Computers, Materials & Continua
DOI: 10.32604/cmc.2022.028646
Article

Tech Science Press

An Asset-Based Approach to Mitigate Zero-Day Ransomware Attacks

Farag Azzedin*, Husam Suwad and Md Mahfuzur Rahman

Information & Computer Science Department, KFUPM, Dhahran, KSA
*Corresponding Author: Farag Azzedin. Email: fazzedin@kfupm.edu.sa
Received: 15 February 2022; Accepted: 19 April 2022

Abstract: This article presents an asset-based security system where security practitioners build their systems based on information they own and not solicited by observing attackers' behavior. Current security solutions rely on information coming from attackers. Examples are current monitoring and detection security solutions such as intrusion prevention/detection systems and firewalls. This article envisions creating an imbalance between attackers and defenders in favor of defenders. As such, we are proposing to flip the security game such that it will be led by defenders and not attackers. We are proposing a security system that does not observe the behavior of the attack. On the contrary, we draw, plan, and follow up our own protection strategy regardless of the attack behavior. The objective of our security system is to protect assets rather than protect against attacks. Virtual machine introspection is used to intercept, inspect, and analyze system calls. The system call-based approach is utilized to detect zero-day ransomware attacks. The core idea is to take advantage of Xen and DRAKVUF for system call interception, and leverage system calls to detect illegal operations towards identified critical assets. We utilize our vision by proposing an asset-based approach to mitigate zero-day ransomware attacks. The obtained results are promising and indicate that our prototype will achieve its goals.

Keywords: Zero-day attacks; ransomware; system calls; virtual machine introspection

1 Introduction

Data security and privacy is growing as an open research area [1–3]. The vulnerabilities in poor encryption or network protocols directly affect the confidentiality and integrity of data. To make edge security successful, stored data needs to be protected against illegitimate access or manipulation [2,4]. Hence, valuable information and data need security solutions to stay out of reach of attackers. Despite continuous security solutions, attackers are still capable of penetrating security systems causing damage to valuable data and affecting economy impact [5,6]. A zero-day vulnerability is an unknown computer software bug discovered first by attackers before the vendor has become aware of it [6,7]. Until the vulnerability is mitigated, attackers can exploit it to adversely affect programs or

data. An exploit directed at a zero-day is referred to as zero-day attack. Attackers hide their behavior by changing their attack vector to avoid systematic antivirus software [8].

During the first half of 2021, Kaspersky experts have observed an increase in attacks exploiting zero-days [9]. Duqu 2.0 has been used to attack victims in countries in the Middle East, Asia as well as Western countries. Not to mention Stuxnet, which sabotaged centrifuges for uranium enrichment plant and spread through USB flashes exploiting four zero-day vulnerabilities. At the end of 2018, Shamoon 2 attack came back with new features since its appearance in 2012. It achieved its maximum damage by overwriting the master boot records and wiping entire hard disks. In June 2019, Microsoft was yet another victim of zero-day attack exploiting the local escalation privileges that were a vulnerable component of Microsoft Windows.

It is anticipated that ransomware attacks will keep growing costing victims a total of 265 billion by 2031. Despite this financial burden, it is also estimated that the ransomware attacks success rate will grow to an attack every two seconds by 2031 [10]. Victims are spending significantly in remediation attempts whereby the cost of recovery doubled from about 761, 106 in 2020 to 1.85 million in 2021 [10].

Most of the financial and damage impact comes from zero-day attacks and ransomware malware. A zero-day attack is an undisclosed vulnerability that hackers can exploit to adversely affect computer programs [6]. Ransom malware is a type of malicious software that blocks access to data or threatens to publish it unless a ransom is paid [11]. CryptoLocker, CryptoWall, WannaCry, Jigsaw, TeslaCrypt, Bad Rabbit, and Petya are examples of famous and recent ransom malware [5]. The importance of securing systems against zero-day ransomware attacks is needed and therefore, a security system must be put in place to ensure that such attacks are prevented [12]. This motivated academics and researchers [6–8,13] to achieve more secure environments and reduce big losses resulting from such attacks. Undoubtedly, the growing rate of security incidents and cost show that current security solutions cannot stop the sophistication and complexity of these attacks since we are guarding against the unknown [14]. This is evident by the fact that virus scan programs always need to be updated.

In this article, we argue that the failure of current security systems is the result of building security systems based on information solicited by observing attackers' behavior. Defenders are followers and defense systems are built by de-fenders based on solicited input from the attacker itself. After the defender collects the input and builds its defense system, the attacker changes its attack vector and new defense system needs to be rebuilt and the vicious cycle continues. As such, current security solutions have several drawbacks: (a) provide subjective solutions, (b) provide solutions incapable of detecting unknown attacks, and (c) provide solutions that use predictive or reactive strategies. This paper proposes to change the role of defenders. Defenders know exactly their assets, know exactly their security requirements, and know exactly what the consequences are if security requirements are violated. Therefore, defenders can build defense systems based on information they own not solicited nor given by attackers.

As such, the contribution of this paper is proposing a new vision to the security life cycle. Our proposed asset-based security system enables security practitioners build their security systems based on information they own. The idea is to completely build security systems based on information we own and not solicited by observing attackers' behavior. This vision has threefold: First, it enables defenders to lead the security game. Second, it provides security solutions capable of detecting known and unknown attacks. Third, it provides security solutions that use proactive instead of predictive or reactive strategies. That is, our proposed approach is an online security evaluation framework that leverages dependencies between OS-level objects to protect critical asset. The different components of our approach address three important limitations faced by existing security techniques. First, a

reachability graph captures the subjective security requirements and minimizes administrator input. Second, our proposed approach does not rely on assumptions about the behavior of attackers or system vulnerabilities. Third, a monitoring phase is combined with the reachability graph to evaluate the system wide compromise without making assumption about attack vectors. It should be noted that the monitoring collects system calls generated by the virtual machine without its knowledge since system calls are intercepted and logged at the hypervisor-level.

1.1 Article Organization

The rest of the article is organized as follows. Section 1.1 states the problem statement while Section 2 discusses the related work. The proposed approach and the system design are outlined in Sections 3 and 4, respectively. Performance evaluations are conducted in Section 5 while Section 6 concludes the paper and envisions future directions.

1.2 Problem Statement

Current security solutions rely on information coming from attackers. Examples are current monitoring and detection security solutions such as intrusion prevention/detection systems and firewalls. These security solutions rely on either history-matching or behavior-monitoring. History-matching is seen as offering a temporary solution [8] because this history is provided by the attacker itself. That is, if a new attack comes, history-matching will fail. On the other hand, behavior is also provided by the attacker itself. As such, the success of behavior-monitoring depends on how close the new attack's behavior is to the old attacks' behavior. Behavior-based and signature-based security systems are creating an imbalance in favor of attacks. Recent defense systems such as Moving Target Defenses (MTDs) rebalance the ground between attackers and defenders. This article envisions creating an imbalance between attackers and defenders in favor of defenders. As such, we are proposing to flip the security game such that it will be led by defenders and not attackers. We are proposing a security system that does not observe the behavior of the attack. On the contrary, we draw, plan, and follow up our own protection strategy regardless of the attack behavior. The objective of our security system is to protect assets rather than protect against attacks.

2 Related Work

The ever-increasing pace of computing infrastructures and systems such as Internet of Things have resulted in the extreme need to secure against the exponential attack growth in number and complexity [15]. Zero-day attacks have been surveyed and classified by many researchers and practitioners [6,7,16]. The work done in [17,18] surveyed and classified detection techniques into signatures detection techniques, statistical-based techniques, and behavior-based detection techniques. On the other hand, researchers in [19] introduced a metric to rank safety from zero-day attacks by counting how many such vulnerabilities would be required before compromising network assets. The ranking algorithm included insider attackers and gave the same weight to all zero-day vulnerabilities. Another classification techniques were presented in [20]. Data mining was used to detect and classify zero-day malware based on the frequency of windows API calls using supervised learning algorithms. Various classifiers were trained through analyzing the behavior of large database with and without malicious codes. This system depends mostly on features extracted from previous attacks. Detecting zero-day attacks was also the concern of many researchers. A behavior-based scheme was proposed in [21] to detect zero-day android malware before releasing android applications into the public domain. This

technique automatically monitors dangerous behaviors of such applications to warn users of zero-day attacks such as launching roots exploit or sending background SMS messages. Recently, authors in [22] conducted a survey on machine learning techniques to detect ransomware attacks. The survey shows that machine learning techniques are used efficiently in various applications such as ransomware detection, spam detection, text classification, and pattern recognition. Further, it was shown that deep learning mitigates the burden of re-engineering the features for the new types of malware.

The recent zero-day ransomware attacks also earned the attention of the research arena. Authors in [23] introduced R-Locker to countermeasure zero-day ransomware attacks. A honey file is created and acts as a trap to elude and minimize the damage done to real assets. Various links are added to the honey file to divert the malware and learn its tools, tactics, and motives. Countermeasures will then be initiated to eradicate the damage, if any, done by the ransomware.

Network intrusion detection system was proposed in [24] to detect zero-day attacks. The proposed system is required to detect attacks in network traffic. The proposed approach consists of three phases namely, preprocessing, feature elimination, and detection. In the first phase, the network traffic is preprocessed through transformation and min-max methods. Then, the random forest recursive feature elimination method is utilized to identify optimal features that positively impact the performance. Finally, Support Vector Machine types are used to detect intruders.

A survey is conducted in [25] to pinpoint ransomware success factors. This survey found that reasons behind the spreading of ransomware attacks and their success are the techniques used by the ransomware itself nor unknown-nature of zero-day ransomware attacks. The available technology and applications played a vital role in enabling the adversary to hide their payment transaction with the ability to reach as many victims as possible in short time. Due to inefficiency and the static-nature of antivirus programs, authors in [26] developed a behavior-based compromise system. This system detects data breach using machine learning techniques by analyzing network traffic to identify zero-day ransomware. Authors targeted WannaCry ransomware.

Android ransomware attacks were the focus in [27]. A large-scale of 2,721 Android ransomware sample was collected and characterized to ensure the majority of existing Android malware are covered and reflected in the sample. The paper proposed RansomProber, a real time behavior-based ransomware detection system. Evaluation experiment was conducted to compare the overall detection accuracy analysis tools, anti-virus solutions, and RansomProber. RansomProber outperformed two state-of-the-art malware analysis tools and several commercial solutions with a detection accuracy of 99%. A group of academics [28–33] introduce MTD. The MTD is the concept of morphing the target, making it unfamiliar to the attacker. Therefore, the attacker is forced to learn the target repeatedly. Consequently, this will (a) reduce the attacker's window of success and (b) increase the costs of their probing and efforts of their attack. MTD systems basically disturb the reconnaissance phase by creating asymmetric uncertainty on the attacker's side. Therefore, MTD systems constantly adapt to the current intrusion trends. This makes MTD systems learning systems which is necessary to have an effective asymmetric uncertainty. As such, MTD systems are still learning from attackers and therefore depending on information they themselves do not own.

3 Our Security System

As shown in Fig. 1, our approach consists of three phases, namely information collection, monitoring, and decision. In the information collection phase, we identify critical assets and their security requirements, while the monitoring phase captures system calls that need to be investigated by the decision phase. The decision phase assures that critical assets security requirements are not

violated. If there is an attempt of violation, a decision is needed to deal with this attempt and alert the security system.

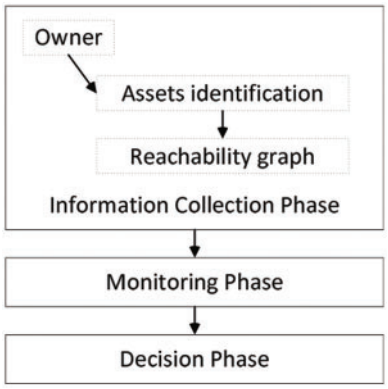

Figure 1: Asset-based security system

3.1 Information Collection Phase

This phase involves collecting information about the guest operating system, the critical assets, and the security requirements of these critical assets. Critical assets are assumed to be objects (e.g., files, processes, sockets) that are created and managed by the guest virtual machine. Therefore, the paths of these objects are collected along with their security requirements. In addition, information related to the guest operating system is also collected such as operating system type, system calls and how these system calls map to security requirements. It should be noted that critical assets are always associated with security requirements.

3.2 Critical Assets Identification

The objective here is to identify critical assets along with their security requirements. The asset owner provides this information to the security practitioner. After the critical assets are identified, the security requirement for each asset must be also specified by the asset owner. For example, in a university environment, the Registrar database might be identified as the critical asset. This identification is done by the University Board. The University Board might require only Registrar database integrity since integrity is crucial to issue transcripts and degree certificates. To the University Board, availability and confidentiality might not be as important as integrity for the University Registrar database. The security practitioner needs to identify the process(s) used to access the critical asset, let us say it is process p1. This means that the Registrar database identified as a critical asset can be accessed only by p1. Hence, p1 is the only authorized process to modify the Registrar database.

3.3 Reachability Graph

The reachability graph captures the low-level interrelation-ships between the critical assets and any other objects in the system. In a nutshell, the reachability graph tells us which processes and files are used to reach the critical assets identified in the Assets Identification step. This is established by intercepting system calls at the hypervisor-level. Particularly, we will identify all low-level objects that cause data dependencies with the critical assets identified in the previous phase. For example, to access the Registrar database r, the database administrator uses process p that might call another process q or use another file f. All processes and files involved in this cycle (i.e., p, q, and f) are added as

critical assets. It should be noted that all low-level critical assets will have integrity as their security requirement because any unauthorized modification to the low-level critical assets can violate the security requirement of the critical assets identified by the user. Let us assume that p reads the grades from f and writes them to r. Then, if f is modified by unauthorized user, the integrity of r is violated.

Therefore, the Information Collection Phase will generate critical assets that can be provided as a simple list, a prioritized list, or a more complex representation. For simplicity and clarity, the critical assets might be represented as a set C = {r, p, q, f}. Therefore, C is a set of elements in which each element is a 2-tuple containing critical asset and policy. Each critical asset has a policy composing of the critical asset's security requirement and the set of processes authorized to access the critical asset.

$$C = \{(r, P_r), (p, P_p), (q, P_q), (f, P_f)\} \tag{1}$$

$$P_r = \{integrity, \{p\}\} \tag{2}$$

$$P_p = P_q = P_f = \{integrity, \{\}\} \tag{3}$$

As shown in Eq. (1), C has four critical assets r, p, q, and f. Eq. (2) defines the policy of r (Pr) as only process p is authorized to modify r while Eq. (3) equates the policies of p, q, and f as no process is authorized to modify their respective critical assets. Here, the policy for each critical asset is a set of processes authorized to access the critical asset without violating its security requirement. Policies can be more complicated. For example, we can use one-time passcode as well as time, date, or frequency of access to the critical asset. Specific content-based security features include restricting who can open, email, print or edit a piece of content and placing a time limit on how long a user can access a given piece of content. Content can expire from a given repository and no longer be viewable by anyone.

The security requirements are mapped to system calls by the security practitioner. Knowing the guest operating system type, system calls, as well as the security requirement, the security practitioner will identify system calls that must be prevented to preserve the security requirement. For example, the following system calls, namely NtWriteFile and NtSetInformationFile must be prevented to preserve integrity. NtDeleteFile and NtSetInformationFile system calls must be prevented to preserve availability. Likewise, NtRead and NtSetInformationFile system calls must be prevented to preserve confidentiality.

3.4 Monitoring Phase

This is the phase responsible for virtual machine introspection by collecting system calls generated by the virtual machine without its knowledge since system calls are intercepted and logged at the hypervisor-level. As shown in Algorithm 1, this phase starts by initializing S, the set of intercepted system calls. This phase monitors only data flows to the critical assets and therefore, we need to know if system call s is trying to access any critical asset c ∈ C to decide if s needs further inspection. If s is trying to access c ∈ C, then add s to S and pass s to the Decision Phase for further investigation.

Algorithm 1: Monitoring Phase

```
1 S = {}; //initializing the set of
    intercepted system calls
2 foreach intercepted system call s do
3 |   Parse s //get c and any other
  |    relevant information
4 |   if ( c ∈ C ) then
5 |   |   S = S + s //for possible
  |   |   post-mortem analysis
6 |   |   Decision Phase(C, s) //call
  |   |    Decision Phase
7 |   end
8 end
```

Fig. 2 shows a sample of intercepted system calls generated by processes running in the virtual machine. Suppose that the file abc.txt (which appears in the second system call) is in C, then only the second system call will be added to S. It should be noted that our system does asset-based monitoring. In our system, system calls are treated independently, and no conclusion is inferred regarding the behavior of theses system calls. In other approaches, monitoring is done to learn behavior or match signatures.

```
[SYSCALL] vCPU:1 CR3:0x4926d000,explorer.exe SessionID:1 ntoskrnl.exe!NtQueryDirectoryFile Arguments: 11
        IN HANDLE FileHandle: 0x3c0 -> '\'
        IN HANDLE Event: 0x0
        IN PIO_APC_ROUTINE ApcRoutine: 0x0
        IN PVOID ApcContext: 0x0
        OUT PIO_STATUS_BLOCK IoStatusBlock: 0x440d8f8
        OUT PVOID FileInformation: 0x440d910
        IN ULONG Length: 0x268
        IN FILE_INFORMATION_CLASS FileInformationClass: 0x3
        IN BOOLEAN ReturnSingleEntry: 0x1
        IN PUNICODE_STRING FileName: 0x440d870 -> 'Users'
        IN BOOLEAN RestartScan: 0x0
[SYSCALL] vCPU:0 CR3:0x88e60000,notepad.exe SessionID:1 ntoskrnl.exe!NtWriteFile Arguments: 9
        IN HANDLE FileHandle: 0x154 -> '\Users\hs7\Desktop\abc.txt'
        IN HANDLE Event: 0x0
        IN PIO_APC_ROUTINE ApcRoutine: 0x0
        IN PVOID ApcContext: 0x0
        OUT PIO_STATUS_BLOCK IoStatusBlock: 0xcec30
        IN PVOID Buffer: 0x2f5ba0
        IN ULONG Length: 0x22
        IN PLARGE_INTEGER ByteOffset: 0x0
        IN PULONG Key: 0x0
[SYSCALL] vCPU:2 CR3:0x875dd000,mspaint.exe SessionID:1 ntoskrnl.exe!NtSetInformationFile Arguments: 5
        IN HANDLE FileHandle: 0x1e8 -> '\Users\hs7\Desktop\test.png'
        OUT PIO_STATUS_BLOCK IoStatusBlock: 0x27d6b0
        IN PVOID FileInformation: 0x27d6f0
        IN ULONG Length: 0x8
        IN FILE_INFORMATION_CLASS FileInformationClass: 0x7fe0000000e
```

Figure 2: Snapshot from raw system calls

3.5 Decision Phase

The goal of this phase is to catch any attempts to violate the security requirements of critical assets. This is done by assuring that s obeys the critical asset's policy. Algorithm 2 outlines the steps of this phase. The algorithm starts by accepting the input data passed from the Monitoring Phase namely, C and s. We start by initializing the decision to allow. Then, we test if the asset c specified in s matches any critical assets specified in C. Next, we check if the process p and the system call name n specified in s is among the allowed processes in the policy of the critical asset.

Algorithm 2: Decision Phase

Data:
The critical asset set C
The intercepted system call s
Output: Either prevent or allow
1 decision ← allow; //initialize *decision to allow*
2 **if** $(c \in C)$ **then**
3 | **if** *((p and n) do not violate* P_c*)* **then**
4 | | **return** *allow*;
5 | **end**
6 **end**
7 **return** *prevent*;

4 System Design

In this section, we present a systematic approach of defining all system components to satisfy the needs and requirements to design a coherent and well-running system. Both the functional and the operational architectures of our proposed security system are presented to illustrate the working order of the various system components, as shown in Fig. 3 as well as information flow between these components.

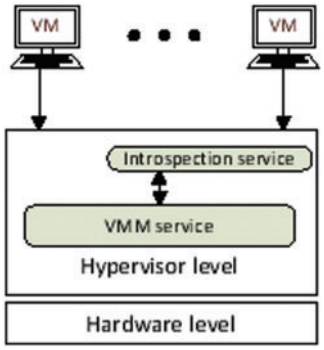

Figure 3: Overview of the system architecture

4.1 Functional Architecture

As shown in Fig. 4, the functional architecture is presented, and the aim is to show the segregation of functionalities across the different layers of the architecture. On top of the hardware layer, the hardware abstraction and the creation and management of multiple computing environment instances are the functions of the virtualization layer. The hypervisor is built on top of it. This layer, the introspection layer, sets the stage for tools and utilities to establish the core Asset-based functionalities of our system. As depicted in the figure, there are five core components in our security system. Critical assets identification along with their security requirements and the low-level interrelationships between the critical assets are done by the Identification and the Generator components, respectively. The monitoring functionality is carried out by the Observer while the Analyzer component will inspect the system call after it has been captured by the Observer. Finally, a decision needs to be made by the Decision Maker to either prevent or allow the execution of the system call. The Decision Maker also alerts the security system of potential violation attempts targeting critical assets.

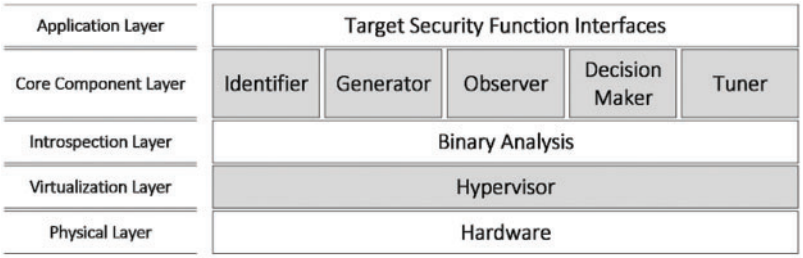

Figure 4: Functional architecture design of the proposed security system

4.2 Operational Architecture

In this section, we describe how operations are employed to accomplish functions. The primary objective is to show the derivation of operational profile from functional profile. In the operational profile, we include details such as tasks, operational elements, and information flows required to accomplish or support the functionalities of our security system. We show the operational architecture before and after the security system deployment. Fig. 5 shows the operational architecture before deployment where tasks and information are depicted as they flow between the Information Collection and the Monitoring phases. After collecting input from the assets owners as well as system call input from the operating system, critical assets (high and low) with their security requirements are generated and given to the mapper. In turn, the mapper creates the critical asset scope of control after consulting the security mapping and the high-level critical assets security policy. Now, the critical assets scope of control will be used as a reference model after the security system deployment.

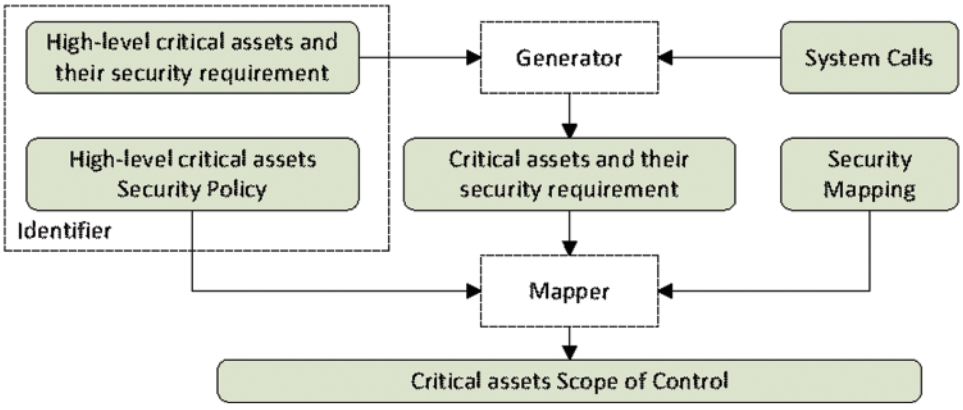

Figure 5: Operational architecture before system deployment

After deploying the security system, the Monitoring and Decision phases begin as shown in Fig. 6. The observer starts monitoring user processes-initiated system calls and consulting with the reference model (i.e., the critical assets scope of control). The observer reports any system call in violation of the reference model and will generate captured system call report that is fed to the decision maker. Consequently, the system call execution will be interrupted and stopped from execution. A warning message is sent by the decision maker to the tune our security system. Here, we can employ different strategies to harden accessibility to our critical assets. What we can do here is to make the attack surface dynamic using techniques such as bio-inspired MTD, cloud-based MTD, and dynamic

network configuration. It should be noted that this dynamicity is done without observing attack behavior.

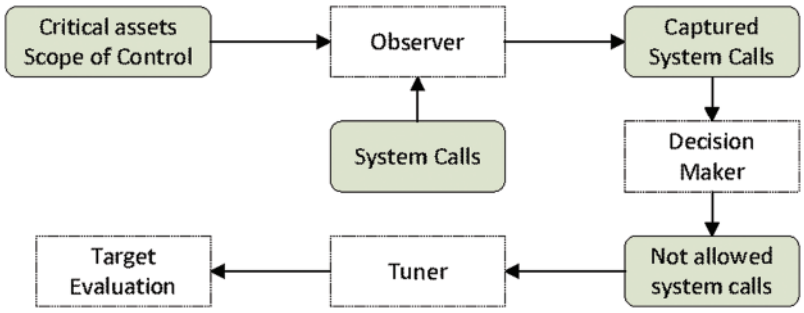

Figure 6: Operational architecture after system deployment

5 Performance Evaluation

To evaluate our security model, we implemented a prototype to present a working security model and determine its functional feasibility. In our prototype, we utilized an Intel Core i7 Quad Core 2.4 GHz and 24 GB RAM machine running Ubuntu 16.10 64 bit as the host OS. As for the guest OS, we used Windows 7 64 bit running on a virtual machine having single Core 3 GB RAM with 20 GB HDD. We used the Xen [34] hypervisor to host the virtual machines and DRAKVUF [35] to provide agentless virtual machine introspection. The Xen hypervisor is an open source baremetal hypervisor, which makes it possible to run many instances of an operating system or indeed different operating systems in parallel on a single machine. DRAKVUF provides in-depth execution tracing on guest OS using a set of plugins, each of which, traces certain events in the context of the guest OS. We also conducted several experiments to test the effectiveness, the agility, and the performance of our security model.

5.1 Xen and DRAKVUF

The Xen Project [36] is an open-source bare-metal hypervisor making it possible to run many instances of a single operating system or different operating systems in parallel on a single machine. It is the only available open source as bare-metal hypervisor. It is used as the basis for several different commercial and open-source applications, such as: security applications, Infrastructure as a Service (IaaS), desktop or server virtualization, embedded and hardware appliances. The Xen Project is the leading virtualization platform that powers some of the largest clouds giants such as Amazon Web Services and Verizon Cloud. It is also integrated into multiple cloud orchestration projects such as OpenStack and CloudStack [37].

DRAKVUF is a virtualization-based black-box binary analysis system. It does not require any special software within the virtual machine used for analysis [] [38]. DRAKVUF provides in-depth execution tracing on guest OS using a set of plugins such as SYSCALLS to trace system calls usage within the guest OS. It does so, by injecting break points at the beginning of each system call and whenever that break point is hit a callback function is invoked which prints all the details of that system call. These details are the arguments for the invoked system call, which may include a process name, process id, or a file name. We utilize this plugin to implement our prototype.

Presently, many binary analysis systems exist such as Cukoo and DRAKVUF. We selected DRAKVUF since it is out-of-box (virtual machine introspection) while Cukoo is inside-the-box (sandbox). Cuckoo can trace out only user-level malware while DRAKVUF can trace out kernel-level as well as user-level malware. We also selected Xen as a hypervisor since DRAKVUF runs best on Xen and also since Xen is type I hypervisor.

5.2 Verification and Validation

We conducted an experiment to verify and validate our prototype by running the Task Manager within the virtual machine to provide the list of the running processes (Fig. 7 left-hand side). Our Monitor system calls plugin should provide the same list of running processes assuming that all these processes are generating system calls. Indeed, and as illustrated in Fig. 7, the running processes captured by the task manager within the virtual machine are also captured by our security system (Fig. 7 right-hand side). In our next experiment, we exposed our prototype to an academic crypto-ransomware identical to the famous jigsaw crypto-ransomware [39]. A crypto-ransomware traverses interesting directories and encrypts all files that match certain file extensions. The ransomware contains 3 files, namely Server.exe, ransomware.exe, and Unlocker.exe. The Server.exe file emulates a connection between the victim machine and money seeker. The server is used to store the victim's information and the unique encryption key. The ransomware.exe file encrypts the files inside the victim's machine using AES-256-CTR and generates a list of the encrypted files and instruction for decrypting them. After following the instructions and the payment is confirmed, the encryption key and the Unlocker.exe can be used by the victim to decrypt the files. According to [39] and the analysis done by [5,40,41], the Ransomware process works in stages as follows:

1. Query the original file to be encrypted
2. Create/Open temporary output file
3. Read the content from the original file, encrypt it, and send the encrypted content to the temporary file
4. Close the original and the temporary files
5. Move the contents of the temporary file to the original file
6. Close both files and wait for all other original files to be encrypted
7. Rename the original file
   a) Create a file with base64 equivalent filename
   b) Move the encrypted content from the original file to the file with the base64 equivalent filename
   c) Delete the original file

To further verify and validate our security system in capturing system calls, we monitored the Ransomware process and captured any system call generated by the process name ransomware.exe for file issa.txt. Once our security system captures the system calls, the analysis of these system calls should follow the stages outlined above. To relate the captured system calls to the number of stages, we consulted [42] for the meaning of the system calls and we show our findings in Tab. 1. Tab. 2 shows the captured system calls, the file path accessed by ransomware.exe is accessing, and the corresponding stage number according to our analysis. As shown in Tab. 2, the file to be encrypted issa.txt is queried so that ransomware.exe can collect relevant information and then a temporary file is created and opened.

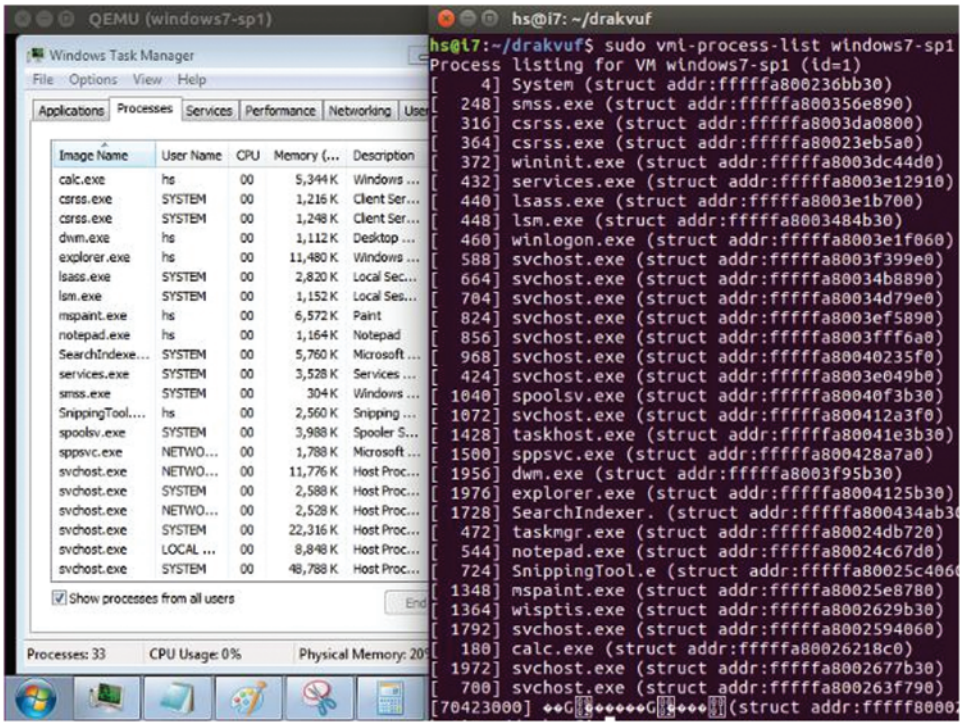

Figure 7: Process list generated within and outside the virtual machine

Table 1: Meaning of system calls

System call	Meaning
NtCreateFile	Creates new file or directory, or opens existing file, device, directory, or volume
NtOpenFile	Opens existing file, device, directory, or volume, and returns file object handle.
NtQueryFull-AttributesFile	ZwQueryFullAttributesFile gives network open info. for specific file.
NtQuery-InformationFile	ZwQueryInformationFile returns various information about a file object.
NtReadFile	ZwReadFile reads data from an open file.
NtSet-InformationFile	ZwSetInformationFile changes various information about a file object.
NtWriteFile	ZwWriteFile writes data to an open file.

Stage 3 then starts by opening and reading from isaa.txt and writing to the temporary file. Stage 4 then closes the original and the temporary files. During the final stage (i.e., stage 7), ransomware.exe creates a file with base64 equivalent filename and move the encrypted content from the original file to the file with the base64 equivalent filename and finally deletes issa.txt. This shows that our security system captured all system calls generated by ransomware.exe and in the correct sequence.

Table 2: System calls initiated by ransomware.exe for issa.txt

Captured system call	File path	S#
NtQueryFull AttributesFile	\Users\hs\Desktop\issa.txt	1
NtCreateFile	\Users\hs\AppData\Local\Temp\issa.txt	2
NtCreateFile	\Users\hs\Desktop\issa.txt	3
NtWriteFile	\Users\hs\AppData\Local\Temp\issa.txt	3
NtReadFile	\Users\hs\Desktop\issa.txt	3
NtWriteFile	\Users\hs\AppData\Local\Temp\issa.txt	3
NtReadFile	\Users\hs\Desktop\issa.txt	3
NtClose	\Users\hs\AppData\Local\Temp\issa.txt	4
NtClose	\Users\hs\Desktop\issa.txt	4
NtCreateFile	\Users\hs\Desktop\issa.txt	5
NtSet InformationFile	\Users\hs\Desktop\\&\issa.txt	5
NtCreateFile	\Users\hs\AppData\Local\Temp\issa.txt	5
NtReadFile	\Users\hs\AppData\Local\Temp\issa.txt	5
NtWriteFile	\Users\hs\Desktop\issa.txt	5
NtReadFile	\Users\hs\AppData\Local\Temp\issa.txt	5
NtClose	\Users\hs\Desktop\issa.txt	6
NtClose	\Users\hs\AppData\Local\Temp\issa.txt	6
NtCreateFile	\Users\hs\Desktop\issa.txt	7
tCreateFile	\Users\hs\Desktop\aXNzYS50eHQ = .encrypted	7
NtReadFile	\Users\hs\Desktop\issa.txt	7
NtWriteFile	\Users\hs\Desktop\aXNzYS50eHQ = .encrypted	7
NtReadFile	\Users\hs\Desktop\issa.txt	7
NtClose	\Users\hs\Desktop\aXNzYS50eHQ = .encrypted	7
NtOpenFile	\Users\hs\Desktop\issa.txt	7
NtQuery-InformationFile	\Users\hs\Desktop\issa.txt	7
NtSet InformationFile	\Users\hs\Desktop\issa.txt	7

Finally, we conducted an experiment to verify that ran-somware.exe is working properly in our environment. There-fore, we did not identify any critical files on the virtual machine and ran ransomware.exe on the virtual machine while our security system is running on the hypervisor-level. The ransomware process was able to encrypt all files. This is expected since there are no critical files and therefore our security system has nothing to defend, and the ransomware was allowed to encrypt all files.

5.3 Agility of the Security Model

Furthermore, we run yet another experiment to defend a critical asset identified as abc.txt. We started this experiment by setting the security requirements for abc.txt according to Eqs. (1) to (3) explained in Section 4-A as follows:

$$C = \{(abc.txt, P_{abc})\} \tag{4}$$

$$P_{abc} = \{(confidentiality, \{\}), (integrity, \{\}), (availability, \{\})\} \tag{5}$$

This means that confidentiality, integrity, and availability for abc.txt is defended against any process in the system. While our security system was running, we tried to access abc.txt within the virtual machine using various processes and indeed our security system enforced the security requirements for file abc.txt by not allowing any process to access abc.txt.

We conducted another experiment, where we identified three critical assets as abc.txt, abd.txt, and abe.txt. The security requirements were also identified for each file. The security requirement for abc.txt is confidentiality, integrity, and availability. similarly, the security requirements for abd.txt are integrity and availability whereas only availability is required for abe.txt. Therefore, and according to Eqs. (1) to (3) explained in Section 3.3, we have the following:

$$C = \{(abc.txt, P_{abc}), (abd.txt, P_{abd}), (abe.txt, P_{abe})\} \tag{6}$$

$$P_{abc} = \{(confidentiality, \{Notepad\}), (availability, \{Notepad\})\} \tag{7}$$

$$P_{abd} = \{(integrity, \{Notepad\}), (availability, \{Notepad\})\} \tag{8}$$

$$P_{abe} = \{(availability, \{Notepad\})\} \tag{9}$$

While our security system was running, we tried to access abc.txt within the virtual machine using various processes and indeed our security system enforced the security requirement for file abc.txt by not allowing any process to access abc.txt. We were successful in reading abc.txt using Notepad but we could not modify the file using Notepad. Other processes could not modify, rename, copy, or delete abc.txt. If the process is not Notepad, our security system blocks all system calls containing abc.txt as a file parameter. We also applied our security system to defend abd.txt and abe.txt and similar results were obtained.

5.4 Agility Against Crypto-Ransomware

We exposed our prototype to crypto ransomware [39] to test the agility of our security system in defending assets against a real-world ransomware attack. We created two critical files abd.txt and abe.txt with their security requirements as explained in Eqs. (4), (6) and (7). The crypto ransomware was able to encrypt all files, as shown in Fig. 8, except the two identified critical files namely, abd.txt and abe.txt. We investigated the system calls that were blocked from accessing abd.txt and we found out that our security system blocked NtWriteFile, NtSetInformationFile, and NtDelete-File. These are exactly the system calls mapped to integrity and availability. Similarly, the system calls blocked from accessing abe.txt are exactly the ones corresponding to NtReadFile and NtSetInformationFile. These are exactly the system calls mapped to availability. This shows that our security system defends identified critical assets by ensuring that their security requirements are not violated. Our security system does not require the signature, nor the behavior of the ransomware and it does not depend on information provided by the ransomware. As such, our security system is purely asset-based.

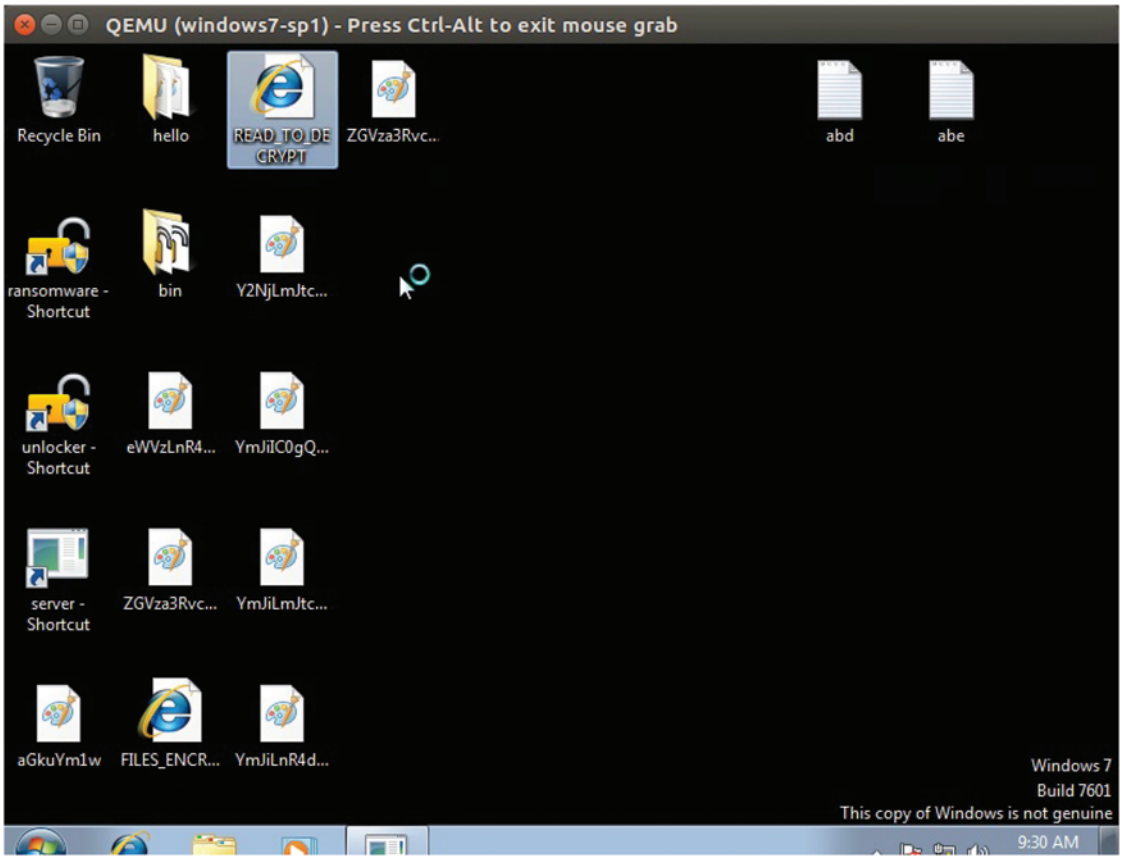

Figure 8: Crypto ransomware was able to encrypt all files except the two identified critical files

6 Conclusion and Future Work

This article proposes an asset-based security system where security practitioners build their security systems based on information they own. The idea is to completely build security systems based on information we own and not solicited by observing attackers' behavior. In this article, we outline the architecture of the proposed asset-based security system and developed a prototype of our system. We also conducted extensive evaluation experiments to evaluate the feasibility and performance of our prototype. Obtained results are encouraging and show the agility of our prototype to ransomware attacks. This paper proposed a solution to detect potential zero-day attacks by building an adaptive defense system, which including three phases of collecting information to identify critical assets, monitoring the running processes (system calls), and then making corresponding decisions to catch the malicious behaviors of accessing the critical assets. The authors prototyped the solution, showing the effectiveness of proposed methodology. As future work, we will evaluate the overhead of our security system and design it to be operating system independent. Another future direction is to expand the implementation to support Linux, Android or any other OS.

Acknowledgement: This research was funded by King Abdulaziz City for Science and Technology (KACST) under the National Science, Technology, and Innovation Plan (Project Number 11-INF1657-04). The authors would like to acknowledge the support provided by the Deanship of Scientific Research at King Fahd University of Petroleum & Minerals (KFUPM). This project is

funded by King Abdulaziz City for Science and Technology (KACST) under the National Science, Technology, and Innovation Plan (Project Number 11-INF1657-04).

Funding Statement: This project is funded by King Abdulaziz City for Science and Technology (KACST) under the National Science, Technology, and Innovation Plan (Project Number 11-INF1657-04).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] T. Wang, Y. Mei, X. Liu, J. Wang, H. -N. Dai *et al.,* "Edge-based auditing method for data security in resource-constrained internet of things," *Journal of Systems Architecture*, vol. 114, pp. 101971, 2021.

[2] R. Bingu, S. Jothilakshmi and N. Srinivasu, "A comprehensive review on security and privacy preservation in cloud environment," in *Sustainable Communication Networks and Application*, Springer, Singapore, pp. 719–738, 2022.

[3] J. Liang and J. Bai, "Data security technology and scheme design of cloud storage," in *Int. Conf. on Big Data Analytics for Cyber-Physical System in Smart City*, Singapore, Springer, pp. 87–93, 2022.

[4] X. Liu, J. Zhao, J. Li, B. Cao and Z. Lv, "Federated neural architecture search for medical data security," *IEEE Transactions on Industrial Informatics*, 2022. [Early Access].

[5] Q. Chen, S. R. Islam, H. Haswell and R. A. Bridges, "Automated ransomware behavior analysis: Pattern extraction and early detection," in *Int. Conf. on Science of Cyber Security*, China, Springer, pp. 199–214, 2019.

[6] R. Tang, Z. Yang, Z. Li, W. Meng, H. Wang *et al.,* "Zerowall: Detecting zero-day web attacks through encoder-decoder recurrent neural networks," in *IEEE INFOCOM 2020-IEEE Conf. on Computer Communications*, Virtual Conference, pp. 2479–2488, 2020.

[7] W. Haider, N. Moustafa, M. Keshk, A. Fernandez, K. -K. R. Choo *et al.,* "FGMC-HADS: Fuzzy Gaussian mixture-based correntropy models for detecting zero-day attacks from linux systems," *Computers & Security*, vol. 96, pp. 101906, 2020.

[8] C. Karr, "The IT security vicious cycle of "Assuming compromise"," 10 Feb. 2015. [Online]. Available: http://www.itproportal.com/2015/02/10/security-vicious-cycle-assuming-compromise, Accessed: 25 Apr. 2021.

[9] N. Daswani and M. Elbayadi, *Big Breaches: Cybersecurity Lessons for Everyone*, New York City, Apress, Springer, 2021. [Online]. Available: https://link.springer.com/book/10.1007/978-1-4842-6655-7.

[10] K. Balaji and T. Subbulakshmi, "Malware analysis using classification and clustering algorithms," *International Journal of e-Collaboration (IJeC)*, vol. 18, no. 1, pp. 1–26, 2022.

[11] A. Young and M. Yung, "Cryptovirology: Extortion-based security threats and countermeasures," in *Security and Privacy*, *1996 Proc., 1996 IEEE Symp. on*, Oakland, CA, USA, IEEE, pp. 129–140, 1996.

[12] L. Bilge and T. Dumitras, "Before we knew it: An empirical study of zero-day attacks in the real world," in *Proc. of the 2012 ACM Conf. on Computer and Communications Security*, Raleigh, North Carolina, USA, ACM, pp. 833–844, 2012.

[13] J. H. Jafarian, E. Al-Shaer and Q. Duan, "Adversary-aware IP address randomization for proactive agility against sophisticated attackers," in *Computer Communications (INFOCOM)*, *2015 IEEE Conf. on*, IEEE, Kong, China, pp. 738–746, 2015.

[14] S. Sibi Chakkaravarthy, D. Sangeetha, M. V. Cruz, V. Vaidehi and B. Raman, "Design of intrusion detection honeypot using social leopard algorithm to detect IoT ransomware attacks," *IEEE Access*, vol. 8, pp. 169944–169956, 2020.

[15] S. Hariri, "Cybersecurity lab as a service (CLaaS)," 2018. [Online]. Available: http://nsfcac.arizona.edu/research/claas.html, Accessed: 20 Nov. 2021.

[16] A. Blaise, M. Bouet, V. Conan and S. Secci, "Detection of zero-day attacks: An unsupervised port-based approach," *Computer Networks*, vol. 180, pp. 107391, 2020.
[17] H. Hindy, D. Brosset, E. Bayne, A. K. Seeam, C. Tachtatzis *et al.,* "A taxonomy of network threats and the effect of current datasets on intrusion detection systems," *IEEE Access*, vol. 8, pp. 104650–104675, 2020.
[18] R. Kaur and M. Singh, "A survey on zero-day polymorphic worm detection techniques," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1520–1549, 2014.
[19] L. Wang, S. Jajodia, A. Singhal, P. Cheng and S. Noel, "K-zero-day safety: A network security metric for measuring the risk of unknown vulnerabilities," *IEEE Transactions on Dependable and Secure Computing*, vol. 11, no. 1, pp. 30–44, 2014.
[20] V. Kumar and D. Sinha, "A robust intelligent zero-day cyber-attack detection technique," *Complex & Intelligent Systems*, vol. 7, no. 5, pp. 2211–2234, 2021.
[21] M. Grace, Y. Zhou, Q. Zhang, S. Zou and X. Jiang, "Riskranker: Scalable and accurate zero-day android malware detection," in *Proc. of the 10th Int. Conf. on Mobile Systems, Applications, and Services*, New York, USA, ACM, pp. 281–294, 2012.
[22] N. Rani, S. V. Dhavale, A. Singh, A. Mehra, "A survey on machine learning-based ransomware detection," in *Proc. of the Seventh Int. Conf. on Mathematics and Computing*, Singapore, Springer, pp. 171–186, 2022.
[23] J. Gómez-Hernández, L. Álvarez-González and P. García-Teodoro, "R-Locker: Thwarting ransomware action through a honeyfile-based approach," *Computers & Security*, vol. 73, pp. 389–398, 2018.
[24] M. Mehmood, T. Javed, J. Nebhen, S. Abbas, R. Abid, *et al.,* "A hybrid approach for network intrusion detection," *Computers, Materials & Continua*, vol. 70, no. 1, pp. 91–107, 2022.
[25] B. A. S. Al-rimy, M. A. Maarof and S. Z. M. Shaid, "Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions," *Computers & Security*, vol. 74, pp. 144–166, 2018.
[26] K. Ganame, M. A. Allaire, G. Zagdene and O. Boudar, "Network behavioral analysis for zero-day malware detection–A case study," in *Int. Conf. on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments*, Vancouver, BC, Canada, Springer, pp. 169–181, 2017.
[27] J. Chen, C. Wang, Z. Zhao, K. Chen, R. Du *et al.,* "Uncovering the face of android ransomware: Characterization and real-time detection," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1286–1300, 2017.
[28] J. Cho, D. P. Sharma, H. Alavizadeh, S. Yoon, N. Ben-Asher *et al.,* "Toward proactive, adaptive defense: A survey on moving target defense," *IEEE Communications Surveys Tutorials*, vol. 22, no. 1, pp. 709–745, 2020.
[29] B. Liu and H. Wu, "Optimal D-facts placement in moving target defense against false data injection attacks," *IEEE Transactions on Smart Grid*, vol. 11, no. 5, pp. 4345–4357, 2020.
[30] X. Feng, Z. Zheng, D. Cansever, A. Swami and P. Mohapatra, "A signaling game model for moving target defense," in *INFOCOM 2017-IEEE Conf. on Computer Communications*, Atlanta, GA, USA, pp. 1–9, 2017.
[31] A. G. Bardas, S. C. Sundaramurthy, X. Ou and S. A. DeLoach, "MTD CBITS: Moving target defense for cloud-based IT systems," in *Euro-pean Symp. on Research in Computer Security*, Oslo, Norway, Springer, pp. 167–186, 2017.
[32] M. Albanese, S. Jajodia and S. Venkatesan, "Defending from stealthy botnets using moving target defenses," *IEEE Security & Privacy*, vol. 16, no. 1, pp. 92–97, 2018.
[33] J. Tian, R. Tan, X. Guan and T. Liu, "Enhanced hidden moving target defense in smart grids," *IEEE Transactions on Smart Grid*, vol. 10, no. 2, pp. 2208–2223, 2018.
[34] WIKI, "Xen project software overview," 24 Jan. 2017. [Online]. Available: https://wiki.xenproject.org/wiki/Xen_Project_Software_Overview, Accessed: 20 Nov. 2021.
[35] D. Kapil and P. Mishra, "Virtual machine introspection in virtualization: A security perspective," in *2021 Thirteenth Int. Conf. on Contemporary Computing (IC3-2021)*, New York, USA, pp. 117–124, 2021.
[36] L. Abeni and D. Faggioli, "Using xen and KVM as real-time hypervisors," *Journal of Systems Architecture*, vol. 106, pp. 101709, 2020.
[37] A. Qadeer, A. Waqar Malik, A. Ur Rahman, H. Mian Muhammad and A. Ahmad, "Virtual infrastructure orchestration for cloud service deployment," *The Computer Journal*, vol. 63, no. 2, pp. 295–307, 12 2019.

[38] D. C. D'Elia, E. Coppa, F. Palmaro and L. Cavallaro, "On the dissection of evasive malware," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2750–2765, 2020.
[39] Mauri de Souza Nunes, "A POC windows crypto-ransomware (Academic)," 5 Sep. 2016. [Online]. Available: https://github.com/mauri870/ransomware, Accessed: 09 Nov. 2021.
[40] A. Akkas, C. N. Chachamis and L. Fetahu, "Malware analysis of WanaCry ransomware," 2017. [Online]. Available: http://courses.csail.mit.edu/6.857/2017/project/20.pdf.
[41] N. Scaife, H. Carter, P. Traynor and K. R. Butler, "Cryptolock (and drop it): Stopping ransomware attacks on user data," in *Distributed Computing Systems (ICDCS)*, *2016 IEEE 36th Int. Conf. on*, *IEEE*, Nara, Japan, pp. 303–312, 2016.
[42] Microsoft, "Microsoft API and reference catalog," 2018. [Online]. Available: https://msdn.microsoft.com/en-us/library/ms123401.aspx, Accessed: 20 Nov. 2021.