greatest inspiration for humans to solve many complex engineering problems. The life and behavior of various animals have been mapped to solve specific problems. For example, the Genetic algorithm was inspired by nature's evolution process. Particle Swarm Optimization (PSO) algorithm was inspired by the social behavior of birds searching for food. Ant colony optimization (ACO) was a metaheuristic algorithm inspired by the foraging behavior of ants. Marriage in Honey Bee Optimization (MBO) imitates the reproduction process in the honey bee colony. The monkey Search algorithm was inspired by the monkey in search of food resources, the behavior of cats inspired the Cat Swarm algorithm, and The Cuckoo Search algorithm was developed based on the reproduction process of Cuckoos. The Water flow-like algorithm mimics the water flowing process. Dolphin Partner Optimization was inspired by the behavior of dolphins. The firefly algorithm imitates the social behavior of fireflies. Bacterial Foraging algorithm was inspired by the foraging of bacteria. The Fish School Search simulates the clubbable behavior of oceanic fish. The bat-inspired algorithm was developed from the echolocation behavior of bats. The pollination characteristics of flowering plants inspired the flower pollination algorithm. Wolf Search algorithm mimics the behavior of Wolves. The Social Spider Optimization algorithm was developed from the social behavior of Spiders. Forest Optimization Algorithm was inspired by the life span of different kinds of trees in the forest. The Lion Optimization algorithm was developed from the social organization and behavior of Lions [9].

C.Lion Optimization Algorithm (LOA)

The LOA [9] is based on the social behavior and organization of the lions that exhibit high cooperation levels. Lions have two types of social organization, namely, residents and nomads. Residents live in groups, whereas the nomads are independent. The lion optimization algorithm's essential operation is the territorial defense and territorial takeover, which is used to find the worst solution by using the new solutions. The existing bio-inspired optimization algorithms incur more computational time, which is not suitable for malware detection since the execution time plays a significant role in malware detection. From the literature review, it is observed that the existing ACO, LOA, throws uncertainty in its convergence time. Hence, there is a need for an innovative and standardized algorithm to obtain the best solutions with less computational time.

Therefore, this paper proposes a new bio-inspired algorithm (SoLA) which provides the best solution in less time. The most important operations of SoLA are territorial defense and territorial take over where the new best solutions replace the old solutions. Also, it has a 1:N ratio, which helps to find the malicious process easily since the malicious process always occupies the first position in the territory. Whereas in other algorithms, a separate method should be devised to do the same.

D. Complex Event Processing (CEP)

CEP is used for processing the event data from multiple sources and helps in discovering complex events by analyzing and correlating the other events. The goal of CEP is to identify meaningful events and respond to them as quickly as possible [12]. CEP provides highly expressive rules using first-order logic, having the features of representing the temporal relations. The CEP is a scalable, flexible technique for constructing refined data, and it aims at data correlation to identify pre-defined patterns. It also contains tools for processing continuous and timely events from different sources and extracts high-level knowledge events from lower-level events, which are referred to as complex events. CEP has an inference engine that takes decisions based on a rule base and working memory. Rule base contains all the rules, and working memory keeps records about the recent states in the system. Whenever a feature extracted packet arrives, it goes to the inference engine for its identification; if it is detected as an attack, the administrator would be notified at the earliest.

CEP is used to convert all pre-existing knowledge about ransomware and their behavior within a domain to generic rule based on blacklists/white lists, moving averages, known behavioral patterns, outliers, and transition events. CEP is capable of inspecting multi-gigabit traffic because of the Distributed CEP engine (DCEP)), which uses computing intelligence and resources to meet changing demand for event processing. DCEP integrates the event processing network with more number of nodes, which reduces the load and makes the whole system more robust and reliable. A detailed explanation of the CEP and its related engines are given in the paper [12]. However, there is no existing mechanism found in the literature which utilizes the CEP engine for Honeypot.

From the literature review, it is found that detecting and identifying ransomware attacks in post-attack conditions are trivial and risky. Traditional analysis techniques like sandboxes, AVs, application firewalls concentrate on techniques like static and dynamic analysis, which are unsuccessful in distinguishing the ransomware activities from user activities. From the recent studies, it has been witnessed that the SDN deployment strategies can improvise the ransomware mitigation by blocking the communication between the infected host and C&C servers. However, the contemporary ransomware incorporates sophisticated malware intelligence in their design to evade the perimeter defenses. It is also found that the single source of features is not capable of detecting ransomware. This motivation leads to the development of a robust security system to mitigate ransomware. Hence this paper proposes a robust IDH. CEP has a substantial impact on information systems designed with decentralized data, which is being used in many places, including finance, manufacturing, etc. The large-scale real-time stream processing has become more critical and demanding with the evolution of technologies such as big data and IoT. CEP allows faster processing and aggregation of data from different security systems, which are the essential attributes in identifying real-time suspicious processes, making the overall detection of suspicious activity simpler and efficient and supporting complex operations like time windows and temporal query patterns.

III.SOCIAL BEHAVIOUR OF LEOPARD

Leopards are solitary and predominantly nocturnal in nature among the socially inclined wild cat species. Leopards have a strong dimorphism with two types of social organization: i) Residents ii) Migrants. Resident males are dominant males who live within a territory, along with very few females. A male leopard territory can have overlapping territories of 2 or 3 female leopards. Generally, the territory is marked using urinal scents, defecation, vocalization, and scratching trees or small plants. The size of the territory range depends on the prey availability and habitat. Migrant males are roaming males who are independent adult cubs or weak resident males defeated by other migrants. Pairs are seen only during reproductions. Unlike other wild cats, leopards are opportunistic hunters who are solitary in nature. The most important characteristics of leopards are territorial defense and take over. The migrant leopard fights with the resident males and takes over the territory if it wins. When applied to the Ransomware mitigation system, these characteristics provide better solutions by replacing the old solutions. SoLA is more comfortable to implement, and the solution is definitely obtained.

1. Initialization

In the proposed SoLA, the leopards are initial solutions generated randomly, which are either malicious processes (male leopard) or user processes (female leopard). The leopards are partitioned into residents and migrants in such a way that half of the leopards are residents, and the other half are migrants. The residents are represented as territories that have one malicious process and n user processes. Any leopards outside the territory are migrants, and it may include both male and female leopards.

2. Fitness calculation

The fitness of all the resident and migrant leopards is calculated. Each process has a list of features, as shown in Table 1. The process features are extracted, and the score is calculated based on the thresholds. Each feature is compared with the particular threshold, and if it is greater than the threshold, a score is generated. Adaptive thresholds and scores are used for this purpose.

3. Territorial Defence

The fitness of the male on territory and other females is compared. If the male is found to have lower fitness than females, then the male is removed from the territory, and it is made as a migrant.

4. Territorial takeover

The fitness of the migrants and the resident males are compared, and if the male has lower fitness than a migrant, then the migrant is made as the dominant in the territory, and the resident male removed from the territory.

5. Convergence

The algorithm is made to run for 20secs continuously, and the results are provided to the CEP engine to make the final decision about the process.

Table 1. **Process Features**

S. No	Features
1	CPU consumption
2	Memory Consumption
3	Read bytes
4	Write bytes
5	Active time of the process
6	Owner of the process
7	File renamed

IV. Intrusion Detection Honeypot (IDH)

This section presents the design of the proposed IDH.

A. System Architecture

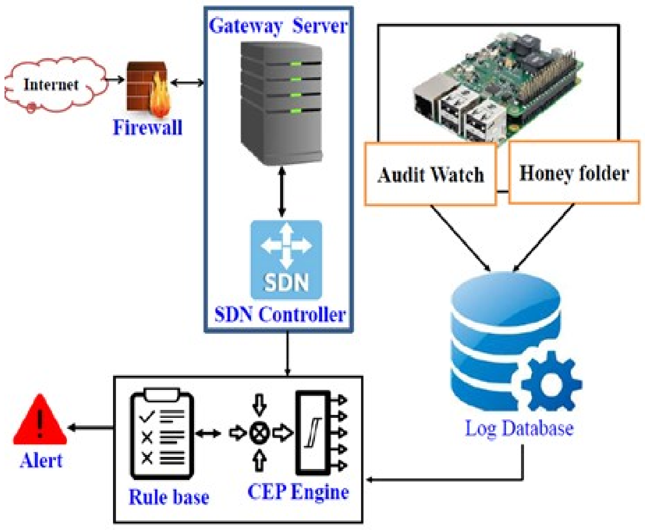


Figure 1. **The proposed IDH Architecture**

Figure 1 presents the architecture of the proposed IDH. The proposed IDH has a default Low interaction Honeypot server and a Network IDS deployed at the Chokepoint on the network. The Honeypot server and IDS captures all the traffic and analyze them. In case of any direct attacks (malicious links, website redirect), the IDS and Honeypot detect and report the behavior of the attack to the firewall. CEP engine correlates various events from hosts and networks, Honeypot agents, SDN controller, Audit Watch, logs, etc. Based on the CEP result and scores, the malicious process is identified and killed.

Present-day malware writers are continually improving the evasion scheme in their malware to bypass the existing defenses. Recent ransomware attacks reveal that these malwares compromise all the internet-connected devices, including desktops, smartphones, and IoT devices. In order to address these issues, a post-attack detection model using Honeypot has been proposed. The key idea of the IDH is to create decoy folders with the dummy files in each partition of the disk in such a way that it meets the requirements of the ransomware to encrypt so that the attacker first tries to encrypt the decoy folder. The Decoy folder is randomly generated with the random files of types (documents, media, files) and allowed for the initial compromise in case of the ransomware attacks. Each host system has a Honeypot agent installed and running on them. The Honeypot agents are designed based on SoLA.

In general, the ransomware encryption process includes reading the file, encrypting the file, writing the file, and deleting the encrypted file. Hence the encryption process carried out by the ransomware is modeled as the states $S=\{S_1..S_4\}$, namely read, encrypt, write and delete. When there is a transition from one state to another in the decoy folder as an IO activity, it checks whether it is user activity or ransomware activity. When a file is read and encrypted, it marks it as suspicious activity and checks specific parameters like IO frequency (how frequent the file is read and written, how frequent it is renamed), and scores are calculated based on the Jaccard similarity measure [13]. Based on the score, the decision is made (user or ransomware activity). The result is sent to the CEP engine and firewall. Besides, the host systems are configured and connected to the real world using an SDN network. The SDN network is advantageous to the traditional network because the SDN allows users to manage and control the network programmatically.

B. Honey agent (Honeyfolder)

The Honey agent is modeled using SoLA and installed on each host and performs continuous host monitoring. The Honeypot agent is managed by the Honeypot server using the CEP engine. The four states of SoLA are modeled as Read, Encrypt, Write, and Delete. The possible state transitions are, Read-Encrypt, Read-Write, Read-Delete, Write-Delete, Encrypt-Delete, Encrypt-Write, Write-Delete. The honey agent continuously monitors the decoy folders, and whenever there is a transition from one state to another in SoLA, it checks whether the action is user activity or ransomware activity. Parameters like Read frequency, Write frequency are considered, and a score is generated. If the score crosses a particular threshold, the process is identified and alerted to the firewall to kill that process.

1. Social Leopard Algorithm (SoLA)

Initially, the solutions are randomly divided into Partitions and Migrants. Each partition has one malicious and N user processes. The fitness of each process is calculated and compared with all other processes in the partition and the migrants. Adaptive scoring is used to calculate the fitness, where the thresholds are adaptively changed on each iteration based on the system's state. Calculated fitness will help identify the malicious processes that will be sent to the CEP engine for final processing.

2. Need for Adaptive Thresholds and Scores

Since the SoLA algorithm is designed to run on each system in the network, the Static threshold and scoring mechanism may not work because each system may have varying specifications. For example, the CPU consumed in a single-core processor is higher than that of a multicore processor since it uses the parallel execution of threads in different cores. Similarly, in the static scoring mechanism, the score will be calculated repeatedly for the same feature again and again, which will result in false positives, which leads to the Adaptive Thresholds and scoring mechanism.

3. Adaptive threshold

Each processor may have many threads running in different logical processors. The CPU consumed by each thread is extracted, and the total CPU used by the process is calculated by adding all the values. Similarly, the CPU that is currently used by the system is calculated by adding the CPU consumption of all active processes. The threshold is calculated using the formula,

(Overall CPU used by single process / Total CPU used in the system) * 100

The threshold is calculated for every iteration by setting the max value as the primary threshold in each iteration.

4. Adaptive Scoring

Initially, the score is distributed equally for all the features (1/ total no of features). On every deduction of a malicious process, the probabilities of the features responsible for the same are increased. The maximum probability of a feature is limited to 30%, and the minimum probability of a feature is limited to 5%. Hence the disadvantage of focusing on a single feature is eliminated.

5. How SoLA Works?

Once the Ransomware Mitigation System (RMS) identifies the access in the decoy folders, it extracts the features of the process that accessed the folder. The features include the CPU consumption, Memory consumption, Bytes read, Bytes written, Attribute change, Owner of the process, etc. The features are extracted and stored in the array named ip. The ip array and the size N of the array are given as the input to the SoLA algorithm.

```
Input: (ip, N)
Output: List of PID
Begin
    Generate initial solutions L of length N
    Split N/2 samples as P and others as $M_i$
    Each P has M and U of ratio 1: N
    For i = 1 to $P_n$
        Compute $F_M = fit(x_i)$
    End
    For i = 1 to $M_{i_n}$
        Compute $F_{m_i} = fit(x_i)$
    End
    While true
        For i = 1 to $P_n$
            If $F_{v_i} > F_M$
                Swap $F_M$ and $F_{m_i}$
                Swap M and $M_i$
                $M \leftarrow U$
            End if
        End
        For i = 1 to $M_{i_n}$
            If $F_{m_i} > F_M$
                $M \leftarrow M_i$
            End if
            If $F_m < T_h$
                $M_i \leftarrow M$
            End if
        End
    End
End
```

Figure 2. **Social Leopard algorithm (SoLA)**

The SoLA algorithm (Figure 2) splits the input array to Partition P and Migrant Mi arrays. The index of the element is stored in the P and Mi arrays. The first half of the input array forms the Partition array, and the second half forms the Migrant array. The structure of P is a matrix, with each row representing each territory. The first element of each row represents the malicious process (male leopard), and the other elements represent the user processes. (Female leopards). Each row has the malicious and user process in the ratio of 1: N.

The fitness of each process in the P and M arrays are calculated. The Adaptive scoring algorithm calculates the score for each process based on the Adaptive threshold, which will be later discussed in the chapter. Once the score is generated for each process, it is compared with those of other processes. The fitness F of user process U is compared with that of malicious process M. If the fitness of U is less than that of M, then the user process is made malicious, and the malicious process is made as to the U. Similarly, all the processes in all territories are compared.

The fitness of Migrant process M is compared with the user Process U. If the U < M_i, the migrant is made as to the User process, and it is placed inside the territory. The user process is removed from the territory and made as a migrant. The user process with low fitness is also removed from the territory. The same process would be continued for more iteration. The convergence of the SoLA algorithm is 20secs. The SoLA algorithm marks the malicious process and sends it to the CEP engine for every 20secs.

Adaptive Threshold

For the feature extraction process, the RMS extracts the features and stores them in the array. For example, the list of all threads of a process is stored in the array A_{tid}. The processor information and CPU usage per thread are stored in A_P and A_{cpt} arrays, respectively. The overall CPU O_{cpu} used by the process is calculated by summing all values in the array A_{cpt}. Similarly, the T_{cpu} (CPU currently used by the system) is calculated. The threshold is calculated using the formula,

$$(O_{cpu} / T_{cpu}) * 100$$

Where O_{cpu} is the overall CPU used by the single process, and Tcpu is the overall CPU used by all active processes. Similarly, for memory usage, the threshold is calculated using,

$$(O_{mem} / T_{mem}) * 100$$

Where O_{mem} is the overall memory used by the single process, and Tmem is the overall memory used by all active processes.

The maximum threshold in each iteration is fixed as the CT. The extracted CPU is checked, whether it is in the range of CT - α and CT + α. Where α = 10 (constant). Since the auto-tuning feature is not available in the proposed IDH, we set the alpha value to 10, which is the highest tuned and hardened value. Similarly, the same process is used to calculate the memory threshold of MT. The thresholds vary in every iteration based on the system's state.

C. *Complex Event Processing for IDH*

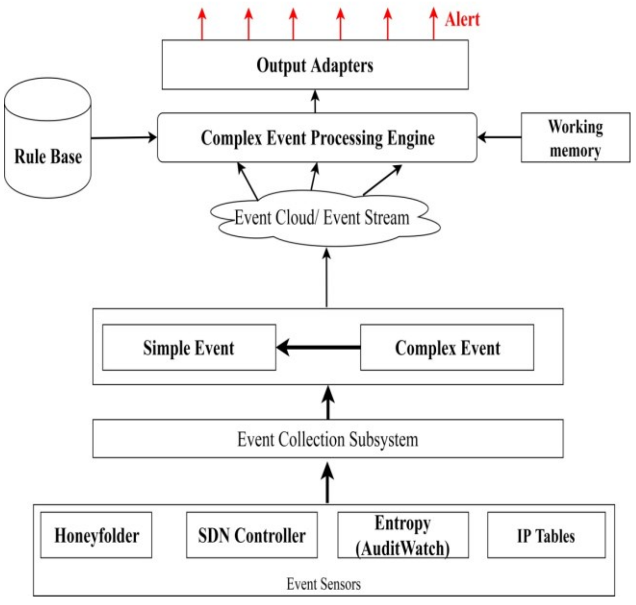


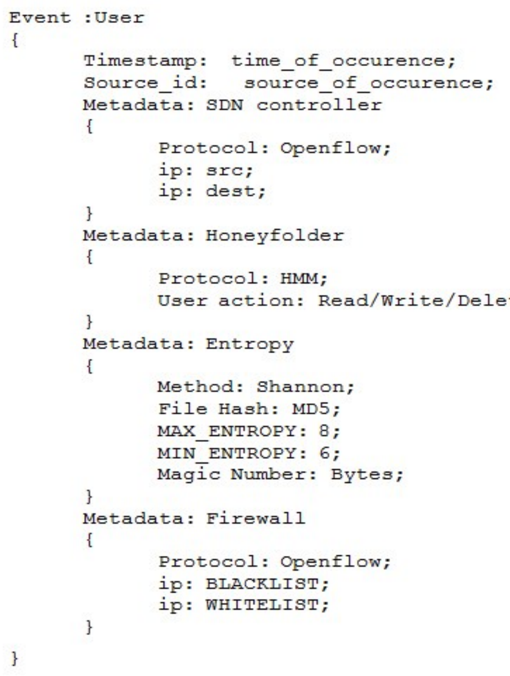
Figure 3. **CEP for IDH**

Figure 3 shows the flow of events generated by different systems. The events are aggregated to form a convoluted event that signifies a pattern or behavior of interest. The events considered in this paper are listed as follows.

- State transition when there is an action in the Honey folder.
- Entropy (Audit Watch) estimation indicating the File/Folder changes.
- Timestamps of all the event occurrences.
- Network features (UNSW-NB15 dataset [21]) captured by the SDN controller.
- Host features captured by the Honey agent.
- Rules fired by the firewall (pfsense).
- Events generated by the other security systems.
- All the events and actions performed by the Honeyfolder, Firewall, Audit Watch, SDN controller, etc.

The events are processed by an Event processing model, which is adapted from the previous work of Author 4 [12].

Event Collection (Level 0): Events from various sources such as Honeyfolder, Firewall, Audit Watch, SDN controller along with timestamps arrive as a stream to the CEP engine. Event Collection subsystem reads the input from different sources and the events.



```
Event :User
{
        Timestamp:  time_of_occurence;
        Source_id:   source_of_occurence;
        Metadata: SDN controller
        {
                Protocol: Openflow;
                ip: src;
                ip: dest;
        }
        Metadata: Honeyfolder
        {
                Protocol: HMM;
                User action: Read/Write/Delete;
        }
        Metadata: Entropy
        {
                Method: Shannon;
                File Hash: MD5;
                MAX_ENTROPY: 8;
                MIN_ENTROPY: 6;
                Magic Number: Bytes;
        }
        Metadata: Firewall
        {
                Protocol: Openflow;
                ip: BLACKLIST;
                ip: WHITELIST;
        }
}
```

Figure 4. **General Event Structure**

Event Refinement and Pre-processing (Level 1): It is the first step in CEP where events from various sources are checked for missing values, noise, and other discrepancies and filtered. Primary feature extraction, normalization, and selection are performed in this stage. Events from different sources are transformed into a compatible format to be processed by CEP Engine. A general event structure is given in Figure 4.

Situation Refinement (Level 2): In this stage, the event is classified as normal or abnormal events based on the range of values for each event. In IDH, it refers to the aggregation of values such as missing net flow in the controller flow table, blocked processes, diverse entropy range, firing firewall rules, etc.

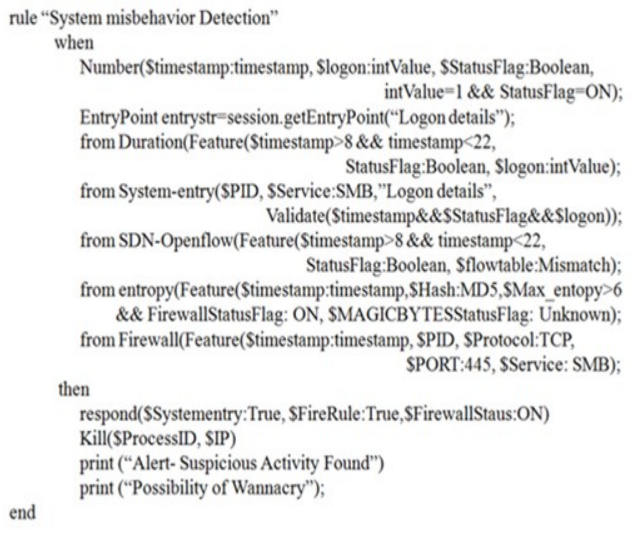
Threat assessment (level 3): This stage correlates the event from level 1 and level 2 using logical, spatial, and temporal constructs to form a complex event that models the ransomware attack, thereby differentiating the user and ransomware activities.

Process Refinement: Process refinement step is carried out to tune and update the decision variable to improvise the system's performance. A sample complex event scenario is modeled using event expression and discussed below.

Example 1: System misbehavior detection[1]

When a legitimate user's system misbehaves and attempts the following actions.

1. Software trying to perform communication with other servers (Remote, TOR) located in different locations.
2. Software accessing unauthorized system via network (SMB).
3. Software sending bulk mails/ copying/ deleting bulk data.
4. Software trying to escalate the privilege.
5. Software trying to encrypt the files.



```
rule "System misbehavior Detection"
    when
        Number($timestamp:timestamp, $logon:intValue, $StatusFlag:Boolean,
                                        intValue=1 && StatusFlag=ON);
        EntryPoint entrystr=session.getEntryPoint("Logon details");
        from Duration(Feature($timestamp>8 && timestamp<22,
                                StatusFlag:Boolean, $logon:intValue);
        from System-entry($PID, $Service:SMB,"Logon details",
                            Validate($timestamp&&$StatusFlag&&$logon));
        from SDN-Openflow(Feature($timestamp>8 && timestamp<22,
                            StatusFlag:Boolean, $flowtable:Mismatch);
        from entropy(Feature($timestamp:timestamp,$Hash:MD5,$Max_entopy>6
             && FirewallStatusFlag: ON, $MAGICBYTESStatusFlag: Unknown);
        from Firewall(Feature($timestamp:timestamp, $PID, $Protocol:TCP,
                                        $PORT:445, $Service: SMB);
    then
        respond($Systementry:True, $FireRule:True,$FirewallStaus:ON)
        Kill($ProcessID, $IP)
        print ("Alert- Suspicious Activity Found")
        print ("Possibility of Wannacry");
end
```

Figure 5. **Sample Event expression model (Wannacry)**

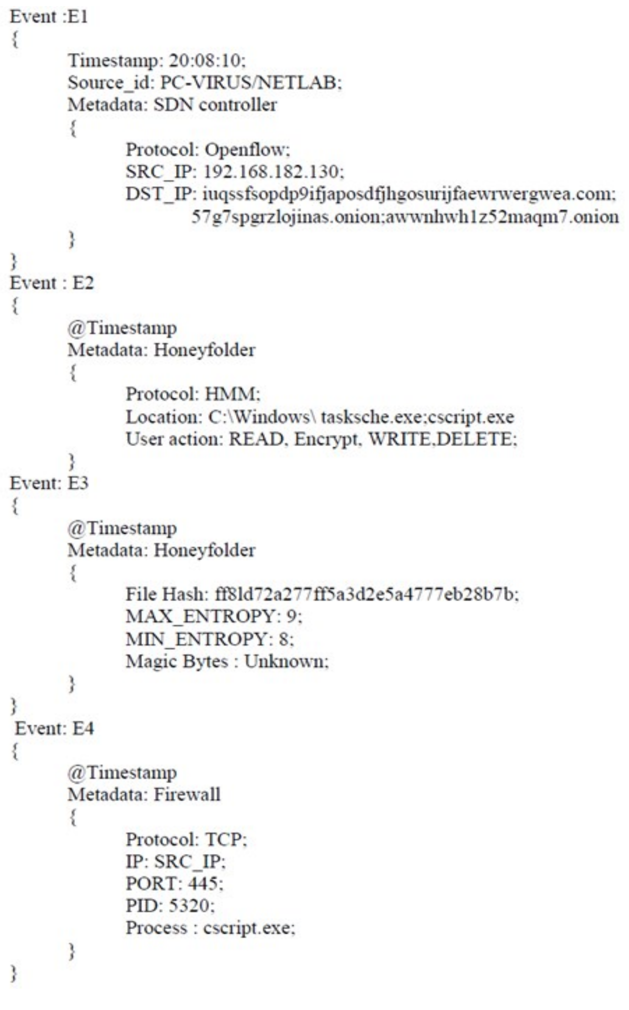
The intricate pattern formulated for Figure 5 is given below.

During ((coincides (E1, E4) ^ coincides (E2, E3)), timestamp)

The rule for the above-mentioned intricate pattern is given in Figure 6.

D. Ransomware detection using IDH

Consider the scenario of CryptoLocker ransomware. The various processes involved in CryptoLocker are given as follows. **Infection**: Assume that a CryptoLocker compromises the victim through a spear-phishing email or an exploit kit. The exploit kit targets the unpatched security vulnerabilities of the software running on the victim device. The angular exploit kit is a well-known method for the CryptoLocker to gain execution. Angular exploit kit focuses more on the vulnerabilities found in network services and authentication. **Behavior & Implants**: Upon the successful execution, CryptoLocker implants their variants and maintains consistent persistence in their behavior. **Delivery**: The malware executables are delivered through separate encrypted channels in order to bypass the traditional defenses.



```
Event :E1
{
        Timestamp: 20:08:10;
        Source_id: PC-VIRUS/NETLAB;
        Metadata: SDN controller
        {
                Protocol: Openflow;
                SRC_IP: 192.168.182.130;
                DST_IP: iuqssfsopdp9ifjaposdfjhgosurijfaewrwergwea.com;
                        57g7spgrzlojinas.onion;awwnhwh1z52maqm7.onion
        }
}
Event : E2
{
        @Timestamp
        Metadata: Honeyfolder
        {
                Protocol: HMM;
                Location: C:\Windows\ tasksche.exe;cscript.exe
                User action: READ, Encrypt, WRITE,DELETE;
        }
Event: E3
{
        @Timestamp
        Metadata: Honeyfolder
        {
                File Hash: ff8ld72a277ff5a3d2e5a4777eb28b7b;
                MAX_ENTROPY: 9;
                MIN_ENTROPY: 8;
                Magic Bytes : Unknown;
        }
}
Event: E4
{
        @Timestamp
        Metadata: Firewall
        {
                Protocol: TCP;
                IP: SRC_IP;
                PORT: 445;
                PID: 5320;
                Process : cscript.exe;
        }
}
```

Figure 6. **Sample CEP rule for detecting Wannacry**

Note: [1]Likewise, any number of CEP rules can be formed for any variants.

Files & Folders: Most of the ransomwares place their executables in the temporary file folders (AppData/Roaming, Temp**. Left off point:** Most of the malware adds where it left off points to continue their encryption process from the left off point if the infected device is rebooted. **Encryption:** Current CryptoLocker uses a secure encryption cipher such as AES 256. In order to initiate the encryption process, most of the malware communicates with C2 servers requesting the key. **System tags:** If the infection is high, malware tags the infected system using a unique identifier so that C2 servers can differentiate the different victims. Finally, the CryptoLocker encrypts all the files using the unique extensions.

Consider the above-discussed scenario to detect and mitigate the CryptoLocker. As discussed earlier, the proposed IDH consists of Honeyfolder, AuditWatch, and CEP engine as a core module. According to the above-discussed example scenario, the victim mistakenly opens the spear-phishing email and gets infected by the CryptoLocker. All the File systems used for experimentation are single partitioned and configured with Honeyfolder by some random files (likely to attract but contain bogus files such as backup archives, documents, videos, photos). From the experimental analysis, it is found that some of the ransomwares use asymmetric cryptography in order to make the threat harder to beat. Hence pre-configured SDN infrastructure helps to filter out the suspicious traffic and disrupt the communication between the CryptoLocker and C2 server without starting the encryption process. If the ransomware uses symmetric cryptography, the key is hardcoded within the malware, which allows itself to initiate the encryption process. In this case, Honeyfolder lures the CryptoLocker to perform its action in the Honeyfolder. When the CryptoLocker starts to encrypt the files, the Honeyfolder sends an alert regarding the suspicious process and halts the process. Furthermore, to verify the process, the AuditWatch calculates the entropy value [23] for encrypted folders and sends the entropy value to the CEP engine. CEP engine correlates the values from the Honeyfolder, AuditWatch, and SDN application and generates an alert based on an advanced set of rules that provides a high degree of output accuracy. Thus, the proposed IDH successfully halts the ransomware process with minimal loss.

V. Experimental Setup

A scenario of a small-scale network with 4 Raspberry Pi (Pi 4 with 4 GB RAM running in Raspbian OS) and a server is considered for the experimentation (proof-of-concept). The clients' infrastructure is networked using Edge core 4610-30T SDN switch running in the Open Network Operating System (ONOS), and the POX controller is utilized. Open flow protocol is used to control and monitor the traffic from the clients. Besides, the packet flow in the network is compared to the flow table. Based on the results, the necessary action, such as flow entry matching or forwarding to the controller is performed. The entire experimental setup, as given in Figure 7 is protected by the Snort IDS, Dionaea Honeypot, and a software-based firewall (pfsense).

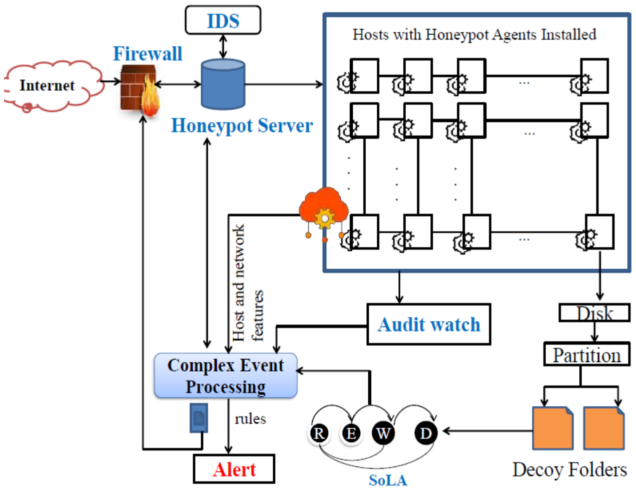


Figure 7. **Experimental setup and Deployment architecture of IDH**

A. Server Configuration

The proposed IDH is experimentally tested in a secured testbed using a dedicated server with Software Guard Extension (SGX) enabled Intel Xeon (6th Gen) processor, 28 GB RAM, 2 TB Hard drive running on Ubuntu Server. The configuration includes Dynamic monitoring and behavior Analysis (DBA), which involves the runtime monitoring of the malware execution. All the network interfaces are interconnected, and the network flow is verified by the SDN controller (adapted from the [14] [20]). The experimental setup for SDN carried out in [14] is utilized and deployed using Edge-core 4610-30T hardware switch.

B. Ransomware Samples

Nearly 1000 samples were downloaded from [16-19]. From the downloaded samples, 50 samples were modified. All the modified samples belong to the different variants of the ransomware. Besides, a self-developed ransomware named Blackwolf is used to validate the proposed IDH. Blackwolf is sophisticated ransomware, and upon the successful execution, the below-mentioned actions are performed:

- A start-up entry to load the malware on every boot (OS).
- Blackwolf is designed in such a way that it can digitally sign itself with a valid digital certificate.
- A secured remote connection is established between the victim and the C&C server.
- Blackwolf utilizes GMAIL as a Command and Control server to evade the existing security systems.
- Blackwolf infiltrates the encryption key through Gmail.
- Blackwolf tracks all the user activities in the victim machine.
- Blackwolf employs obfuscation techniques and implants its new version upon every successful infection.

C. CEP Engine

WSO2 CEP engine with the Python-based library called peak-rules is used along with the support of multimethod dispatch.

VI. Results and Discussion

A ransomware dataset structure [19] is taken into consideration, and an automation python script is executed to extract all the features as listed in the dataset. Besides, some of the host and network features [8] are also considered. Figure 8 shows the test bed's network statistics during ransomware attacks, where the samples send multiple DNS requests at a particular time interval. Figure 9 shows the ransomware activities at a particular timestamp. The experimental analysis, found that more than 200 private proxies, and 100+ onion sites are utilized for ransomware communication. However, all of the above-stated domain traffics is filtered and blacklisted by the SDN controller. If any infected host is trying to communicate with any of the blacklisted domains, the controller automatically blocks the communication and alerts the system.

The experimental analysis reveals that, on average, a single infected host tries to contact 60 different domains to establish C&C communication. This results in verifying the contacted domains, and if the contacted domain is found to be malicious, all the traffic from the contacted domain is denied before the infected host receives the encryption key.

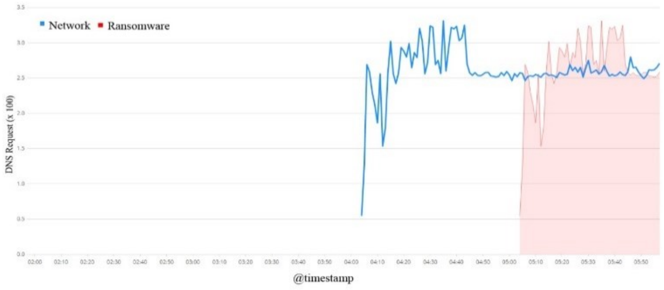


Figure 8. **DNS request - Ransomware vs. Normal**

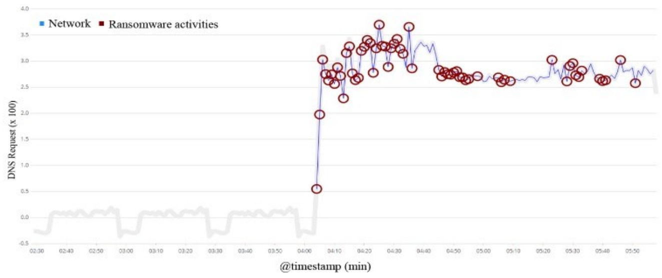


Figure 9. **Ransomware activities detected by IDH**

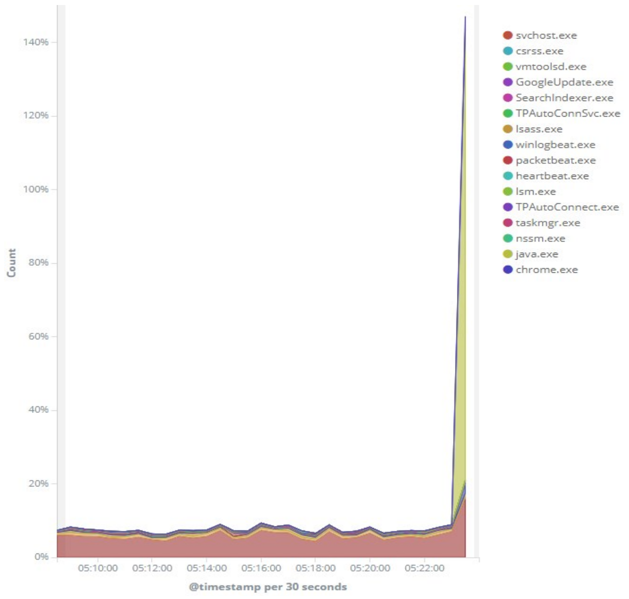
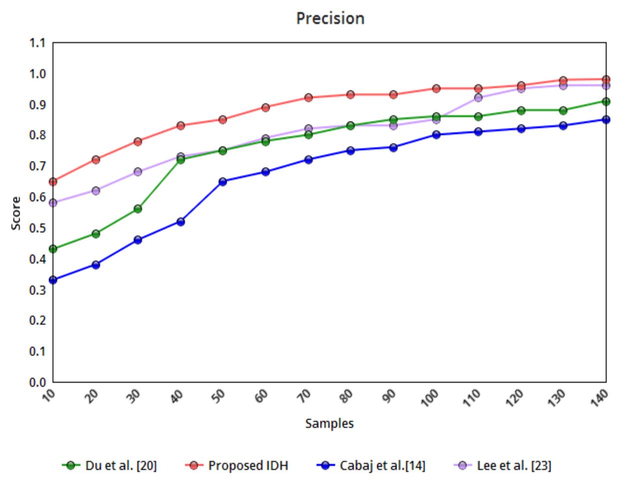
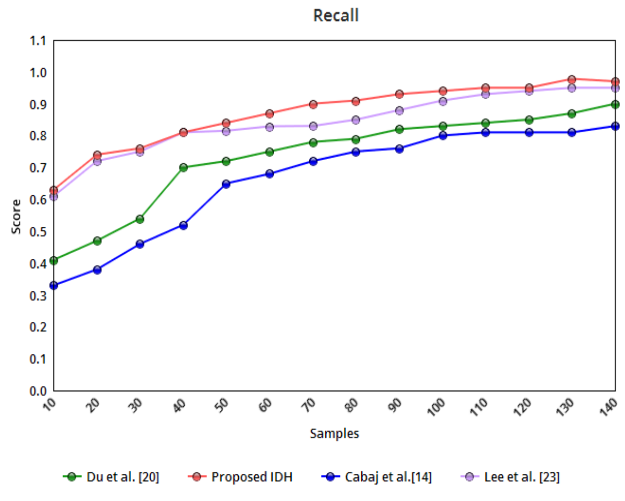


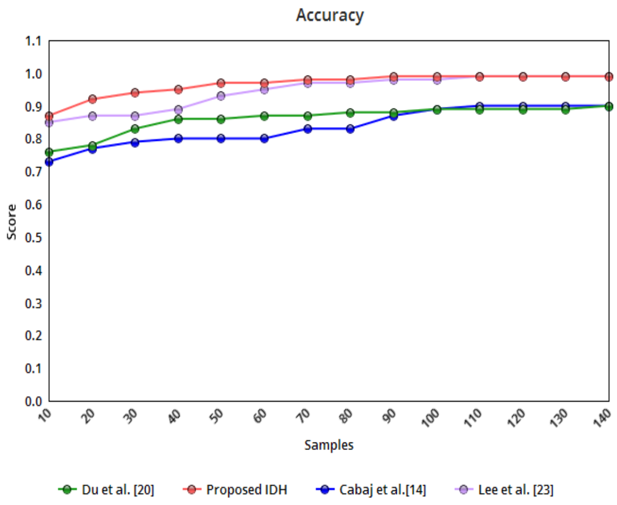
Figure 10. **CPU usage per process - attack scenario**



(a)



(b)



(c)

Figure.11 **Performance plot - a) Precision b) Recall c) Accuracy**

The average round trip time taken by the infected host to download the encryption key was 10.7s (based on the analyses), whereas the SDN reaction time to deny the suspicious traffic is about 1.7s. Hence it is proved the SDN based solutions are feasible and the features such as DNS response, DNS incoming bytes and outgoing bytes, DNS requests, DNS requests at specific timestamps, Total count of traffic rate generated by the infected hosts, DNS timestamps, etc., are considered to be the essential features to analyze the malware communication. Further, for host monitoring, a self-developed shell script is used to monitor CPU usage, memory usage, file system activities, processes, registry entries. Figure 10 shows the results of CPU usage per process during ransomware activities.

A. *Discussion*

The samples are tested sequentially, and the entire experimentation takes more than 1.5 hours to process the samples as a whole (1000 downloaded samples + 50 modified samples + 200 benign samples). From the experimental results, it is found that the IDH is capable of detecting all the downloaded, modified samples, and the Blackwolf ransomware samples. The performance of the IDH is compared with the state of the art Honeypot and ransomware detection systems [20] is shown in Figure 11. In order to measure the efficiency of the proposed IDH and Honeypot, ransomware detection systems [20][14][23], the same number of samples is tested and validated in both the proposed IDH and Honeypot, ransomware detection systems [20][14][23]. A confusion matrix is constructed based on the results obtained, and the performance is evaluated based on the accuracy (a ratio of correctly predicted ransomwares to the total number of samples), precision (a ratio of correctly predicted positive ransomwares to the total predicted positive ransomwares) and Recall (a ratio of correctly predicted positive ransomwares to the total number of samples).

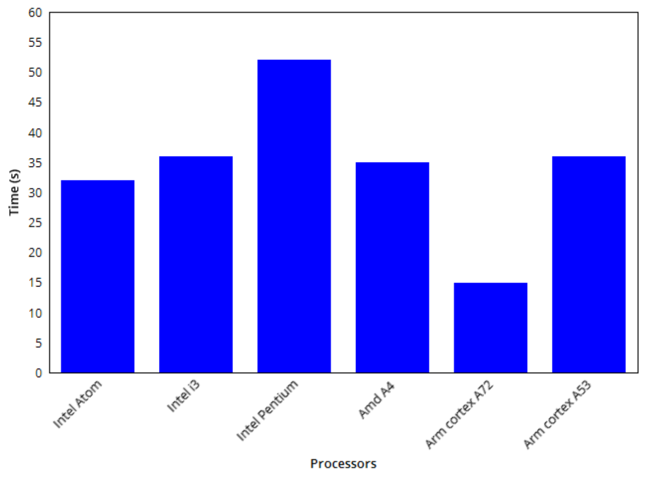


Figure.12 **Execution time of various processors**

It is observed that the proposed IDH outperforms the existing Honeypot and ransomware detection systems [20][14][23] in terms of detection accuracy. Figure 12 shows that the proposed IDH is cost-efficient and capable of running in a minimal configuration, and the time taken for ransomware detection is reasonable when deployed in IoT platform.

VII. Conclusion

This paper proposed a novel IDH. The proposed IDH utilizes the CEP technique to correlate the host features, network features, and various events from other systems such as Audit watch and firewall, thus producing the aggregated results with better accuracy. Further, the proposed IDH can be deployed in the production environment at ease. The results show that the Honeyfolder deployed for monitoring file system (host) activities is highly effective at drawing attention to the host's ransomwares. Further, it is found that the utilization of SDN infrastructure significantly improves network protection by applying simple control rules. The experimental evaluation also confirms that the proposed IDH is efficient in restricting the ransomware activities without a minimal data loss. There are more possibilities to extend the proposed IDH model to investigate ransomware attacks on healthcare implants and other internet-connected gadgets such as connected toys, etc. In the future, we planned to enhance the proposed IDH with transfer learning ability to optimize the load on the resource-constrained devices. Furthermore, the auto-tuning feature will also be added to the proposed IDH.

References

1. A. L. Young and M. Yung, "On Ransomware and Envisioning the Enemy of Tomorrow," in *Computer*, vol. 50, no. 11, pp. 82-85, November 2017.
2. A. O. Almashhadani, M. Kaiiali, S. Sezer and P. O'Kane, "A Multi-Classifier Network-Based Crypto Ransomware Detection System: A Case Study of Locky Ransomware," in IEEE Access, vol. 7, pp. 47053-47067, 2019.
3. N. Eliot, D. Kendall and M. Brockway, "A Flexible Laboratory Environment Supporting Honeypot Deployment for Teaching Real-World Cybersecurity Skills," in IEEE Access, vol. 6, pp. 34884-34895, 2018.
4. W. Tian et al., "Prospect Theoretic Study of Honeypot Defense Against Advanced Persistent Threats in Power Grid," in IEEE Access, vol. 8, pp. 64075-64085, 2020.
5. W. Fan, Z. Du, M. Smith-Creasey and D. Fernández, "HoneyDOC: An Efficient Honeypot Architecture Enabling All-Round Design," in IEEE Journal on Selected Areas in Communications, vol. 37, no. 3, pp. 683-697, March 2019.
6. L. Shi, Y. Li, T. Liu, J. Liu, B. Shan and H. Chen, "Dynamic Distributed Honeypot Based on Blockchain," in IEEE Access, vol. 7, pp. 72234-72246, 2019.
7. S. Sharmeen, Y. A. Ahmed, S. Huda, B. Ş. Koçer and M. M. Hassan, "Avoiding Future Digital Extortion Through Robust Protection Against Ransomware Threats Using Deep Learning Based Adaptive Approaches," in *IEEE Access*, vol. 8, pp. 24522-24534, 2020
8. R. Mohan, V. Vaidehi, Ajay Krishna A, Mahalakshmi M and S. S. Chakkaravarthy, "Complex Event Processing based Hybrid Intrusion Detection System," *2015 3rd International Conference on Signal Processing, Communication and Networking (ICSCN)*, Chennai, 2015, pp. 1-6.
9. M. Yazdani and F. Jolai, "Lion Optimization Algorithm (LOA): A nature-inspired metaheuristic algorithm," Journal of Computational Design and Engineering, vol. 3, no. 1, pp. 24–36, 2015.
10. M. Al-Hawawreh, F. d. Hartog and E. Sitnikova, "Targeted Ransomware: A New Cyber Threat to Edge System of Brownfield Industrial Internet of Things," in *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 7137-7151, Aug. 2019
11. A. O. Almashhadani, M. Kaiiali, S. Sezer and P. O'Kane, "A Multi-Classifier Network-Based Crypto Ransomware Detection System: A Case Study of Locky Ransomware," in *IEEE Access*, vol. 7, pp. 47053-47067, 2019
12. Pathak, R., &Vaidehi, V. (2015). Complex Event Refinement by Statistical Augmentation Model. International Journal of Intelligent Information Technologies, 11(2), 55-69.
13. D. Wu and J. M. Mendel, "Similarity Measures for Closed General Type-2 Fuzzy Sets: Overview, Comparisons, and a Geometric Approach," in *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 3, pp. 515-526, March 2019.
14. K. Cabaj and W. Mazurczyk, "Using Software-Defined Networking for Ransomware Mitigation: The Case of CryptoWall," in *IEEE Network*, vol. 30, no. 6, pp. 14-20, November-December 2016.
15. E. Berrueta, D. Morato, E. Magaña and M. Izal, "A Survey on Detection Techniques for Cryptographic Ransomware," in IEEE Access, vol. 7, pp. 144925-144944, 2019.
16. Ransomware dataset, http://kharon.gforge.inria.fr/dataset/, accessed 22.02.2020
17. Ransomware Payments in the Bitcoin Ecosystem, https://zenodo.org/record/1238041#.XzpQs-gzbcc, accessed 22.02.2020
18. Ransomware samples, https://github.com/fabrimagic72/malware-samples, accessed 22.02.2020
19. Ransomware dataset, https://github.com/behas/ransomware-dataset, accessed 22.02.2020
20. M. Du and K. Wang, "An SDN-Enabled Pseudo-Honeypot Strategy for Distributed Denial of Service Attacks in Industrial Internet of Things," in IEEE Transactions on Industrial Informatics, vol. 16, no. 1, pp. 648-657, Jan. 2020.
21. Ransomware dataset, https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/, accessed 22.02.2020.
22. Adam Bradley, The Real Cost Of Ransomware And How We Stop Paying It,

https://www.forbes.com/sites/adambradley1/2020/06/11/the-real-cost-of-ransomware-and-how-we-stop-paying-it/#1b2f41766e51, accessed 12.06.2020.

23. K. Lee, S. Lee and K. Yim, "Machine Learning Based File Entropy Analysis for Ransomware Detection in Backup Systems," in *IEEE Access*, vol. 7, pp. 110205-110215, 2019.
24. D. Su, J. Liu, X. Wang and W. Wang, "Detecting Android Locker-Ransomware on Chinese Social Networks," in *IEEE Access*, vol. 7, pp. 20381-20393, 2019
25. F. Khan, C. Ncube, L. K. Ramasamy, S. Kadry and Y. Nam, "A Digital DNA Sequencing Engine for Ransomware Detection Using Machine Learning," in IEEE Access, vol. 8, pp. 119710-119719, 2020.
26. D. Javaheri, M. Hosseinzadeh and A. M. Rahmani, "Detection and Elimination of Spyware and Ransomware by Intercepting Kernel-Level System Routines," in IEEE Access, vol. 6, pp. 78321-78332, 2018

Acronyms and Abbreviations

IoT	Internet of Things
IDPS	Intrusion Detection and Prevention System
IDS	Intrusion Detection System
AV	Anti-virus
IDH	Intrusion Detection Honeypot
SoLA	Social Leopard Algorithm
CEP	Complex Event Processing
AP	Access Points
C&C	command and control
SDN	Software Defined Networking
PSO	Particle Swarm Optimization
ACO	Ant colony optimization
MBO	Marriage in Honey Bee Optimization
LOA	Lion Optimization algorithm
D-CEP	Distributed CEP engine
ONOS	Open Network Operating System
SGX	Software Guard Extension
DBA	Dynamic monitoring and behavior Analysis