BREAKING THE SECURITY VICIOUS CYCLE: AN ASSET-BASED APPROACH

BY

HUSAM ISSA MOHAMMAD SUWAD

A Dissertation Presented to the

DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

In

COMPUTER SCIENCE AND ENGINEERING

MAY 2018

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **HUSAM ISSA MOHAMMAD SUWAD** under the direction of his thesis adviser and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE AND ENGINEERING**.

Dissertation Committee

[illegible]

Dr. Farag Azzedin (Adviser)

[illegible]

Dr. Shokri Z. Selim (Member)

[illegible]

Dr. Marwan Abu-Amara (Member)

[illegible]

Dr. Moataz Ahmed (Member)

[illegible]

Dr. Mohammad Alshayeb (Member)

For: [illegible]

Dr. Adel Ahmed
Department Chairman

[illegible]

Dr. Salam A. Zummo
Dean of Graduate Studies

23/7/2018

Date

Dedication

To my **Father** and my **Mother**

To my **Wife** and my **kids** Sama, Laya and Issa

To my **Brother** and my **Sisters**

To my homeland **Halhul**

For the spirit of **Martyrs**

ACKNOWLEDGMENTS

*All praise is due to **ALLAH**, the lord and sustainer of the worlds, for his countless favour and seeing me this far in life. I appreciate the support and prayers of my **parents** all through my life and specially in this work. To my **family** and **relatives**, I say thank you all for being there for me.*

*My profound gratitude goes to my academic father **Dr. Farag Azzedin** for his constructive criticism, guidance, and the assistance he offered me throughout my thesis journey. I thank all my committee members **Prof. Shokri Z. Selim**, **Dr. Mohammad Alshayeb**, **Dr. Moataz Ahmed**, and **Dr. Marwan Abu-Amara** for their comments and support. Finally, I appreciate the help and efforts of **Mr. Turki Al-hazmi**.*

*My special thanks go to my dear **wife**, and our children **Sama Suwad**, **Laya Suwad**, and **Issa Suwad**, for their love, care, understating, patience, and thoughts throughout the entire Phd program.*

*To my special friend **Mr. Ahmad Azzedin**, my **friends**, and all **Shami Community** in **KFUPM**, I wish you all the best. I would like to Acknowledge **KFUPM** for giving me this opportunity.*

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

THESIS ABSTRACT

NAME:	HUSAM ISSA MOHAMMAD SUWAD
TITLE OF STUDY:	BREAKING THE SECURITY VICIOUS CYCLE: AN ASSET-BASED APPROACH
MAJOR FIELD:	COMPUTER SCIENCE AND ENGINEERING
DATE OF DEGREE:	May, 2018

This thesis argues that the trend of constantly chasing and changing attack vectors is contributing to the continuity of attackers-led security vicious cycle. Attackers are leading and defenders are learning. This paradigm needs to be shifted. This thesis proposes an asset-based security system where security practitioners build their security systems based on information they own. The idea is to completely rely of ourselves in building security systems and require nothing from attackers. This way, attackers chase defenders which will not just level the security playing field but will give advantage to defenders. 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 goal of mitigating attacks based on defender-only supplied information.

ملخص الرسالة

الاسم الكامل: حسام عيسى محمد سواد

عنوان الرسالة: كسر حلقة الامن المفرغة : نهج قائم على الاصول

التخصص: علوم وهندسة الحاسب الآلي

تاريخ الدرجة العلمية: ايار، 2018

نناقش في هذه الرسالة التوجه الحالي القائم على التصدي للمهاجم من خلال ملاحقة اسلوب الهجمات المتغير الذي ساهم في استمرارية الحلقة المفرغة التي يقودها المهاجمون. المهاجمون يتصدرون والمدافعون يتعلمون منهم، هذا النموذج يحتاج الى التغيير. تقترح هذه الرسالة نظام امن قائم على الاصول حيث يقوم خبراء الامن ببناء انظمتهم الامنية اعتمادا على معلومات يملكونها، والفكرة من ذلك هو الاعتماد بشكل كلي على انفسنا في بناء الانظمة بدون الحاجة لمعرفة اي معلومة من المهاجم. بهذه الطريقة سيكون المدافعون هم المتصدرون والمهاجمون هم الذين يطاردون ويتعلمون، فنكسب تحكم ورفع مستوى الملعب الامني بالاضافة الى اعطاء افضلية للمدافعين.

نحن نستخدم رؤيتنا من خلال اقتراح نهج قائم على الأصول للتخفيف من هجمات الفدية دون انتظار. النتائج التي تم الحصول عليها واعدة وتشير إلى أن نموذجنا الأولي سوف يحقق هدفه المتمثل في التخفيف من حدة الهجمات على أساس المعلومات المقدمة من المدافع فقط.

CHAPTER 1

INTRODUCTION

Valuable information and data need security solutions to stay out of reach of attackers. In spite of continuous security solutions, attackers are still capable of penetrating security systems causing damage to valuable data and affecting economy impact [4] [5] [6] [7]. Each time security systems are penetrated, a patch is needed to be placed to prevent the attackers from reaching valuable data [5]. This cycle cannot continue and a solution with new approach must be presented to break this cycle.

The first step in building a security solution is to decide what needs to be protected, which we will refer to as an asset. Assets can be any valuable resource to the asset owner and can span over employee data, intellectual property, bank accounts, and so forth. It is then up to the asset owner to decide on the security requirements that need to be protected by the security system. As such, security solutions are necessary to meet organization needs. Organizations basically need to protect their assets. This is not enough for the organization as it needs to determine the value or the importance of these assets. The organization should consider scenarios in which these assets are lost, stolen, compromised or corrupted. As such, what is needed is to place a value in terms of time and money on each category identified as valuable.

We start this chapter by section 1.1. Here, we highlight security attacks and their economy impact, while section 1.2 elaborates on the continuous need for security systems. Section 1.3 identifies in details security stages. In section 1.4, we explain the research problem and what motivated us to write this thesis. Objectives of the thesis are outlined in section 1.5 while section 1.6 discusses the thesis contributions. We present the organization of the thesis in section 1.7.

1.1 Attacks Economy Impact

During the first half of 2012, Hotmail, Azure, SkyDrive, MSN, Office 365, and Twitter experienced an outage for few hours [8] and during the second half of 2012, attacks targeted GoGrid [8], Dropbox [9] and Saudi Aramco [9]. Furthermore, a virus, identified later as Stuxnet, sabotaged centrifuges for uranium enrichment plant located in Iran. The attackers first infected five companies by targeting their computers. These companies are believed to be connected to the Iranian nuclear site. The Iranian attack is believed to spread through USB flashes [10] [11] [12], exploiting four zero-day vulnerabilities. Sony's PlayStation network, Epsilon, as well as Stratfor [8] were hit by a data breach attack in 2011. Year 2009 was no exception, Bit Bucket's server (which belongs to Amazon EC2 Cloud) went down for 19 hours [8].

Two of the top ten attacks for 2014 listed in [13] compromised Gmail where five million passwords were exposed. An attack also hit eBay compromising 145 million user accounts. Educational institutions are also targets for security attacks [14]. Attacks hit Harvard University and Penn State University causing leakage of students and faculty information [15]. In 2016, Kaspersky lab was also one victim hit by Duqu 2.0 attack [15]. Cellebrite, a company that helped the FBI to break the protection on a terrorist's locked iPhone [16], was also hacked and its products were publicly distributed. At the end of 2016, Shamoon 2 attack [17] came back with new features since its appearance in 2012. This time, it achieved its maximum damage to the oil sector in the Gulf area by overwriting the master boot records and wiping entire hard disks.

Most of the damages come from zero-day attacks and ransomware malware. A zero-day attack is an undisclosed vulnerability that hackers can exploit to adversely affect computer programs [18]. A typical zero-day attack can last 10 months on average and can infect huge number of nodes. In a zero-day attack, attackers target one or more security requirements of one or more assets. Attackers change their attack vector in order to hide their behavior and avoid systematic antivirus software [19]. Saudi Aramco was a victim of a zero-day attack in 2012. These attacks stole usernames and passwords to access users accounts and infect more than 30,000 Aramco workstations [9] [20]. Ransom malware is a type of malicious software that blocks access to data or threatens to publish it unless a ransom is paid [21]. CryptoLocker, CryptoWall, WannaCry, Jigsaw, TeslaCrypt, Bad Rabbit, and Petya are examples of famous and recent ransom malware [4].

A study done by Ponemon Institute [17] shows that business loss due to zero-day attacks continues to climb by 19% and shows the average annual loss to companies worldwide exceed 7.7 million dollars per company [22]. The average data breach costs U.S. organizations approximately 6.5 million dollars [23]. Another study conducted in 2013 and is based on a sample of 252 organizations in seven countries, concluded that 87% of small companies and 93% of large companies reported some zero-day breaches [24]. Furthermore, the UK Government estimated zero-day attacks will cost 27 billion Pounds per year.

One of the most famous and recent computer zero-day ransomware is CryptoLocker [25] which costed 30 million dollars in 100 days with 500,000 victims, with speculation that at least 0.4% of CryptoLocker victims end up paying the ransom [26]. Another damaging incident is WannaCry [27] spreading in at least 150 countries costing an estimated losses that could reach 4 billion dollars [28].

1.2 Need for Security

As we move forward towards the ever increasing importance and continuous use of data-age technology, security projects are becoming the primary focus for many practitioners and research groups. A focus research group in Oxford University [29] directs their research to find security solutions to insider threats initiated directly from employees. Products [30] such as TRITON APX, TRITON AP-

EMAIL, TRITON AP-WEB, TRITON AP-DATA and TRITON AP-ENDPOIN are available to end users as content management solutions.

One of the lead companies in security and anti-virus solutions [31] conducts a yearly cyber security competition to nurture the interest of talented people and to raise users' awareness for cyber security. Security solutions, such as the ones offered by Optilab [32] designed to handle eavesdroppers and other security threats, identify legitimate users by applying screen protection using cameras. If the user is not identified, then the screen becomes blurry to protect the information and the intruder photo is captured.

The United States Department of Homeland Security (DHS) [33], targeting to achieve it's core mission, is employing more than $240,000$ individuals in security related sectors such as border and aviation security, emergency response, chemical facility inspection, and cyber-security analysis. With securing cyberspace, DHS funds a wide variety of cyber-security projects aiming at improving security in federal and global networks. Some of these cyber-security projects are anonymous networks & currencies, critical infrastructure design and adaptive resilient systems, and cyber-security forensics.

Helping victims of ransomware attacks to retrieve their original data by decrypting the files without paying the ransom to cyber criminals, was the main idea behind establishing a website called *www.nomoreransom.org*. This is an initiative by the national high technology crime unit of the Netherlands' police, Europol's European cybercrime centers, Kaspersky, and McAfee [34]. The established websites offer decryption tools to decrypt victim files hit by any ransomware listed on the websites, which are updated each time a new ransomware is discovered. The list of ransomware include LAMBDALOCKER, NEMUCODAES, MACRANSOM, JAFF, ENCRYPTILE, AMNESIA, AMNESIA2, MOLE, BTCWARE, CRY128, CRY9, and CRYPTON. If a victim was hit by an unknown ransomware, the victim needs to only upload a sample of the encrypted files to the website where these infected files will be scanned to classify the type of ransomware. Once the ransomware is identified, a solution is provided.

Academics and researchers [19] [35] [36] are working to establish more secure environments and reduce the big losses resulting from such attacks. Researchers and academics started by surveying the existing attacks and collecting information to know the power of these attacks and what damage they can cause [35]. Others studied the attack stages [37] to establish patterns of behavior in order to match them to avoid and capture future attacks. As such, models can be established to recognize and predict normal patterns or behavior and therefore capture abnormalities. In [36] and [19] current attack solutions offered by companies were assessed and innovative strategies were introduced to countermeasure such attacks.

There is no doubt that security-related incidents are increasing. Specifically, securing systems against attacks are surely needed and therefore a security solution must be put in place to ensure that such attacks are prevented and countermeasured [37].

1.3 Adaptive Security Life Cycle

Adaptive security architectures were recently proposed by [1] [38] [39] to illustrate the four stages in any adaptive security architecture as illustrated in Figure 1.1. Benefits of an adaptive approach to security include reducing the overhead in terms of time and resources as well as empowering security teams and engaging them in worthwhile activities that will limit serious damage and protect against advanced threats. Adaptive security architectures suggest four stages to the adaptive security life cycle:

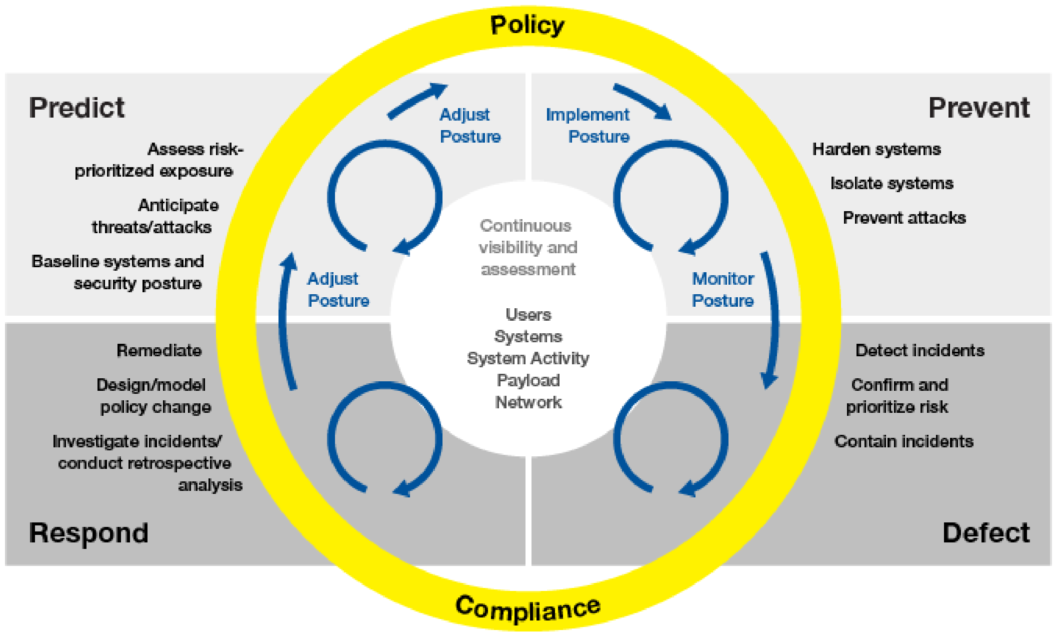

Policy
Predict
Assess risk-prioritized exposure
Anticipate threats/attacks
Baseline systems and security posture
Prevent
Harden systems
Isolate systems
Prevent attacks
Adjust Posture
Implement Posture
Adjust Posture
Monitor Posture
Continuous visibility and assessment
Users
Systems
System Activity
Payload
Network
Remediate
Design/model policy change
Investigate incidents/ conduct retrospective analysis
Respond
Detect incidents
Confirm and prioritize risk
Contain incidents
Defect
Compliance

Figure 1.1: Stages of an adaptive security architecture. Adapted from [1].

- Prevent: In this stage, known attacks are blocked before they create damage.
- Defect: In this stage, detection tools will reduce the impact of attacks, propagated from the “Prevent” stage, by limiting the time these attacks have to act on a system.

- Respond: This stage investigates any security issues that are discovered in the previous two stages. This will help avoid a recurrence of the same attack.

- Predict: In this stage, technology will be used to anticipate potential threats.

Protective measures may involve one or a combination of deterrence, avoidance, prevention, detection, recovery, and correction that should form part of the enterprise's security approach [40]. In addition, the term security has been defined by many researchers. In [40], security is defined as “A condition that results from the establishment and maintenance of protective measures that enable an enterprise to perform its mission or critical functions despite risks posed by threats to its use of information systems”. Security is defined also as “A discipline concerned with protecting networks and computer systems against threats such as hacking exploits, malware, data leakage, spam and Denial of Service (DoS) attacks, as well as ensuring trusted access through mechanisms such as IPsec or SSL” [41]. Furthermore, resiliency is the ability to quickly adapt and recover from any known or unknown changes to the environment through holistic implementation of risk management, contingency, and continuity planning. Similar definition for resiliency states that it is stated in the ability to continue to: (a) operate under adverse conditions or stress, even if in a degraded or debilitated state, while maintaining essential operational capabilities; and (b) recover to an effective operational posture in a time frame consistent with mission needs.

From the definitions of security and resiliency, we notice that there are threats, assets, and protection systems. Threats can cause possible harm and therefore they are dangerous if they can penetrate to our assets through exploiting a vulnerability. The role of the protection system is either deterrence, avoidance, prevention, detection, recovery, or correction. These terms are defined in [40] [42] as follows:

- Deterrence is reducing an intelligent threat by discouraging action, such as by fear or doubt.
- Avoidance is reducing a risk by either reducing the value of the potential loss or reducing the probability that the loss will occur.
- Prevention is impeding or thwarting a potential security violation by deploying a countermeasure.
- Detection is determining that a security violation is impending, is in progress, or has recently occurred, and thus make it possible to reduce the potential loss.
- Recovery is restoring a normal state of system operation by compensating for a security violation, possibly by eliminating or repairing its effects.
- Correction is changing a security architecture to eliminate or reduce the risk of re-occurrence of a security violation or threat consequence, such as by eliminating a vulnerability.

From the above definitions, we notice that resiliency deals with recovery and correction. That is, resiliency comes into existence after detecting an attack [40]

[42]. Resiliency is launched after a security breach is detected. Some sub-areas of resiliency include positive network control, threat mitigation and incident handling and forensics [43]. In [44], authors classify some resilience techniques as proactive and reactive. Some of proactive techniques are segmentation, isolation, randomness, and distribution. While reactive techniques include deception, dynamic reconfiguration and dynamic composition [44].

As a summary of the subsection, we leave the reader with a summary as illustrated in Table 1.1. This Table maps protective measures to the adaptive security life cycle.

Table 1.1: Mapping protective measures to the adaptive security life cycle.

Security Protective Measure	Adaptive Security Life Cycle Stages			
	Prevent	Defect	Respond	Predict
Deterrence	✓			
Avoidance	✓			
Prevention	✓			
Detection		✓		
Recovery			✓	
Correction				✓

1.4 Motivation and Research Problem

Undoubtedly, the growing rate of security incidents and cost show that current security solutions cannot stop the sophistication and complexity of attacks [18] [37]. This is evident by the fact that virus scan programs always need to be updated. Widely used devices can be a source of complex and sophisticated attacks. For example, mobile devices can be used as an attack source, an attack target or part of an attack [45]. Furthermore, there is a vicious cycle between attackers and defenders, which is evident by the fact that virus scan programs always need to be current. This security phobia is the drive behind the efforts put forwards by academics and researchers to deal with security attacks to achieve more secure environments, and indirectly reduce big losses resulting from such attacks [46] [1] [38] [39].

Traditional security solutions are detection-oriented and rely on information coming from the attacker. Traditional prevention and detection methodologies, like deploying antivirus software, IDS/IPS and firewalls, have become less effective. These detection systems rely on history to catch attacks, but the more history an organization has enabled, the more performance degradation. Furthermore, detection systems are seen as offering a temporary solution [19] because we cannot defend against all attacks. That is, if a new attack comes, history-matching will fail and behavior monitoring success will depend on how close the new attack's behavior is to the old attacks' behavior. As a result, the giant Symantec Corporation has announced that anti-virus is dead [19].

In order to stay ahead of attacks, organizations must keep away from predictive and reactive approaches. Attack-based taxonomies classify existing attacks to inform us of possible attacks one can expect. On the other hand, defense-based taxonomies determine suitable defense solutions for specific attacks. What is needed is a passive and proactive approach that provides protection against known and unknown attacks without the need for constant patches.

In attack-centric taxonomies, the analysis is done based on the attacker perspective. For instance, those taxonomies use the attack vector which considers the goal of the attack. If the attacker targets file integrity, which is not achieved but the attacker managed to harm the availability of the system, this is not considered from the attacker point of view [18]. To countermeasure such an attack, a defender does not really know which security requirements nor which assets were the target of the attacker.

All of the above motivated us to study security from an asset perspective resulting in defense mechanisms with the following characteristics (a) asset owner view point, (b) proactive approach, and (c) defending assets.

Furthermore, an asset-based approach will clearly draw the boundaries between the various security spaces. As it is, security is spread across a spectrum of security prefixes coining security terms that include "physical security", "information security", "data security", "cyber security", and other additional terms of security. Currently, there are various definitions and interpretations to these terms that are either used interchangeably, overlapping, conflicting, or vague. Cyber security is defined in [40] as "Measures taken to protect a computer or computer system (as on the Internet) against unauthorized access or attack", while information security is defined in [29] [40] as "The protection of information and information systems from unauthorized access, use, disclosure, disruption, modification, or destruction in order to provide confidentiality, integrity, and availability". These two definitions are similar and as a result authors use information security and cyber security interchangeably. Cyber security is defined in [41] as "The protection of cyberspace itself, the electronic information, the information communication technology (ICT) that support cyberspace, and the users of cyberspace in their personal, societal and national capacity, including any of their interests, either tangible or intangible, that are vulnerable to attacks originating in cyberspace". Furthermore, terms such as "data security" and "information security" are used almost interchangeably. Data security is defined in [29] [40] as "Protection of data from unauthorized (accidental or intentional) modification, destruction, or disclosure". This definition is very similar to information security making the whole security spectrum vague in definition with no clear boundaries.

In this thesis, we argue that the failure of current security solutions is the result of the attacker leading the security game. Defenders are followers and defense systems are built by defenders based on solicited input from the attacker. 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 solutions incapable of detecting unknown attacks, and (b) provide solutions that use predictive or reactive strategies.

This thesis proposes to change the security game such that the game will be led by defenders. This is the natural way to play such a game because the defender knows exactly its assets, knows exactly its security requirements, and what are the consequences of violating these security requirements. Therefore, the defender can build a defense system based on information it owns not solicited nor given by the attacker.

1.5 Thesis Objectives

The goal of this thesis is to design and implement an asset-based security system. Specific thesis objectives are:

1. To study and classify the current security systems.
2. To study the relationship between critical assets, security requirements and security spaces.
3. To investigate and identify the strength and weakness of existing security taxonomies.
4. To design a taxonomy based on defending assets rather than defending against attacks.
5. To design and implement an asset-based security system.

1.6 Thesis Contributions

Our proposed security system has the following characteristics (a) relies only on information from the defender, (b) defending assets rather than defending against attacks, (c) proactive and (d) passive. The proposed security system relies on information supplied by the assets' owner. Therefore, our system discovers attacks based on information solicited by the defender. As such, the monitoring is not done to discover nor prevent attacks based on attack vectors nor attack behaviors. Since the proposed approach defends assets, it is generic and is not tailored to a set of attacks. Furthermore, the proposed approach is proactive that involves anticipating violations in advance of their actual occurrence and making appropriate organizational shifts in its response. Finally, our approach is passive since it works at the hypervisor-level. More importantly, it is transparent to the guest operating systems making it difficult for running processes to detect if they are being monitored.

We envision that our proposed asset-based approach will have the following contributions:

1. Draws clear boundaries between the various security spaces.

2. Enables asset defense solutions.

3. Enables proactive defense solutions.

4. Enables passive defense solutions.

1.7 Thesis Organization

There is no doubt that security issues are on the rise and defense mechanisms are becoming one of the leading subjects for academic and industry experts. In this thesis, we focus on the security domain and suggest a new way of looking at the security life cycle. We also discuss the problem statement and our motivation to solve the problem. The remainder of this thesis is organized as follows. For clarity and completeness purposes, chapter 2 outlines the existing definitions of various security spaces as well as security requirements. The literature review is outlined in chapter 3. In chapter 3, we survey the existing security requirements taxonomies, security attack taxonomies as well as security defense taxonomies. Chapters 2 and 3 lay down the ground for the proposed work in this thesis. We propose an asset-based taxonomy in chapter 4 and a methodology to achieve our goals of this thesis. Chapter 5 presents the proposed asset-based security model, while chapter 6 describes the evaluation environment needed to do the performance evaluation, and chapter 7 outlines in details the performance evaluations. Finally, chapter 8 concludes the thesis and envisions future directions.

CHAPTER 2

SECURITY SPACES

In any security domain there exists assets, security requirements, attackers and defenders. Attackers exploit vulnerabilities and target one or more security requirements of one or more assets. These attacks are mitigated by safeguards and defenses [47]. We start this chapter by defining threats, vulnerabilities, and assets. We follow up with few sub-sections looking at the various security spaces, security requirements. We also link security requirements to security spaces. We conclude this chapter by drawing our observations and remarks.

2.1 Threats, Vulnerabilities, and Assets

External or internal malicious actions usually target a quality or state that is exposed. The nature of being exposed renders a system state to be defendless. Malicious actions' goal is to harm assets and this is achieved by penetrating through system weaknesses. In this section we shed some light on these three important components of any security system, namely targets or assets, aggressive actions or threats against the target, and weaknesses or vulnerabilities.

Assets are defined in [42] as "A system resource that is (a) required to be protected by an information system's security policy, (b) intended to be protected by a countermeasure, or (c) required for a system's mission". In the content of information security, computer security and network security, an asset is defined as "any data, device, or other component of the environment that supports information-related activities. Assets generally include hardware (e.g. servers and switches), software (e.g. mission critical applications and support systems) and confidential information" [48]. Assets should be protected from illicit access, use, disclosure, alteration, destruction, and/or theft, resulting in loss to the organization. Therefore, assets are necessary for systems to achieve their functionality and there must be a security system to protect these assets.

Vulnerabilities on the other hand, are defined as "the intersection of a system susceptibility or flaw, attacker access to the flaw, and attacker capability to exploit the flaw" [48]. To exploit a vulnerability, an attacker must use applicable tool or technique that can utilize a system weakness. Therefore, vulnerabilities are also known as the attack surface. In [42] authors define vulnerability as "A flaw or weakness in a system's design, implementation, or operation and management that could be exploited to violate the system's security policy". The vulnerability in a system can be in design, specification, in implementation, or in operation and management [42].

Finally, threats are defined as "any circumstance or event with the potential to adversely impact organizational operations (including mission, functions, image, or reputation), organizational assets, or individuals through an information system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service. Also, the potential for a threat-source to successfully exploit a particular information system vulnerability" [49].

2.2 Security Spaces

2.2.1 Physical Security Space

One of the most essential and prominent security spaces is physical security space. Physical security is defined in [50] as "The protection of personnel, hardware, programs, networks, and data from physical circumstances and events that could cause serious losses or damage to an enterprise, agency, or institution. This includes protection from fire, natural disasters, burglary, theft, vandalism, and terrorism". Also, physical security is defined in [51] as "The protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability, and confidentiality of information system resources (includes hardware, software, firmware, information, data, and telecommunications)". Another definition of physical security [52] is "That part of security concerned with physical measures designed to safeguard personnel; to prevent unauthorized access to equipment, installations, material, and documents; and to safeguard against espionage, sabotage, damage, and theft".

From these definitions, we notice that despite what is the target, if there is a physical contact then the whole attack fits in the physical security space. An example is given in [32] where a program called “Private Eye” was designed to handle eavesdroppers and physical security threats by applying screen protection to identify legitimate users. If the user is not identified, then the monitor gets blurry to protect data and the intruder photo is captured. Other examples representing physical attacks [51] are the cases of losing laptops and the other physical devices.

Table 2.1 shows the threats, vulnerabilities, and assets involved in the physical space. Threats are various and can be cutting network cables or stealing a laptop. Vulnerabilities must be physical. Examples of such vulnerabilities are malfunctioning monitoring system or bypassing security guards. Finally, the goal of a physical attack is to target a physical entity that can be a human, building, or a device.

Table 2.1: Physical security space.

Security space	Component		
	Threats	Vulnerabilities	Assets
Physical	Various	Physical circumstances	Physical entity

2.2.2 Information and Communication Technology Security Space

Before defining information and communication technology security space, we have to differentiate between information and information technology. In [53] information is defined as “information can take on many forms. It can be printed or written on paper, stored electronically, transmitted by post or electronic means, shown on films, conveyed in conversation, and so forth”. It seems that there are two parts of information namely, electronic and non-electronic. We will refer to electronic information as e-information, and t-information to refer to non-electronic information or traditional information.

E-information must be stored, transmitted, and processed by technology, this is called information communication technology (e-ICT) as defined in [53]. On the other hand, storing or transmitting t-information, traditional technology or non-electronic technology must be used, which will refer to as t-information communication technology (t-ICT). So whenever we use ICT security space we mean the space in general and when we need to distinguish between them we will use either e-ICT or t-ICT.

ICT is defined as “all aspects relating to defining, achieving and maintaining the confidentiality, integrity, availability, non-repudiation, accountability, authenticity, and reliability of information resources” [53]. It should be noted that the protection of information can be extended to the underlying information resources which is ICT, and that information security depends mostly on ICT security. An example of ICT security space attack is given in [9] [20], when a virus sabotaged 30,000 workstations and destroyed hard disks. Another example of ICT security space attack is when an attacker gains access to a system because of insufficient authentication, insufficient validation, or insufficient password strength. The attacker here can take control of the system or initiate another attack vector.

Table 2.2 shows both e-ICT and t-ICT security spaces and draws the difference between these two security spaces using threats, vulnerabilities and assets. Assets are the targets need to be protected from various threats exploiting vulnerabilities.

Table 2.2: ICT security space.

Security space	Component		
	Threats	Vulnerabilities	Assets
t-ICT	Various	t-ICT	t-ICT
e-ICT	Various	e-ICT	e-ICT

2.2.3 Information Security Space

Information is composed of two parts e-information and t-information with their own vulnerabilities. E-information vulnerability come from e-ICT, and similarly t-information vulnerability comes from t-ICT. Table 2.3 shows the threats, vulnerabilities, and assets of the information security space.

Table 2.3: Information security space.

Security space	Component		
	Threats	Vulnerabilities	Assets
Information	Various	t-ICT, e-ICT	Information

Information security space is ensuring safety and protection of information from illegal access. Authors in [53] defined information security as "the protection of information and its critical elements, including the systems and hardware that use, store, and transmit that information". It is obvious that systems and infrastructures play an important role in the information security process. According to [8], May 13th 2011 is the date of the 2nd largest online data breach in the US. This breach is an information security space attack where attackers exploited computer systems through vulnerabilities such as misconfiguration, kernel flaws, design flaws or buffer overflow [8] [54] [55].

2.2.4 Cyber Security Space

As we mentioned previously users misuse the term of cyber security and use it interchangeably with information security. In [53] cyber security is defined as “the protection of cyberspace itself, the electronic information, the ICTs that support cyberspace, and the users of cyberspace in their personal, societal and national capacity, including any of their interests, either tangible or intangible, that are vulnerable to attacks originating in cyberspace”. Another definition of cyber securityis stated in [56] as “prevention of damage to, protection of, and restoration of computers, electronic communications systems, electronic communication services, wire communication, and electronic communication, including information contained therein, to ensure its availability, integrity, authentication, confidentiality, and nonrepudiation”.

Examples of cyber security attacks [8] include hacking Dropbox facility in July 2012. Usernames and passwords were stolen and used to access the Dropbox accounts. As a result, attackers start bullying Dropbox users causing what is known as cyber terrorism, cybercrime, or cyber espionage [57]. Cyber security space is considered a complex space because it intersects with many other security spaces such as ICT and information. Table 2.4 shows components of a cyber security attack.

Table 2.4: Cyber security space.

Security space	Component		
	Threats	Vulnerabilities	Assets
Cyber	Various	e-information, e-ICT	Human

After we clarify the boundaries of the different security spaces, we can see that penetrating the physical security space can lead to all other security spaces enabling the attacker to skip some countermeasures in other security spaces. The attacker later can start a new attack vector in new security space. As such, the first step always comes from physical or ICT security spaces.

In Table 2.5, we summarize the security spaces. From the Table, assets and their vulnerabilities must be considered in order to fit an attack to a security space. Assets alone or vulnerabilities alone cannot correctly classify attacks according to their security space. This shows us that classifying attacks from defenders point-view is impossible since vulnerabilities are not known to defenders. As such, defense systems are seriously hindered by the lack of information about attacks [58]. This also shows us the importance of assets in mitigating attacks.

Table 2.5: Security spaces.

Space	Threats	Vulnerabilities	Assets
Physical Security	Various	Physical circumstances	Physical entity
e-ICT Security	Various	e-ICT	e-ICT
t-ICT Security	Various	t-ICT	t-ICT
Information Security	Various	e-ICT, t-ICT	Information
Cyber Security	Various	e-ICT, e-information	Humans

2.3 Security Requirements

Security policies and priorities have become complicated, often ambiguous, and even inconsistent; not because of immediate threats but rather the unpredictable, uncertain, and blurring requirements of the security arena [59]. Furthermore, it is becoming an intricate puzzle for security engineers and architects to develop meaningful and realistic secure environments. Many taxonomies are introduced to deal with a single attack, and most of them fail to handle blended attacks [60]. A taxonomy proposed [54], known as AVOIDIT, managed to classify blended attacks. In [59], a holistic security requirement taxonomy was proposed were authors surveyed many security requirement papers and classified them in basic categories.

2.3.1 Confidentiality

Confidentiality is defined in information security as "is the property, that information is not made available or disclosed to unauthorized individuals, entities, or processes". [61]. Confidentiality might seem similar to "privacy" but in fact the two terms are different. Rather, confidentially is a component of privacy that implements to protect our data from unauthorized viewers. Examples of confidentiality of electronic data being compromised include laptop theft, password theft, or sensitive emails being sent to the incorrect individuals [62].

Therefore, confidentiality is basically a set of rules that limits access to information. Formally, [63] defines confidentiality as "the prevention of unauthorized disclosure of information". Another definition is given in [64] "the assurance that information is not disclosed to unauthorized persons, processes or device".

2.3.2 Integrity

Integrity is defined in [63] as "the prevention of unauthorized modification of information". In [64], authors define integrity as "the quality of an information system reflecting logical correctness and reliability of an operating system; the logical completeness of the hardware and software implementing the protection mechanisms; and the consistency of the data structures and occurrence of the stored data". Another definition of integrity [65] states "quality of an information system reflecting the logical correctness and reliability of the operating system, the logical completeness of the hardware and software implementing the protection mechanisms, and the consistency of the data structures and occurrence of the stored data". Yet another definition to data integrity is "the state that exists when computerized data is the same as that in the source documents and has not been exposed to accidental or malicious alteration or destruction. The property that data has not been exposed to accidental or malicious alteration or destruction" [65].

2.3.3 Availability

Availability means expecting to find the entity when the user needs it. In [42], it is defined as "the property of a system or a system resource being accessible, or usable or operational upon demand, by an authorized system entity, according to performance specifications for the system; i.e., a system is available if it provides services according to the system design whenever users request them, sure this property is a different than reliability". Another definition of availability is given in [63] is "the prevention of unauthorized withholding of information". Yet another definition to availability [64] is "the timely, reliable access to data and information services for authorized user".

2.3.4 Access Control

Access control is a vital step in forcing security. In [42], it is defined as "protection of system resources against unauthorized access". The same paper gave another definition to access control as "a process by which use of system resources is regulated according to a security policy and is permitted only by authorized entities (users, programs, processes, or other systems) according to that policy". A third definition provided in [59] as "limitations on interactions between subjects and objects in an information system". Authors in [59] divided access control into identification, authentication, and authorization.

Other researchers [42] define authentication as composing of two steps namely identification and verification. Authentication is defined as "the process of verifying a claim that a system entity or system resource has a certain attribute value" [42]. Another definition [64] to authentication is "security service designed to establish the validity of a transmission, message, or originator, or a means of verifying an individual's authorizations to receive specific categories of information".

Identification means to recognize a user by the system. In [42], identification is defined as "an act or process that presents an identifier to a system so that the system can recognize a system entity and distinguish it from other entities".

Verification as stated in [42] as "the process of examining information to establish the truth of a claimed fact or value". Also in [42], verification is defined as "the process of comparing two levels of system specification for proper correspondence, such as comparing a security model with a top-level specification, a top-level specification with source code, or source code with object code". As such, verification is presenting authentication information that acts as evidence. This evidence proves the binding between the attribute and that for which it is claimed. Authorization is giving permission or privilege to users after being authenticated. Basically, authorization is an approval granted to an entity to access resource.

2.3.5 Non-Repudiation

In [42], non-repudiation is defined as "a security service that provide protection against false denial of involvement in an association (especially a communication association that transfers data)". For example, two separate types of denial are possible. An entity can deny that it sent a data object, or it can deny that it received a data object. Therefore, two separate types of non-repudiation service are possible. Non-repudiation is also defined in [64] as “the assurance the sender of the data is provided with proof of delivery and the recipient is provided with proof of the sender's identity, so neither can later deny having processed the data”. Other researchers [66] define non-repudiation as “prevention of attacks done by one of the two parties in the communication, the sender or the receiver, one of them later may deny that he has sent or received the message. Like when a bank customer asking his bank to send some money to a third party but later denying he has made such request”.

2.3.6 Accountability

Accountability means the responsibility of one’s actions. Formally [65] defines accountability as “assignment of a document control number, including copy number, which is used to establish individual responsibility for the document and permits traceability and disposition of the document”. Accountability is also defined in [42] as “the property of a system or system resource that ensures that the actions of system entity may be traced uniquely to that entity, which can then be held responsible for its actions”. Therefore, to offer accountability, a system should positively associate users’ identities with the method and time access. That way, detection and subsequent security investigation can be done. Users are held accountable for their actions after being notified of their behavior for abusing the system. Proper consequences should be associated (with such abuse) and enforced.

2.3.7 Anonymity

The term “anonymous message” typically means the sender and/or the receiver of the message are not revealed. In many countries, anonymous messages are protected by law and must be delivered as regular messages. Being unknown is the goal when achieving anonymity. Anonymity is defined in [42] as “the condition of an identity being unknown or concealed”. To achieve this definition and to maintain the service at the same time, a third party must hide user information.

2.4 Security Spaces and Requirements

This section studies the relationships between the security requirements and the security spaces. Table 2.6 answers the following question: In which security spaces a certain security requirement is achievable. The answer to this question tells us which security spaces we should consider to provide a holistic security solution. For example, if our security system needs to provide a holistic availability solution for a certain asset, in which security spaces should availability be provided. In Table 2.6, if the security requirement is achievable in the corresponding security space, the table entry is marked **Y**. Otherwise, the table entry is marked **N**.

To elaborate on Table 2.6, we provide an explanation for each row in the table by picking each security requirement and illustrate its mapping to the different security spaces.

Access Control: In the physical security space, access control security requirement is achievable by physical barriers such as doors, walls, or security gates.

Table 2.6: Mapping security requirements to security spaces.

Security requirements	Spaces			
	Physical security	Information security	e-ICT security	t-ICT sec security
Access Control	Y	Y	Y	Y
Availability	Y	Y	Y	Y
Integrity	N	Y	Y	Y
Confidentiality	N	Y	N	N
Anonymity	N	Y	N	N
Non-Repudiation	N	N	Y	Y
Accountability	N	N	Y	Y

In [67], security gates are used to achieve physical access control. Furthermore, in the information security space, passwords are used on the file level (i.e., information level) and this will achieve information access control as done in [68]. Users can also easily set passwords on their web pages and these web pages are considered files. Considering e-ICT security space, access control can be achieved on different e-ICT resources such as wireless networks, PCs, and Servers. Again, this e-ICT access control can be achieved simply through using passwords. In [69], e-ICT access control is applied to computers or networks. Finally, t-ICT controls access to information using traditional techniques such as single-factor authentication by using secure keywords.

Availability: Availability in physical security space is applicable. To achieve availability in physical space, we can apply a solution to insure physical availability for physical entities such as network cables for instance. In [67], video surveillance systems are introduced to maintain availability. This can be considered as a combination of both e-ICT and physical security spaces. Physical security guards, instead of cameras, can be used to achieve availability at the physical security space. We can also achieve availability by performing backups and redundancy as done in [70]. In [70], authors introduced high availability systems for steam processing. Top companies also provide products such as “IBM DB2 HADR” (High Availability Disaster Recovery), “Oracle Data Guard” and “Microsoft SQL Server 2005 Database Mirroring”. Replication is also used to achieve availability as in [71] where HADOOP is introduced to achieve high availability through metadata replication. E-ICT availability solutions can be achieved by: (a) producing reliable e-ICT to persist attacks and maintain availability, (b) having reliable servers, or (c) constructing reliable networks immune to disturbances from attacks [72]. For t-ICT security space, availability is achievable by manufacturing and designing technology which can maintain availability for stored information. This technology can be, for example, producing papers which can survive and maintain information for long periods of time.

Integrity: Integrity is achievable in information, e-ICT, and t-ICT security spaces, but not in physical security spaces. In information security space, we can have this by creating backups on information level [70], hashing algorithms as done in [73] [74] such as MD5, HMAC, or SHA1 to calculate hashes to verify integrity. For e-ICT security space, integrity can be achieved by fault tolerance algorithms either hardware or software [75]. An approach was introduced in [75] to tolerate various malicious code modifications and transient-faults during run time of a computing application system. Furthermore a security solution for data integrity in wireless bio sensor networks was introduced in [76]. For t-ICT security space, handwriting analysis and recognition techniques can be used to achieve integrity.

Confidentiality: There is no solution to apply confidentiality at the e-ICT, nor the t-ICT, nor the physical security spaces. Confidentiality is achievable in the information security space and techniques such as steganography and encryption can be used to achieve confidentiality at the information security space. Steganography as mentioned in [77] is used to conceal the fact that a secret is sent inside a message. Traditional methods like using invisible ink in communication [66] is another way to achieve confidentiality. Finally, encryption also can achieve confidentiality by protecting message contents.

Anonymity: To have a holistic anonymity security requirement solution, we can have it on information security space in [78] they propose an anonymous routing protocol to wireless network, this is similar to onion routing concept used in wired networks, also VPN can have solution to anonymity security requirement. Another solution is to have third party for exchanging information [66], aliasing also can achieve this solution by using identified number instead of name, like in [42] a financial institution may assign account numbers, so transactions can remain relatively anonymous with the transactions accepted as legitimate. There is no solution to apply anonymity security requirement at the e-ICT, t-ICT, and physical security spaces.

Non-Repudiation: to have information non-repudiation security solution in t-ICT security space registered mail used as solution [79], this technology evolved to become digital signature in e-ICT security space, experts found that even digital signature is not enough so they invent capturing unique biometric information and other data about the sender [79], another solution as in [80] they introduce a fair non-repudiation protocol that requires a trusted third party. There is no solution to apply non-repudiation security requirement at the information security space nor at physical security space.

Accountability: to have a holistic accountability security requirement solution in e-ICT security space, specifying transactions for which you want more than one approval, and authorize different levels of account access as in [81] they achieve accountability in their online business, for t-ICT security space, it can be done by feeding users with different contradicting information to reveal the spoke man. There is no solution to apply accountability security requirement at the physical security space nor at information security space.

2.5 Security Requirements Classification

Now, let us classify the security requirements based on attack type and attack target as shown in Table 2.7. Authors in [66] classified attack types to either active or passive. They described an active attack as an attack that may change the data or harm the system as such attacks threatening integrity or availability as active attacks. They also mentioned that in passive attacks, the attacker's goal is just to obtain information and therefore, the attacker does not modify data or harm the system. The system continues its normal operation. However, the attack may harm the sender or the receiver.

From this classification done by [66], we added the attack target. For example, if the assets are the target, then the attackers need to use the assets to achieve their goal. That is, active attacks such as "Availability" target assets as we say that this is an attack on the availability of an asset. On the other hand, passive attacks such as "Anonymity" target either the sender or the receiver and not the asset. For example, "anonymity" is applied to either the sender or the receiver. Unlinkability of sender and receiver also is a way of implementing "Anonymity" which means that though the sender and receiver can each be identified as participating in some communication, they cannot be identified as communicating with each other.

Table 2.7: Classification of security requirements.

Security requirements	Attack Attributes	
	Type	Target
Access control	Active	Asset
Availability	Active	Asset
Integrity	Active	Asset
Confidentiality	Passive	Asset
Anonymity	Passive	Sender/Receiver
Non-repudiation	Passive	Sender/Receiver
Accountability	Passive	Sender/Receiver

2.6 Observations and Remarks

Damage can be done by either internal or external events and these events are basically threats. Threats exploit weakness to perform unauthorized actions within a system. Threats target to damage assets and the damage intensity is correlated with how critical is the asset. These three components (threats, vulnerabilities, assets) are played by different actors in different security spaces as explained in this chapter. Observing this, we can draw the following remarks:

- To classify attacks, one needs to consider assets and vulnerabilities. This consideration is impossible since vulnerabilities are not known to the defender. Therefore, defenders always have incomplete information to establish a defense system.

- For threats and vulnerabilities, attackers have the upper hand since they themselves decide on the attack vector and on the vulnerability to gain access to the system in order to deliver a malicious outcome (i.e., damaging the asset).

- The goal of the attack is to inflict harm or compromise the security requirements of the targeted asset. As such, the attacker needs to learn the critical assets. But since the defender owns these assets, the defender controls this learning phase. That is why zero-day ransomware attacks encrypt all files hoping that a subset of these encrypted files are important for the owner. If zero-day ransomware attacks know the critical files, they will not waste time and effort on other non-critical files.

- Focusing on assets will prevent an attacker from achieving the attack goal. This is evident from Table 2.7. Protecting assets will render attacks useless since this will prevent all attacks violating the CIA triad of confidentiality, integrity, and availability known as the heart of information security.

CHAPTER 3

LITERATURE REVIEW

Researches started introducing requirements-based taxonomies [82] [83] [84] [64] [85] [63] [86] [87] [88] [60] [59] to gain knowledge of the needs to achieve secure systems and continued to understand and analyze the existing attacks [89] [58] [55] [54] [90] [91] to gain knowledge of the behavior and the damage caused by such attacks. Finally, researchers shifted to defense-based taxonomies [58] [92] [91] [93] [94] [95] [18] to countermeasure a specific attack with an effective defense mechanism.

The attack-based taxonomies are widely researched whereas the defense-based taxonomies are hindered by several serious facts [58], lack of detailed information about attack information, lack of benchmarks, and difficulty of large-scale testing in defense systems [58].

In a nutshell, attack-based taxonomies illustrate the process of classifying attacks and enables administrators to gain common security knowledge to become alert in defending when attacks are detected [54]. On the other hand, defense-based taxonomies help in determining the suitable defense mechanism.

3.1 Existing Security Taxonomies

3.1.1 Requirements-Based Taxonomy

One of the well-known requirements-based taxonomies is mentioned in [82] [83] and is referred to as Confidentiality-Integrity-Availability (CIA) triad model. CIA is referred to as at the heart of information security. Later, other taxonomies added more security requirements to CIA, arguing that CIA is not sufficient.

In [84], authors reintroduced a model known as "Parkerian hexad" which was introduced in 2002 by Donn B. Parker. This model is a set of 6 information security requirements. Adding possession, authenticity and utility to the CIA triad model.

In [85], authors introduced Access Control, Authentication, and Accounting (AAA) triad model. AAA is the cornerstone of any systematic discipline of security [85]. Other researchers in [96] [97] introduced security requirements for software systems representing the basic security policy needed in order to protect software system. It has 8 security requirements namely, fair exchange, freshness, secure information flow, guarded access, role base access control, authenticity, secrecy and integrity, and non-repudiation [96] [97].

A detailed quality model for safety, security, and survivability engineering was introduced in [86]. This model describes relationships between concepts that contribute to systemic qualities and decomposed some of the security requirements such as access control to (identification, authentication, authorization), integrity to (data integrity, hardware integrity, personnel integrity, software integrity), and privacy to (anonymity, confidentiality).

An Accelerated Requirements Method (ARM) [87] groups security requirements and utilizes a structured categorization technique to group and name security requirements. In this paper, authors defined six groups, each contains one to four security requirements. namely they are confidentiality, access control, data integrity, manageability, usability, and authentication.

A taxonomy of software security requirements [88] proposed two levels of security requirements. The first level includes integrity, availability, confidentiality and non-repudiation. The second level redefines each of the first level security requirements into more specific terms. For example, availability is refined as (response time requirements, expiration requirements, and resource allocation requirements) [88].

A taxonomy proposed in [60] classifying security requirements as confidentiality, integrity, availability, accountability, and conformance. Each of these requirements branches into sub-categories [60]. A holistic taxonomy of security requirements proposed in [59]. Authors examined similarities and differences between all previously published security requirements classifying them all into two levels, namely basic and cofactor levels. In this paper, security requirements were classified into 13 basic requirements. Each one of these basic 13 security requirements has from 1 to 8 cofactors. For example, privacy is one of the basic requirements and it has 8 cofactors listed as trace, cardinality, content and notification, attribution, aggregation, encryption, confidentiality and anonymity [59].

3.1.2 Attack-Based Taxonomy

Continuing the efforts of classifying and defining security requirements, research efforts are also channeled towards classifying and understanding attacks. Researchers in [89] provided an incident taxonomy based on the attack classification by events. These attacks are analyzed to have the following steps: target, vulnerability, action, tools and unauthorized result. These steps basically determine the attacks directed at a specific target of the attacker. In order for the attacker to reach the target, a specific vulnerability must be utilized resulting in a changed state. This gives a whole picture of all the steps involved in an attack and how an attack grows [89].

A comprehensive taxonomy of attacks targeting availability was introduced in [58]. The aim was to classify attack strategies and list attributes of attack strategies that are essential in developing countermeasures classified by possibility of characterization, attack rate dynamics, degree of automation, source address validity, exploited weakness, victim type, persistent agent set, and impact on victim. A list of attacks were mapped to specific countermeasure and security requirement [55].

A group of academics at Memphis University [54] introduced “AVOIDIT”, a cyber-attack taxonomy describing the nature of an attack using 5 major classification: attack vector, defense, operational impact, informational impact, and attack target. Classification by the defense mechanism to provide information to the system administrator concerning attack remediation or mitigation policies. Mitigation includes action such as removing from network, whitelisting, or referencing advertisements. Remediation includes system patching and code correction. Attack targets might be the OS, the network, a process, or data. This taxonomy lacks defense strategies and cannot deal with physical attacks such as the ones initiated by USB drives [54]. AVOIDIT is able to efficiently categorize mixed attacks.

According to [90], an extensive taxonomy for computer network attacks was explored. This taxonomy introduced 4 hierarchical levels and succeeded to include attackers and defenders. A taxonomy for security threats in emergency management was discussed in [91]. Authors classified attacks by three types: network type, function affected, and attack factor. They examined SMS flooding attacks cellular network, public safety mobile network issue, GPS spoofing attacks in satellite systems, and cyber threats in wired networks. Authors also examined five affected functions, which are: detection of emergencies, planning of operation, transportation, medical service and communication with the public. From the attack vector point of view, authors examined at network misuse (vulnerability of nodes, masquerading, flooding), and software misuse (executed remotely and locally). In [98], researchers introduced attack countermeasures used for security analysis. These attack countermeasures become a vital factor when analyzing the system from the security perspective.

3.1.3 Defense-Based Taxonomy

In [58], a defense taxonomy of mechanisms for Distributed Denial of Services (DDoS) was introduced. Three attributes were taken into consideration to classify defense strategies. These attributes are cooperation degree (autonomous, cooperative or interdependent), activity level (preventive or reactive), and deployment location (victim network, intermediate network or source network).

A defense-centric taxonomy was introduced by [92]. This taxonomy is based on attack manifestations. The manifestations depend on comprising sequences of system calls. This sequence is generated from the activity or presence of an attack. Four classes were introduced in the taxonomy: manifestation by foreign symbol, manifestation by minimal foreign sequence, manifestation by dormant sequences and manifestation by being anomalous [92].

A security threats taxonomy was introduced in [91]. This classification contains three categories: defense type, degree of distribution and organizational element. Organizational element is further branched to system, process and human, while by defense type is divided to preventive (authentication, resilience and self-awareness) and reactive (detection and response).

A reliable defense framework was proposed in [93]. In this framework, authors used countermeasures as well as attacks to recommend an efficient and reliable defense mechanism. Authors assess multi-step attack damages to identify corresponding defense countermeasures in order to mitigate service downtime.

Exploring intrusion detection systems to reduce infiltration done by attackers [94], researchers introduced a taxonomy for intrusion response system and intrusion detection system classifying defenses by: response cost, level of automation, response time, adjustment ability. The same group of researchers [95] introduced a security risk assessment taxonomy adding to their previous work risk assessment as a defense classification attribute.

Authors in [18] explored a defense-centric attack metric, neglecting the effect of ambiguous vulnerability and uncovered attacks, to evaluate the damage done to critical assets by ranking intrusion detection system (IDS) alerts in an automatic manner. This evaluation process depends on a graph connecting assets to consequences for each of the system requirements.

3.2 Recent Development

3.2.1 Zero-Day Attacks

Authors in [99] employ several data mining techniques 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.

A behavior-based scheme was proposed in [100] to spot zero-day android malware. Before releasing android applications into the public domain, the work done in [100] automatically monitors dangerous behaviors of such applications to warn the users of zero-day attacks such as launching roots exploit or sending background SMS messages.

In [101], a machine learning framework is proposed to detect known and newly emerging network attacks using layer three and four data flow characteristics. The framework depends on a supervised classification in detecting known classes and adapts the unsupervised learning phase to detect new classes.

A survey to classify zero-day polymorphic worm detection techniques is done in [35]. Detection techniques survey three signatures detection techniques, namely content-based, semantic-based, and vulnerability-driven. In addition to the signatures detection techniques, authors use statistical-based and behavior-based detection techniques.

Researchers in [102] introduce a metric to rank safety from zero-day attacks by counting how many such vulnerabilities would be required before compromising network assets. The algorithm used assumes insider attackers and gives the same weight to all zero-day vulnerabilities.

3.2.2 Ransomware Attacks

The recent zero-day ransomware attacks also earned the attention of the research arena. Authors in [103] introduced R-Locker to countermeasure zero-day ransomware attacks. A honeyfile is created and acts as a trap to elude and minimize the damage done to real assets. Various links are added to the honeyfile 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.

The zero-day ransomware anomaly detection approach introduced in [104] was highlighted by [105] where I/O operations are analyzed and a sequence of I/O requests is obtained. If the obtained sequence matches a known ransomware sequence, then an alarm is raised. A known ransomware sequence looks like (a) read the file, (b) encrypt the file, and (c) replace the original data by the encrypted data. The authors went even further and compared screen shots to detect screen locker ransomware and also extracted some words from the screen shots to be analyzed and examined.

A survey is conducted in [105] to pinpoint ransomware success factors. This survey found that reasons behind the spreading of ransomware attacks and their success are not the techniques used by the ransomware itself or unknown-nature of zero-day ransomware attacks. Rather 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 [106] 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 in particular.

Android ransomware attacks were the focus in [107]. A large-scale of 2,721 Android ransomware samples were collected and characterized to insure the majority of existing Android malware are covered and reflected in the sample. The paper proposed RansomProber, a real-time behavoir-based ransomware detection system. Evaluation experiments were 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 a number of commercial solutions with a detection accuracy of 99%.

In [107], authors focus in specific kind of ransomware which is related to Android ransomware they collect 2,721 samples of them, they notice that existing anti-virus are useless, so they propose RansomProber which is real-time detection system. They study and analyze the ransomware according to some feature which are :(1) lock screen (2) encrypt file (3) permission uses (4) payment method (5) threatening message. They focus on encrypting ransomware with assumption that ransomware does not elevate privileges, also it is easy to defeat any real-time protection system, and due to malware scanners can detect root exploits, ransomware authors avoid retrieving root privileges which is easy to be done by some root-kit tools. Finally, they assume that the early alarm can reduce the number of encrypted file. RansomProber can detect if the users initiate the file encryption operations by analyzing if there is encryption done to any file, then they check whether the encryption is normal or abnormal operation by doing foreground analysis, then they check user interface widgets which doesn't exist in ransomware sample like: (1) file list to be encrypted which is selected and partial in benign application while in ransomware case it is random and full (2) hint text must shown when you deal with sensitive behavior as encryption, this hint text doesn't exist in ransomware case (3) button which enable the user to interact with encryption process in benign application and it is not used in ransomware case. From all mentioned RansomProber considered to be behavioral based. RansomProber shown to have high accuracy and acceptable runtime performance when detecting encryption done by ransomware through experimental results.

3.2.3 Moving Target Defense

A group of academics [36] [108] introduce a moving target defense (MTD). The MTD is the concept of morphing the target, making it unfamiliar to the attacker [109] [110] [111] [112]. 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.

Since declaring MTD, many researchers adopt it in so many fields in [111] MTD to tuned to deal with stealthy botnets, MTD is needed due to information gained by stealthy botnets by knowing the target network's topology then discovering the location of detectors and avoiding them by selecting path free detectors. An MTD approach proposed in [111] with periodically changing the placement of detectors, making it harder for attacker to compromise hosts and used it as proxies. Experiments done to show that the new approach can effectively reduce the stealthiness of botnets. by comparing traffic flow from MTD points containing data exfiltration by botnets with benign users to detect suspicious flows. Deploying MTD costs the defender, some increase in overhead, this overhead can be controlled by configuring MTD. For example the higher frequency of reconfiguration resulting in increased cost with better security.

Another deployment of MTD comes in smart grids, when hidden MTD approach in [112] was proposed to avoid being detected by the attackers while maintain power flows of the grid.

Due to that passive defense approach usually let the attacker has more knowledge about the defender a solution using MTD [109] placed in protecting a critical resource in a network, so that the information asymmetry is reversed, by proposing "Bayesian Stackelberg" to model this game between the leader who is the defender and the follower who is the attacker. The defender adopts a MTD scheme to thwart attacker strategy. The strategic attacker can watch the defender's movements and then act in a rational way. In addition to attacker and defender there is a critical resource and a fully connected network. The two-player game begins between defender and attacker with the attacker goal is to maximize its payoff by reaching the resource, while the defender goal is to protect the resource with minimal cost.

IT systems using clouds also has it share from MTD as in [110], by applying MTD to cloud-Based IT knowing that MTD core idea is to make a proactive defending system to eliminate the asymmetric advantage of attacker time. The challenge was to adapt MTD to as system with complexity and number of dependencies within components in IT system without impacting the system performance severely or breaking it.

CHAPTER 4

ASSET-BASED SECURITY SYSTEM

The realization of assets importance in security systems is gaining popularity. In one of the largest cyber security summits and particularly during the European Information Security Summit 2016 [113], Will Brandon, Chief Information Security Officer (CISO) at the Bank of England, stressed on the identification of critical processes and the understanding of assets. The CISO stated that organizations should know their critical assets and critical processes. Furthermore, the CISO stressed that organizations should have a way of understanding their assets and score them against the financial impact, against the reputational and operational impact [113] [114].

This chapter proposes an asset-based security system starting with a comparison of security taxonomies in 4.1. Section 4.2 proposes an asset-based taxonomy and finally section 4.5 outlines our proposed asset-based security system.

4.1 Comparing Security Taxonomies

Table 4.1 differentiates between security taxonomies based on some attributes. Attack-based taxonomies are generated from attackers view point. The strategy behind building attack-based taxonomies is to predict the attack behavior in order to detect the attack and become more familiar with it. As such, the goal for attack-based security taxonomies is to classify attacks. Defense-based taxonomies are used to defend against attacks and therefore they react to attacks in order to identify attacks and deal with them. The goal of defense-based taxonomies is to guide security practitioners of how to defend against specific attacks. Requirements-based taxonomies are used by security experts as knowledge base. They are established by security experts and their goal is to establish standards and spread security knowledge. This knowledge is used by security experts to provide security solutions or to establish new security taxonomies. What is missing is a security taxonomy that is built based on owner or stakeholder of the attack target (i.e., the asset).

Our vision of an asset-based taxonomy is centered on the asset owner. A security solution is established and its goal is to defend asset and not defend against attacks. Therefore, the defense strategy is being proactive as opposed to predict or wait and react.

Table 4.1: Comparison of security taxonomies.

Security Taxonomy Based On	View Point	Strategy	Goal
Attacks	Attacker	Predictive	Classify Attacks
Defenses	Defender	Reactive	Defend Against Attacks
Requirements	Security Expert	None	Knowledge Base
Assets	Owner	Proactive	Defends Assets

4.2 Asset-Based Taxonomy

Our proposed asset-based taxonomy builds a comprehensive organization system for asset-based security solutions. After analyzing the assets in each system and the security requirements for it, we will introduce a taxonomy depend on both, this taxonomy can be viewed in Figure 4.1.

In this taxonomy the asset will be classified to categories, the asset will be classified under each category by choice/s. This classification clarifies the asset owner needs, later will be help in making the defense holistic and complete. the categories are (1) Type of the asset which can be hardware containing valuable containment, a t-information revealing secrets communicated or stored using traditional ways ,an e-information contained in a file or any electronic form [36], human with their interests. (2) Security Space: the asset can be reached through one of the spaces, or it may intersect with that space. For example if it is in physical security space so countermeasures in that space should be looked for, it is the same case when it is ICT, information or cyber security space. (3) Security

Requirement: classify the asset regarding its security requirement which lead to better secure environment and better performance, so the security requirement can be one of the CIA (Confidentiality, Integrity, and Availability), or one of the other non CIA security requirement like anonymity, access control, accountability, etc. (4) Rank of the asset: which can be primary asset which is the asset itself or secondary asset which leads directly to the primary asset. (5) Target: the asset as target can be Stationary Target Defense (STD) or Moving Target Defense (MTD). If the asset could be shifted to be MTD that would improve the security but affect the performance as in [109], else it is considered STD.

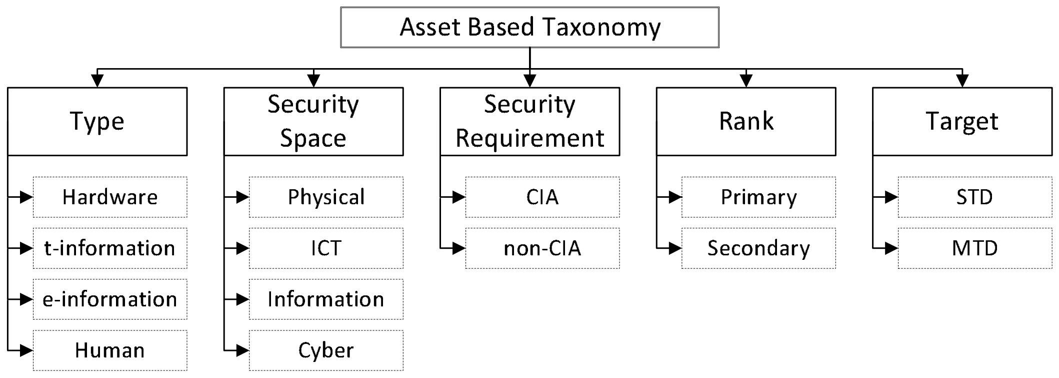


Figure 4.1: Asset-based taxonomy.

4.3 Methodology

The goal of our proposed security system, shown in Figure 4.2, is to develop defense mechanisms based on complete information. Currently, defense mechanisms are built based on incomplete information, which is dictated by attackers. In a sense, we are changing the game from an attacker-led to a defender-led game.

Current antivirus programs achieve their goal, which is detecting viruses, by

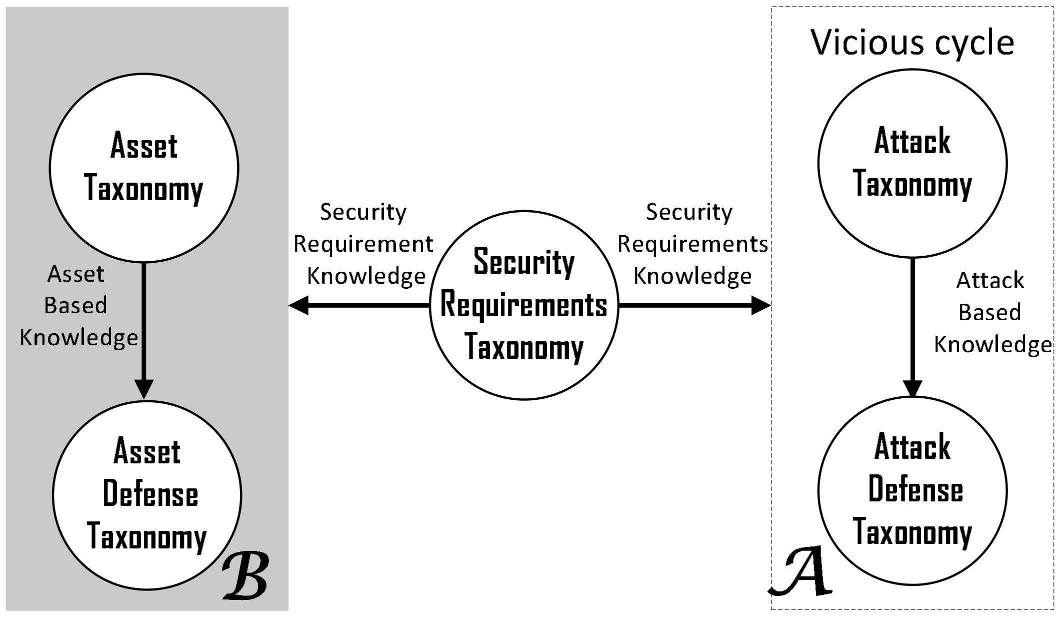


Figure 4.2: Security life cycle: (A) current cycle, (B) proposed cycle.

scanning files. The common way to do this is to use on-access scanning. When you try to open a program, the antivirus software checks the program first, comparing it to known viruses, worms, and other types of malware. The antivirus software relies on virus definitions or signatures to achieve its goal.

What we are proposing here is an approach that reaches the same goal as antivirus programs goal but not relying on a third-party information (i.e. third-party virus definitions or signatures). As shown in Figure 4.2 (A), attacks are classified based on attack vectors including attack type, exploited weakness, and victim impact. These attack vectors are established by defenders to come up with suitable defenses. As such, these defenses are defending against attacks. Figure 4.2 (B) uses an asset taxonomy to build an asset-based knowledge which contains information owned by the asset owner as compared to the attack-based knowledge containing information dictated by attackers.

First, we start by identifying the assets and the security requirements needed to protect the asset. For each of the security requirements, for example integrity, we construct a graph as shown in Figure 4.3, where assets are represented as nodes. These assets might be files, sockets, or processes. The flows between the nodes are reflected in the creation or update of edges between nodes that model the respectively involved nodes. All relevant communication directly or indirectly, imply a data flow between two nodes. During the information identification phase, files F1 and F3 are identified as critical assets with integrity as the security requirement for both files. The reachability graph is generated capturing all processes that can modify these two files. The reachability graph as illustrated in Figure 4.3 shows direct dependency as: (a) P1 can modify F1 (b) P2 and P3 can modify F3, and the indirect dependency as: (a) P1 can modify P2 and P2 modify F3 (b) P4 can modify P2 and P2 modify F3 (c) P4 can modify P3 and P3 modify F3. All of this can be generated during the monitoring phase. Now, later on if P5 tries to access F1, then this is considered a violation and a flag is passed to the decision phase.

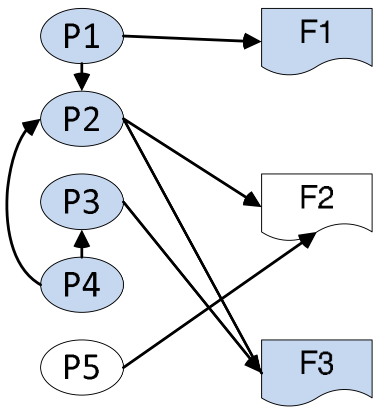


Figure 4.3: Asset Relationship Example.

Weights on these edges can be obtained by accumulating the data flows of the between nodes. It should be noted that weights can be assigned to the edges of the reachability graph. These weights can represent the access frequency, access time, access period, along with other parameters. Using the assigned weights, we can determine the probability of violating the integrity to F3 if P2 is compromised.

4.4 System Calls

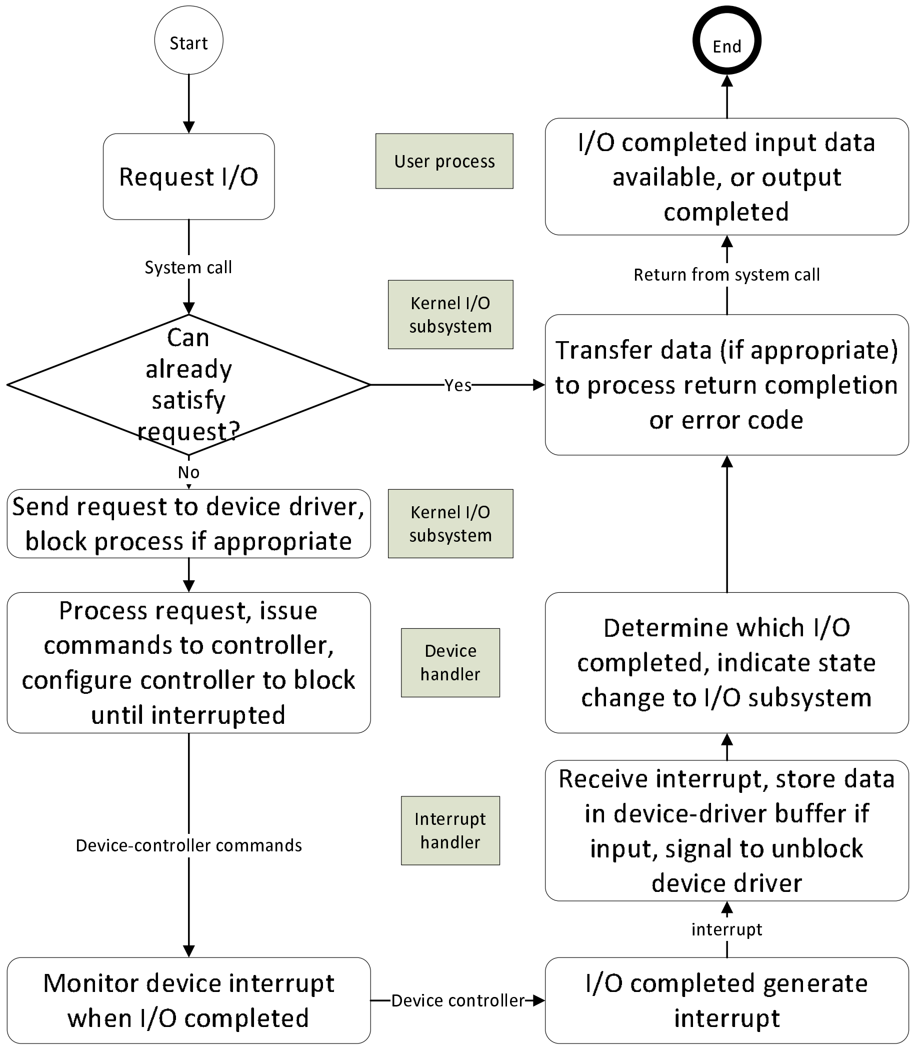
The data flow or the relationship between files or processes mentioned previously can be caught by monitoring system calls. In [2] the cycle of systems is explained and shown in Figure 4.4. The steps are shown from the system call initiation to its completeness. Let us consider that I/O request is initiated by a user process to read some data. The following are the steps needed to execute the I/O request.

- The system call code is executed in the kernel to check the parameters correctness if the block of the data needed to be read is available in the buffer cache.
- If the data is ready the block will be returned to the process and that I/O considered to be completed.
- If the block is not available then a physical I/O request must be sent to the device driver mostly by in-kernel message or subroutine call.
- The device controller receive the data in kernel buffer space, by sending command to the device controller.

- The device controller writes into the device control register, then the device controller will transfer the data by operate the device hardware.
- The driver will check for transfer completion by polling the data status or by receiving an interrupt from Direct Memory Access (DMA) controller if the driver has assigned a DMA transfer by kernel memory
- The device driver signaled by the interrupt handler.
- The device driver signal the kernel I/O with a request has been completed to proceed on with that I/O request.
- The kernel transfer data from its memory space to the user processor space.
- The second step can be executed and the I/O is completed.

4.5 Proposed Security System

As shown in Figure 4.5, our approach consists of 4 phases, namely information collection, monitoring, decision, and feedback. 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. Finally, the need for feedback phase comes into play to strength and improve the security system. For example, a prevention decision,



Start
Request I/O
User process
System call
Kernel I/O subsystem
Can already satisfy request?
Yes
No
Send request to device driver, block process if appropriate
Kernel I/O subsystem
Process request, issue commands to controller, configure controller to block until interrupted
Device handler
Device-controller commands
Interrupt handler
Monitor device interrupt when I/O completed
Device controller
I/O completed generate interrupt
interrupt
Receive interrupt, store data in device-driver buffer if input, signal to unblock device driver
Determine which I/O completed, indicate state change to I/O subsystem
Transfer data (if appropriate) to process return completion or error code
Return from system call
I/O completed input data available, or output completed
End

Figure 4.4: The life cycle of system calls. Adapted from [2].

it means that there are unusual events and therefore our security system takes these events into consideration.

4.5.1 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

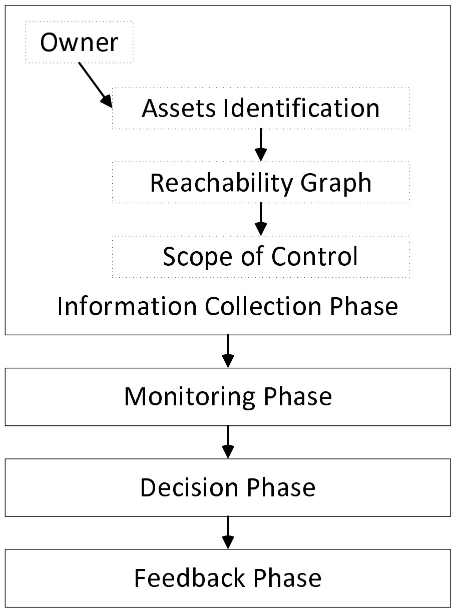


Figure 4.5: Asset based model.

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.

4.5.1.1 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 “integrity” of the Registrar database because as long as “integrity” is preserved, the University still can 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 system files representing the University Registrar database which is referred as r. In addition, the application(s) used to access the critical asset, let us say it is process p_1, needs also to be identified by the security practitioner. This means that the Registrar database identified as a critical asset can be accessed only by p_1. Hence, p_1 is the only authorized process to modify the Registrar database.

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 the purpose of simplicity and clarity, the critical assets might be represented by C which is a set of 2-tuple elements 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. Equations 4.1 and 4.2 represent C and p_r respectively.

$$C = \{(r, P_r)\} \tag{4.1}$$

$$P_r = \{integrity, \{p_1\}\} \tag{4.2}$$

As shown in Equation 4.1, C has one direct critical asset r while Equation 4.2 defines the policy of r (P_r) as only process p_1 is authorized to modify r.

4.5.1.2 Reachability Graph

Automatically capturing the low-level details, during which the interactions between files and processes are tracked in order to identify direct or indirect dependencies among all the system assets. For instance, in a database server, the administrator only needs to list the sensitive database files, and our security system later marks the process "mysqld" as critical because it is in charge of reading and modifying the databases. Such a design greatly reduces the resources and time spent by administrators in deploying our security system.

The reachability graph captures the low-level interrelationships between the critical assets identified by the user 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, the reachability graph found that process p_1 gets the information from process p_2 which reads from file f. All processes and files involved in this cycle (i.e., p_1, p_2, 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. For example if f is modified by unauthorized user, the "integrity" of r is violated.

$$C = \{(p_1, P_{p_1}), (p_2, P_{p_2}), (f, P_f)\} \tag{4.3}$$

$$P_{p_1} = P_{p_2} = P_f = \{integrity, \{\}\} \tag{4.4}$$

As shown in Equation 4.3, C has three new critical assets namely, p_1, p_2, and f. Equation 4.4 equates the policies of p_1, p_2, 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.

The low-level critical assets identified by the reachability graph will be added to the high-level critical assets identified by the assets owner as shown in Equations 4.5 and 4.6.

$$C = \{(r, P_r), (p_1, P_{p_1}), (p_2, P_{p_2}), (f, P_f)\} \tag{4.5}$$

$$P_r = \{integrity, \{p_1\}\}, \quad P_{p_1} = P_{p_2} = P_f = \{integrity, \{\}\} \tag{4.6}$$

4.5.1.3 Scope of Control

The security requirements need to be mapped to system calls. Knowing the guest operating system type, system calls, as well as the security requirement, the mapper will map 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, NtReadFile, NtOpenFile, NtCreateFile, and NtSetInformationFile system calls must be prevented to preserve “confidentiality”.

To catch the indirect relationships to the critical assets, we can consider the data flow direction [18]. For confidentiality, the data flows outwards starting from the critical asset. For the integrity on the other hand, data flows towards the critical asset. Finally, availability data flows in and out of the critical asset. The data flow is going from process to file for a write system call. For example, if the relationship between process p_1 and file f_1 is write, then the corresponding system call will be (NtWriteFile, p_1, f_1) and the data flow will be from p_1 to f_1. But if the relationship between p_1 and f_1 is read, then the corresponding system call will be (NtReadFile, p_1, f_1) and the data flow will be from f_1 to p_1. By tracing the data flow we can identify the indirect critical assets all along the reachability path.

It should be noted that all objects included in the scope of control should have integrity as the security requirement. In addition, all objects in the scope of control should inherit the security requirement of the critical asset identified by the asset’s owner.

Integrity Example

We assume a scope of control is shown in Figure 4.6 and the security requirement for r is integrity. The data flow is indicated by the dashed arrows. Therefore, all objects along these dashed arrows will be included as critical assets. For further illustration and clarification, Table 4.2 outlines some cases and whether these cases should be included in the scope of control. All objects included in the scope of control should have integrity as the security requirement.

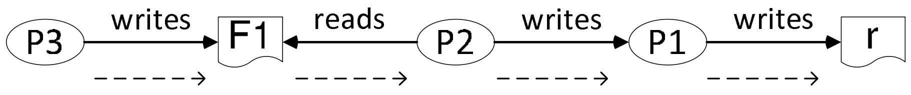


Figure 4.6: Integrity: indirect assets added to the scope of control.

Table 4.2: Integrity: validating the scope of control.

Scenario	Scope of Control	
	Will be included	Will not be included
if p_z is reading from r		✓
if p_z is writing to r	✓	
if p_a is reading from p_1		✓
if p_a is writing to p_1	✓	
if p_b is reading from p_2		✓
if p_b is writing to p_2	✓	
if p_n is reading from p_9		✓
if p_n is writing to p_9		✓
if p_y is reading from f		✓
if p_y is writing to f	✓	

Confidentiality Example

We assume a scope of control is shown in Figure 4.7 and the security requirement for r is confidentiality. The data flow is indicated by the dashed arrows. Therefore, all objects along these dashed arrows will be included as critical assets. For further illustration and clarification, Table 4.3 outlines some cases and whether these cases should be included in the scope of control. All objects included in the scope of control should have integrity as well as confidentiality as the security requirements.

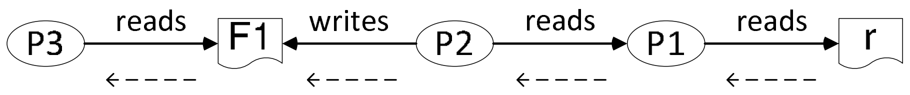


Figure 4.7: Confidentiality: indirect assets added to the scope of control.

Table 4.3: Confidentiality: validating the scope of control.

Scenario	Scope of Control	
	Will be included	Will not be included
if p_z is reading from r	✓	
if p_z is writing to r		✓
if p_a is reading from p_1	✓	
if p_a is writing to p_1		✓
if p_b is reading from p_2	✓	
if p_b is writing to p_2		✓
if p_n is reading from p_9		✓
if p_n is writing to p_9		✓
if p_y is reading from f	✓	
if p_y is writing to f		✓

Availability Example

We assume a scope of control is shown in Figure 4.8 and the security requirement for r is availability. The data flow is indicated by the dashed arrows. Therefore, all objects along these dashed arrows will be included as critical assets. We notice that the direction of data flow is not considered when the security requirement is availability. So regardless of the direction, if there is a data flow between an object and the critical asset, then this object should be added to the scope of control. For further illustration and clarification, Table 4.4 outlines some cases and whether these cases should be included in the scope of control. All objects included in the scope of control should have integrity as well as availability as the security requirements.

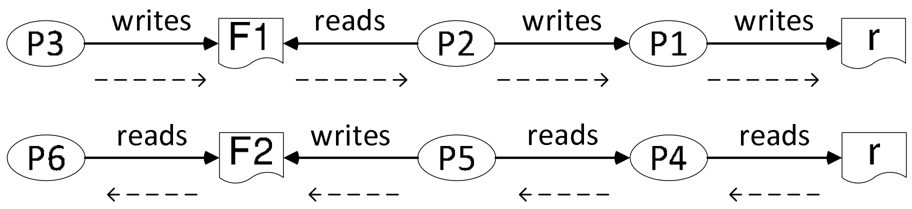


Figure 4.8: Availability: indirect assets added to the scope of control.

Table 4.4: Availability: validating the scope of control.

Scenario	Scope of Control	
	Will be included	Will not be included
if p_z is reading from r	✓	
if p_z is writing to r	✓	
if p_a is reading from p_1	✓	
if p_a is writing to p_1	✓	
if p_b is reading from p_2	✓	
if p_b is writing to p_2	✓	
if p_n is reading from p_9		✓
if p_n is writing to p_9		✓
if p_y is reading from f	✓	
if p_y is writing to f	✓	

4.5.2 Monitoring Phase

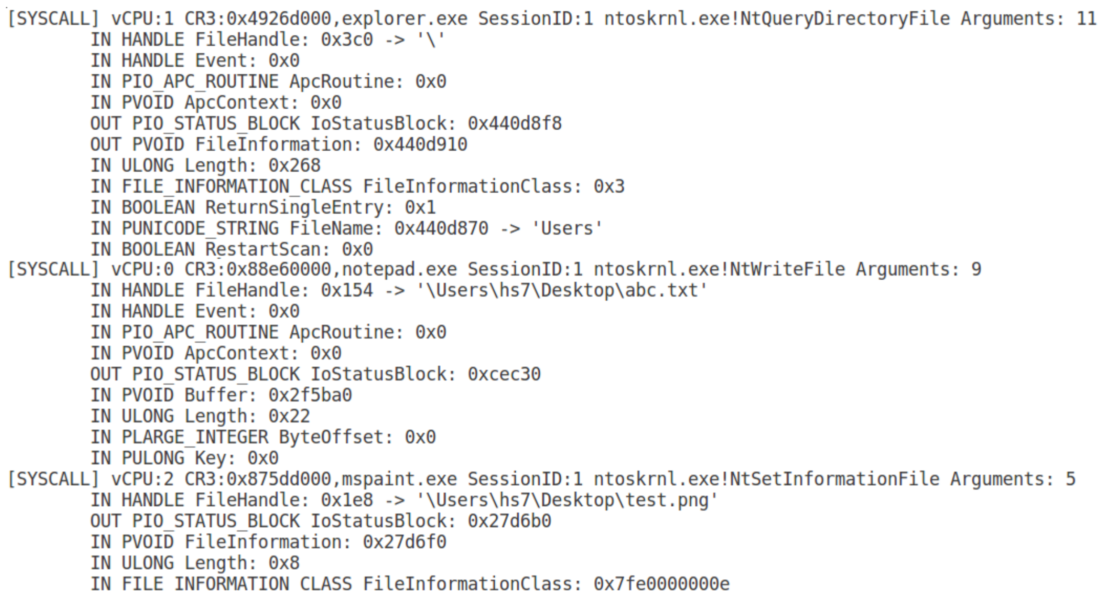
This is the phase responsible for virtual machine introspection by collecting system calls generated by the virtual machine without its knowledge since system calls are intercepted and logged at the hypervisor-level. As shown in Algorithm 1, this phase starts by initializing S, the set of intercepted system calls. This phase can monitor only data flows to the critical assets or monitor all system calls. In the case of “before deployment”, we need to monitor and log all system calls to a file to be processed to generate the reachability graph. While in the case of “after deployment”, we need to know if the captured system call s is trying to access any critical asset $c \in C$. Then, we have to decide if s needs further inspection. If s is trying to access $c \in C$, then add s to S and pass s to the *Decision Phase* for further investigation.

Algorithm 1: Monitoring Phase.

```
S = {}; //initializing the set of intercepted system calls
if Before Deployment then
    intercepted system calls = All;
else
    intercepted system calls = Specified;
end
foreach intercepted system call do
    Parse s //get c and any other relevant information;
    S = S + s //for possible post-mortem analysis;
    if After Deployment then
        if ( c ∈ C ) then
            Decision Phase(C, s) //call Decision Phase;
        end
    end
end
```

Figure 4.9 shows a sample of intercepted system calls generated by processes running in the virtual machine. These processes are being monitored by our security system at the hypervisor-level. 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



```
[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 4.9: Snapshot from raw system calls.

behavior of theses system calls. In other approaches, monitoring is done in order to learn behavior or match signatures.

4.5.3 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. If p is trying to access $c \in C$ and this process is not allowed, then the "decision" is set to "prevent".

Algorithm 2: Decision Phase.

```
Data:
The critical asset set $C$
The intercepted system call $s$
Result: Either prevent or allow
1 decision ← "allow"; //initialize decision to allow
2 if ($s$ contains $c \in C$) then
3     if ($s$ violates $P_c$) then
4         decision ← "prevent";
5         $Feedback\ Phase(C, s)$ //call Feedback Phase;
6     end
7 end
8 return "decision";
```

4.5.4 Feedback Phase

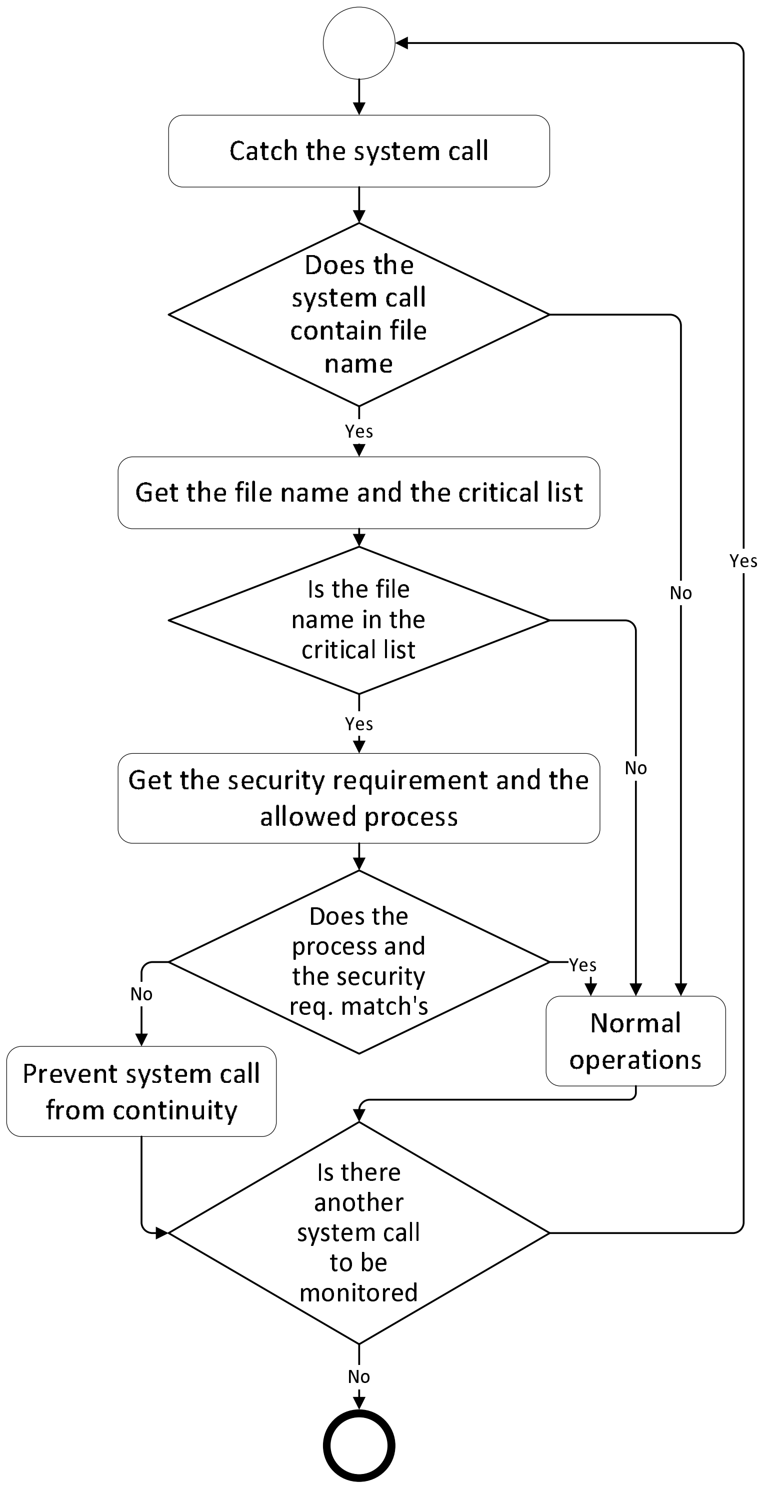
This phase is triggered if certain events are met such as a prevention decision, a controlled modification, or uncontrolled modification. The information taken from the "decision phase" must be fed back to the security system. Prevented decisions coming from the "decision phase" must be inspected. This inspection can be done by the security practitioner. 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. Controlled modification happens when a critical asset, critical process, or security requirement needs to be added or modified. Those changes must be reflected into the security system. Uncontrolled modification happens when an MTD approach is applied to dynamically change some attributes of the critical assets such as a name of a process that is allowed to access one of the critical files.

4.6 Security System Flow Chart

After our security system is deployed, the flow of our security system is shown in Figure 4.10. In our system, the work flow starts with catching every system call. In some operating system such as Windows 7, there is about 700 set of routines. Those 700 routines can be accessed by system call. In user mode applications, system calls must be called to access these set of routines, while in kernel mode they can be called directly. There are two types of these routines [115] namely, NT group and ZW group. NT group can be called from user space when there is no trust while ZW group can be called from the kernel space when there is trust.

Then, a file name will be extracted (if any) from the in the system call parameters. There are more than 4 categories of these system calls. These categories have different number of parameters spanning from one to four parameters. If a file name is found, the file name will be extracted from the system call and stored in a string for later processing.

The stored file name will be matched with the critical file names. If the file name is found in that list, then more processing is needed. Processing done and we fetch information related to that file name such as the security requirement and the process name. Finally, comparison and decision will be taken to either prevent or allow the system calls. Our security system will prevent the system call from continuity by nulling it's parameters if the the decision is to prevent the system call, the decision is to allow the system calls, then our security system will allow the execution to proceed as normal.



Catch the system call
Does the system call contain file name
Yes
No
Get the file name and the critical list
Is the file name in the critical list
Yes
No
Get the security requirement and the allowed process
Does the process and the security req. match's
Yes
No
Normal operations
Prevent system call from continuity
Is there another system call to be monitored
Yes
No

Figure 4.10: Work flow flowchart: after deployment.

CHAPTER 5

ASSET-BASED SECURITY SYSTEM DESIGN

In this chapter, we present the design of our asset-based security system. In our design, we decompose our system into subsystems in order to have a layered architecture. In each of these layers, we state what functions are needed to be performed. As such, we present a systematic approach of defining all system components to satisfy the needs and requirements in order 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 well as information flow between these components. In general, this is a trade off between the comprehensiveness of the monitoring and the performance. However, the system we propose does not require comprehensive monitoring of the guest OS, in fact, it just requires monitoring system calls. Which is available in almost all hypervisor-level monitoring solutions.

5.1 Overall System Architecture

Figure 5.1 shows the overall asset-based security system architecture. Above the hardware subsystem and as shown in Figure 5.1, the hypervisor is composed of two subsystems, namely the Virtual Machine Manager (VMM) and the Virtual Machine Introspection (VMI) subsystems.

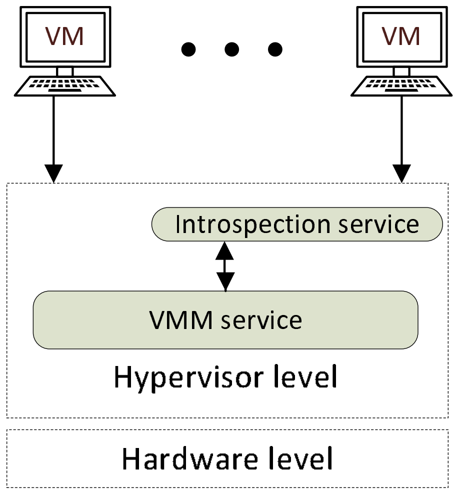


Figure 5.1: Overview of the system architecture.

VMM is a software program that sets the virtualization environment and this basically will enable Virtual Machines (VMs) bootstrapping and governance. VMM manages this operation on top of the hardware layer. VMM provides the virtualization functionality. Once it is installed, VMM facilitates VMs creation with separate operating systems and applications running in each VM. VMM supports the backend operation of allocated VMs by assigning adequate computing power, main memory, secondary storage, as well as other I/O resources. VMM also creates a unified interface for managing the entire virtualization environment.

VMI is basically inspecting the contents of VM in real-time without the agreement nor the knowledge of the guest operating system. In our design, we chose VMI because of this specific property. This contrasts with classic monitoring software on physical systems where the monitoring process runs on the physical system itself. As such, the monitoring system is reachable and indeed other processes running in the guest operating system can know that they are being monitored. What is more, when a virus or malware penetrates a given physical machine, its first task is to deactivate any monitoring process and prevent installation of such processes. This way, the malware can keep on controlling the physical machine.

With our design, it is impossible for processes running on the guest operating system to deactivate or even know of the existence of a monitoring tool. That is why the concept of the VMI is the choice in our design.

5.2 Functional Architecture

As shown in Figure 5.2, 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 at this layer enables an agentless binary analysis system to be 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 6 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 *Identifier* and the *Generator* components, respectively. The *Mapper* translates the security requirements to corresponding system calls needed to be processed. The monitoring functionality is carried out by the *Monitor* while the *Decision Maker* component will inspect the system call after it has been captured by the *Monitor*. 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, through the *Tuner*, of potential violation attempts targeting the critical assets.

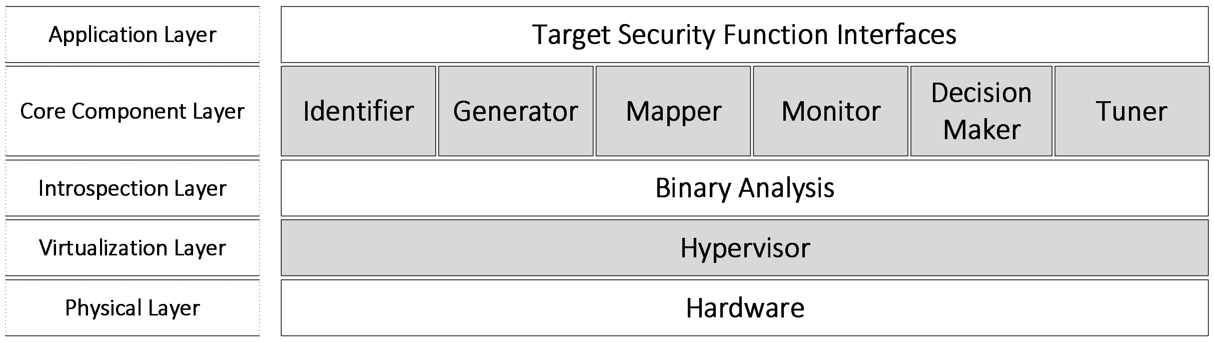


Figure 5.2: Functional architecture design of the proposed security system.

5.3 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. Figure 5.3 shows the operational architecture before deployment where tasks and information are depicted as they flow between the "Information Collection" and the "Monitoring" phases. Assets owners are responsible for providing the name(s) of the asset(s) needed to be protected. Also, the security requirements for these assets need to be provided by the asset owner. The name(s) of the asset(s) as well as the security requirement(s) are given in high level names.

After consulting the security mapping and the security requirements, the mapper translates the security requirements to corresponding system calls. For example NtReadFile, NtOpenFile, NtCreateFile, and NtSetInformationFile system calls must be prevented to preserve "confidentiality" when using windows 7 as the operating system.

After collecting input from the assets owners as well as system call input from the operating system with corresponding system calls coming from mapper, the generator will be ready for processing. Critical assets (high and low) with their security policy, system calls, security requirements translated to system calls will be fed in and given to the generator. In turn, the generator creates the critical asset scope of control. Now, the critical assets scope of control will be used as a reference in the security system after deployment.

If the generator is given only the critical assets and the system calls, then the generator will not be able to generate the scope of control but it will generate the reachability graph.

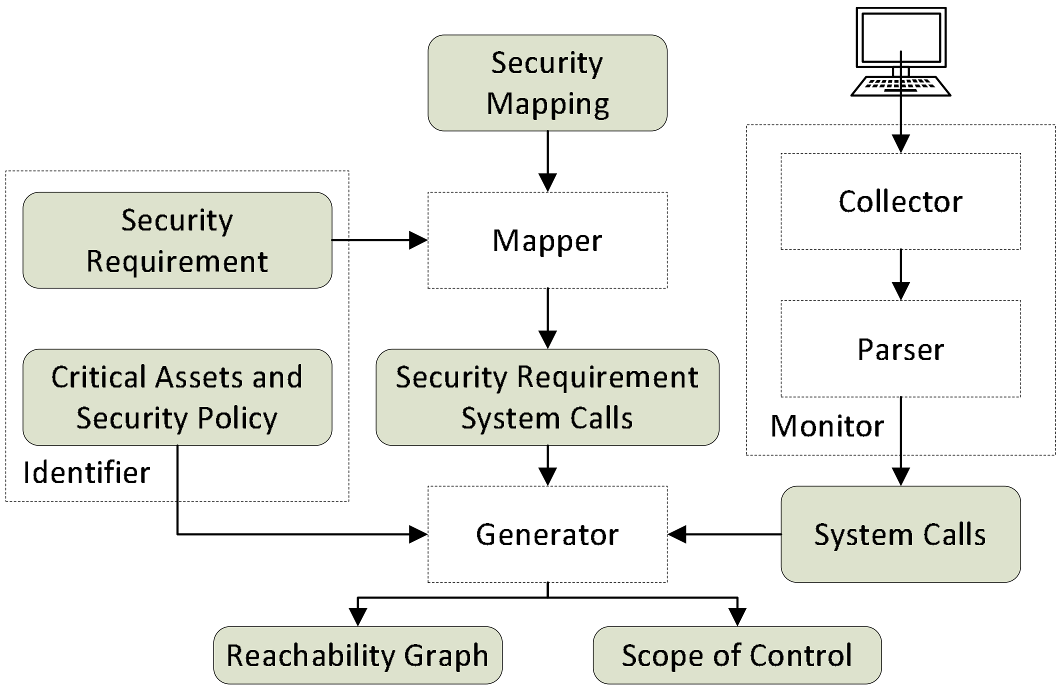


Figure 5.3: Operational architecture before system deployment.

After deploying the security system as in Figure 5.4, the "Monitoring" and "Decision" phases begin. The monitor starts collecting system calls by the "Collector". The collection here will be done to system calls, which will lead to catching every critical operation. The parses then processes collected system calls extracting needed information such as process name, system call name, and file name. The *decision maker* starts checking user processes that initiate system calls and consulting with the reference model (i.e. the critical assets scope of control). The *decision maker* reports any system call in violation of the reference model. Consequently, the system call execution will be interrupted and stopped from execution. If a violation is detected, a warning message is send by the *decision maker* to the *tuner* in our security system. The *tuner* 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.

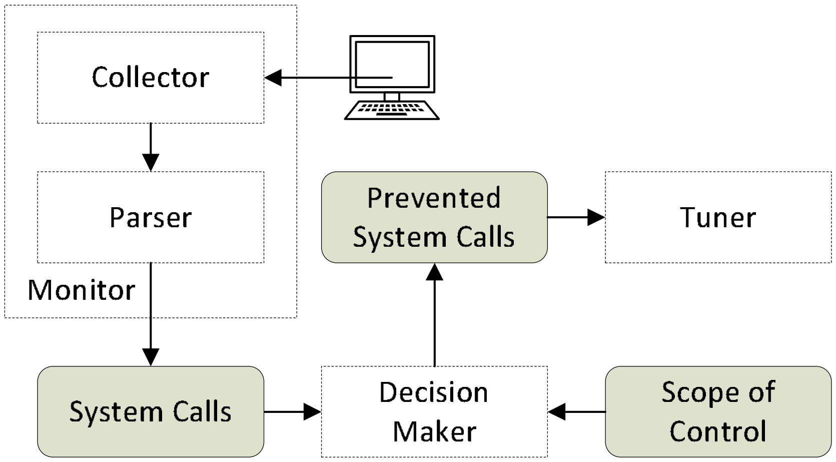


Figure 5.4: Operational architecture after system deployment.

5.4 System Design Validation

In this section, we discuss how the contributions of our security system are achieved by our design choices. One of the main objectives of our security system is to build an asset-based security solution. As such, our design falls under the information security arena not under the physical nor the e-ICT security spaces. Furthermore, our security system defends these assets and hence our next objective is to build a defense-based security solution. This is achieved in our design by having all system components independent of the attack vector. System components start by collecting information regarding the critical assets to be protected and then build other components to defend the identified assets irregardless of attack vectors or surfaces as shown in Figure 5.4. This makes our design proactive since the assets and their security requirements are set before the system goes online and without considering any attack vector. The passive objective means that the attacker will not notice the existence of our system. This is achieved because our system does VMI and operates at the hypervisor level as depicted in Figures 5.1, 6.1, and 6.2. VMI is basically inspecting the contents of VM in real-time without the agreement nor the knowledge of the guest operating system. In our design, we chose VMI because of this specific property. This contrasts with classic monitoring software on physical systems where the monitoring process runs on the physical system itself. As such, the monitoring system is reachable and indeed other processes running in the guest operating system can know that they are being monitored. What is more, when a virus or malware penetrates a given physical machine, its first task is to deactivate any monitoring process and prevent installation of such processes. This way, the malware can keep on controlling the physical machine.

With our design, it is impossible for processes running on the guest operating system to deactivate or even know of the existence of a monitoring tool. That is why the concept of the VMI is the choice in our design.

CHAPTER 6

EVALUATION ENVIRONMENT

The evaluation environment is setup as shown in Figure 5.1. We followed the overall architecture and hence the layered design presented in chapter 5. In this chapter, we first discuss the system specification used in the evaluation environment. Then, we discuss in details our virtualized environment and the tools used to achieve VMI. We finally present the benchmarking tool used for time measurement and how the evaluation environment is setup.

6.1 System Specification

We list here the system specification including hardware, system software, and application software used to perform the evaluation experiments:

- Host machine
  - Type: Alien PC
  - Processor: Intel Core I7 Quad Core 4700MQ @ 2.4GHz

  - RAM: 24 GB
  - HDD: 1TB @ 5400RPM
  - Host OS: Ubuntu 16.10 64Bit
- Guest virtual machine
  - OS: Windows 7 64Bit
  - CPU: 1 Core
  - RAM: 3000MB
  - HDD: 20GB
- Software
  - Hypervisor tool: QEMU, Xen
  - Binary analysis tool: DECAF, DRAKVUF
  - Benchmarking Application Startup Timer: AppTimer

6.2 Virtualization Environment

To set up the virtualization environment, a software layer is needed to virtualize all of the resources of a physical machine (host machine). This software layer is known in the literature as hypervisor or VMM. The hypervisor also defines and supports multiple VMs execution [116]. In our evaluation environment, we used two hypervisors, namely Quick Emulator (QEMU) and Xen.

6.2.1 QEMU

QEMU is a CPU powerful emulator that can emulate a group of processor types. In 2005, QEMU [117] was presented as a fast machine emulator using an original portable dynamic translator. It emulates several CPUs on several hosts like (x86, PowerPC, ARM, SPARC) in addition to Alpha and MIPS. QEMU has the ability to support full system emulation in which a complete and unmodified operating system is run in a virtual machine. QEMU is an open source hosted hypervisor that executes hardware virtualization. As such, QEMU can act as a hypervisor and its strength and popularity come from being an emulator. QEMU is considered Type-II hypervisor that runs as other computer applications do, at the top of an OS.

6.2.2 Xen

Xen was first released in 2003 [118] with the Para Virtualization (PV) approach. The Xen Project 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 physical machine. It is the only available open source as bare-metal hypervisor . It is used as the basis for a number of different commercial and open source applications, such as security applications, Infrastructure as a Service (IaaS), desktop or server virtualization, embedded and hardware appliances [119] [120]. 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 [121].

In Xen PV, hardware virtualization is not needed so guest kernels are modified to avoid binary translation. This way, the guest os can run on Xen hypervisor and detect hypercalls [122]. On the other hand, Xen supports Fully Virtualization (FV) with the Hardware-assisted Virtualization (HVM) option. This option needs CPU with Virtualization technology such as Intel-VT. Therefore, there is no need to modify guest kernels which will not be able to detect virtualization. Due to this, PV would be faster than FV and FV-HVM [123]. In HVM, when critical instructions are caught, traps are put in place so the hypervisor can emulate it in software [122].

Xen comes in different modes or virtualization types. It should be noted that all hypervisors (either Type-II/hosted or Type-I/bare-metal) need an underlying OS. As such, bare metal also has an operating system on top of which the hypervisor runs [123].

6.3 Binary Analysis

After setting the virtualization environment, a binary analysis tool is needed to manipulate the guest OS behavior. Binary analysis can be achieved using various techniques [124] such as the Dynamic Executable Code Analysis Framework (DECAF) and DRAKVUF. These two possible valid binary analysis tools are used in our model. DECAF is built on the top of QEMU with TEMU as a sub-component.

TEMU, Vine, and Rudder are the three main components for BitBlaze [125] [126]. The common technique used among them is capturing persistent changes to system state which done by emulating all code in software.

Injecting breakpoints rather than just logging system calls, is another technique applied to achieve binary analysis. This technique is used by DRAKVUF. In this technique, context switches or system calls are caught and a breakpoint is injected to control the behavior of the execution thread [124].

6.3.1 DECAF

DECAF [127] is a dynamic binary analysis platform based on QEMU [128]. It is virtual machine based, multi-target, whole system dynamic binary analysis framework able to do introspection as Just-In-Time VM. Authors in [128] provided DECAF plugins such as Instruction Tracer, Keylogger Detector, and API Tracer. Those plugins can be modified or updated as needed. DECAF was implemented using C and C++ with approximately 20 thousands lines in code, and evaluated using CPU2006 SPEC benchmarks showing average overhead of 12% for VMI. To show the flexibility and scalability of DECAF, DroidScope, a dynamic Android malware analysis platform, was developed as an extension to DECAF [129], for Android mobile devices.

6.3.2 DRAKVUF

6.4 Selected Environment

6.4.1 QEMU and DECAF

In our environment as shown in Figure 6.1, we assume having DECAF which is integrated with QEMU. The approach works by first loading a plugin to QEMU at runtime. This plugin works by applying a system hook for a specified system call for each newly created process in the system. We will have a callback function that is triggered whenever the system call is fired in the CPU. This is done by first checking the value of the EIP register and comparing it with the target system call address. If it is true we trigger a callback function that retrieves the return address and parameters in a struct. We can use that struct to retrieve useful information about the generated system call.

The approach we used take the advantage of using DECAF which provides a JIT VMI; allowing for run time adjustments for guest operating system. We can load the plugin at any time and get the required results. Also, our approach provides malware analyzers with good information about the specified system call by enquiring the system call parameters for further analysis. Additionally, the technique we used allow for system wide API hooking by tracking all newly created processes. More importantly, it is transparent to the guest operating systems making it difficult for running processes to detect if they are being monitored.

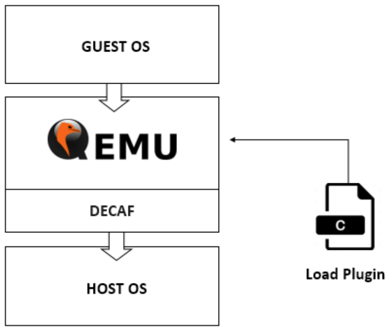


Figure 6.1: QEMU and DECAF evaluation environment.

6.4.2 Xen and DRAKVUF

We used the Xen [120] hypervisor to host the virtual machines and DRAKVUF [130] to provide agentless VMI. With privileges gained from Xen. DRAKVUF can create full VM clones by Copy-on-Write (CoW) memory interface and Copy-on-Write disk capability from Linux LVM. LibVMI library enables DRAKVUF to make use of DMA. LibVMI is "a C library with Python bindings that makes it easy to monitor the low-level details of a running virtual machine by viewing its memory, trapping on hardware events, and accessing the vCPU registers. This is called virtual machine introspection" [131].

At selected code locations, breakpoints are written into the VMs memory. When these breakpoints are reached, DRAKVUF triggers transfer of control to XEN. To achieve stealth, DRAKVUF hijacks an arbitrary process within the VM by using active VMI through breakpoint injection. Rekall is a memory analysis framework [132]. Rekall comes in place to parse the debug data to establish a map of internal kernel functions instead of using the brute force methods (i.e signature based scans) in order for DRAKVUF to automatically locate the kernel in memory. Creating and then reverting analysis container is faster than imaging then reverting physical machines [130]. DRAKVUF is the doing the former while DECAF is doing the latter. That is why DRAKVUF has better performance than DECAF.

Our evaluation environment using Xen and DRAKVUF is shown in Figure 6.2. As shown in the Figure DRAKVUF reside on Domain zero (Dom0). With some privileges DRAKVUF can make clones from the VM to be accessed later, using LibVMI with DMA to monitor context switching and system calls. With Rekall and its predefined kernel profile for specific OS, DRAKVUF can easily trapped specific system calls.

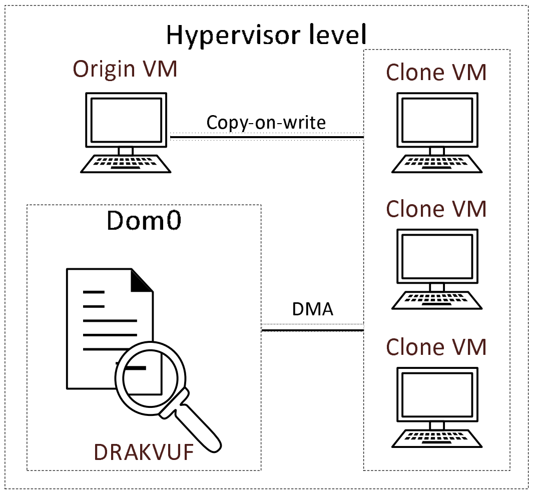


Figure 6.2: Xen and DRAKVUF evaluation environment.

6.5 Evaluation Environment Setup

We setup our evaluation environment using the latest version of Ubuntu 16.04 LTS, Xen 4.9, and DRAKVUF 0.9. We installed Ubuntu 16.04 as the host OS and then installed Xen. Few technical steps are done to merge Xen with Ubuntu before preparing the environment for DRAKVUF installation. We start installing LibVMI and Rekall and then DRAKVUF to complete the VMI process. Now, our virtualization environment is ready and we can finally install a guest OS in a VM. We used Windows 7 as the guest OS and now we can monitor the guest OS from the hypervisor level using DRAKVUF through VNC software.

Full details for the installation guide can be found in Appendix A. For completeness and clarity purposes, these detailed and sequenced steps are summarized as follows:

1. Install the latest version of Ubuntu 16.04 LTS as the host OS.

2. Prepare the environment for virtualization by installing some needed packages such as gcc, python-dev, libc6-dev-i386, libvncserver-dev, and libjson-c-dev.

3. Install a version of Xen that includes a built-in XSM policy required for DRAKVUF.

4. Dedicate some resources specifically for Demo0. In our setup, we dedicated 24 GB of RAM with 4 CPU cores for Demo0.

5. Reboot the and select the following option: "Ubuntu GNU/Linux, with Xen hypervisor". This option guarantees that Demo0 is working with Xen support.

6. Setup LVM Volume Group to hold your VMs disks. Then, create a volume. We created a 20 GB volume for the guest OS.

7. Install Windows 7 from ISO. Enter the LibVMI folder in the DRAKVUF folder and build it.

8. Build and install LibVMI and ReKall [133] [134].

9. Create the Rekall profile for the Windows domain.

10. Test if LibVMI is working by running vmi-process-list.

11. Install DRAKVUF. Trace the execution of the system by picking which DRAKVUF plugins to run. Doing this step will prevent all mentioned plugins from running.

12. DRAKVUF now can run with the selected plugins that point to the virtual machine with the following characteristics: domain name "windows7-sp1", Rekall profile name "windows7-sp1.rekall.json". It should be mentioned that these names contain necessary information about the VM kernel, and therefore we can now all of the guest OS behavior can be monitored by DRAKVUF.

6.6 Application Startup Timer

The Application Startup Timer (AppTimer) is a benchmark utility that will measure how long an application has been running. AppTimer is capable of running an application multiple times and calculating how long it takes for the application to reach a state where user input is being accepted before exiting the application. After each run of the application, AppTimer will attempt to close the application in an automated fashion while logging the startup time measurements to a log file. It's main use is in benchmarking an application's startup time. This can be useful when comparing the performance of different applications on different platforms.

CHAPTER 7

PERFORMANCE EVALUATION

This chapter contains two parts of our system evaluation. The first one considers verification and validation of the security system and the second one considers the performance of the security system.

We conducted several experiments to test the effectiveness, the agility, and the performance of our security model. We started our performance evaluation process in QEMU and DECAF environment and then switched to Xen and DRAKVUF environment. This is explained in sections 7.2 and 7.3, respectively. System call mapping experiments are conducted in section 7.4. We verified and validated our security system in section 7.5. Our security system is evaluated for agility and overhead in sections 7.7 to 7.9.

7.1 Performance Metrics

We conducted a series of evaluation studies to examine the overhead of our security system. The performance measures used in these studies are:

- Response Time. This metric computes the time from when the user submits the request to the time the system completes the response and is calculated as follows:

$$R = T_{res} - T_{req} \tag{7.1}$$

  where T_{req} is the time the user finishes the request and T_{res} is time the system completes the response.

- Generated System Call. This metric measures the number of times a certain user application asks the kernel to execute a privileged I/O instruction.

- Performance Ratio. This metric is calculated as the quotient of the divided R_{on} by R_{off} and is calculated as follows:

$$\text{Performance ratio} = \frac{R_{on}}{R_{off}} \tag{7.2}$$

where R_{on} is the response time when our security system is activated and R_{off} is the response time when our security system is not activated. This performance metric measures our security system overhead or slowdown in terms of response time.

Figure 7.1 shows two possible implication of "Response Time". It is either the time between the user finishing a request and the time when the system starts or completes the response. In our evaluation experiments, we adopted the second definition as outlined above because we want to measure the delay incurred by our security system until the system call is completely executed.

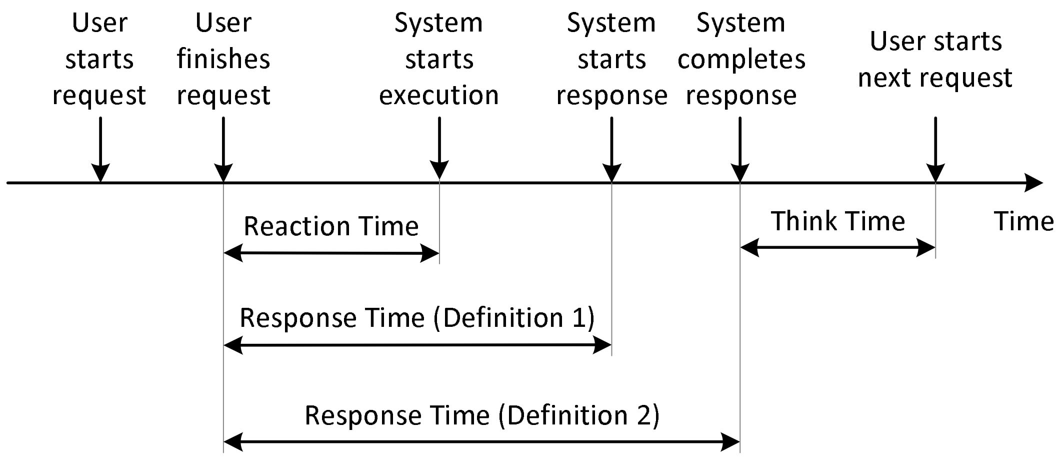


Figure 7.1: Definition of response time. Adapted from [3] .

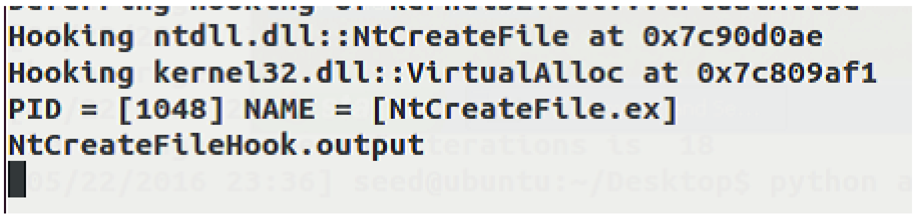
We also examine the number of system calls generated by applications. We do this for two reasons. We want to calculate the number of system calls captured by the monitoring phase and then calculate the number of system calls needed for the analysis phase.

7.2 QEMU and DECAF Experiments

We did the experiments using windows XP as the guest OS, Linux Ubuntu 12.04 as host OS, QEMU version 2.3, and DECAF. We use NtCreateFile system call as an example for the system call. To test our results, we create our own process that only calls the NtCreateFile system call.

We see from Figure 7.2 that we are installing the hook upon knowing the address space of the NtCreateFile system call. This enables us to create a virtual memory to store the hook structs and call stack of the function call.

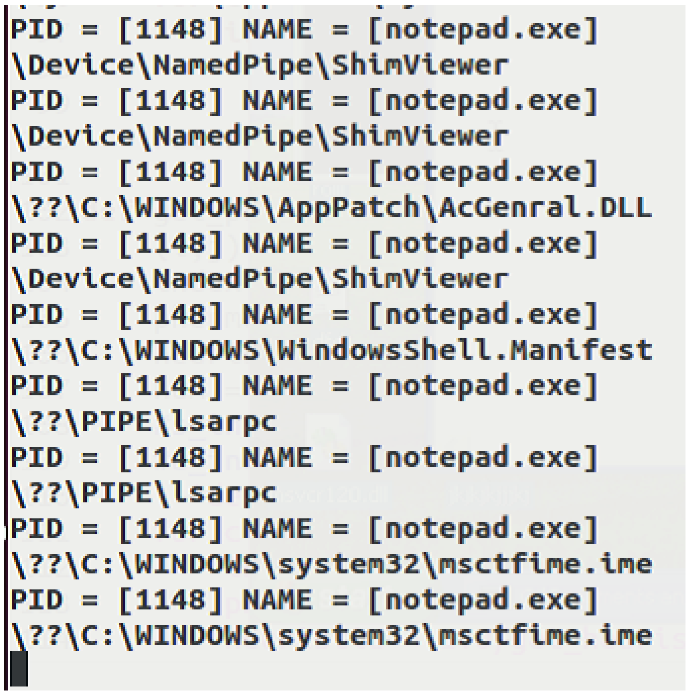
At this stage, we are ready to test the code. We tested our code against notepad.exe and we got the following results. In Figure 7.3, we see that process



```
Hooking ntdll.dll::NtCreateFile at 0x7c90d0ae
Hooking kernel32.dll::VirtualAlloc at 0x7c809af1
PID = [1048] NAME = [NtCreateFile.ex]
NtCreateFileHook.output
```

Figure 7.2: Hooking NtCreateFile.

id, the process name and the filename. The process name can be retrieved by examining the CR3 register to check the page table range then find the corresponding process address space. The filename retrieval is system dependent and highly relies on the file system in the guest operating system. In our example, we have the following signature of the NtCreateFile system call from MSDN.

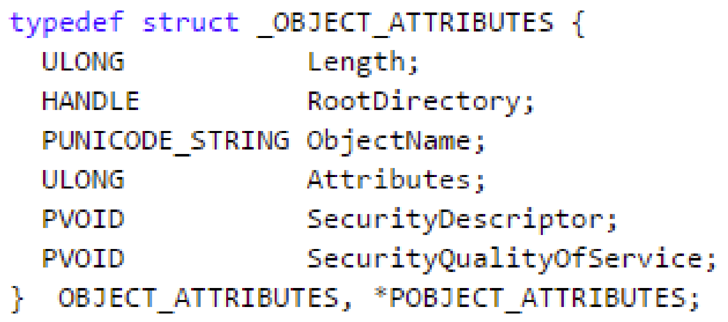


```
PID = [1148] NAME = [notepad.exe]
\Device\NamedPipe\ShimViewer
PID = [1148] NAME = [notepad.exe]
\Device\NamedPipe\ShimViewer
PID = [1148] NAME = [notepad.exe]
\??\C:\WINDOWS\AppPatch\AcGenral.DLL
PID = [1148] NAME = [notepad.exe]
\Device\NamedPipe\ShimViewer
PID = [1148] NAME = [notepad.exe]
\??\C:\WINDOWS\WindowsShell.Manifest
PID = [1148] NAME = [notepad.exe]
\??\PIPE\lsarpc
PID = [1148] NAME = [notepad.exe]
\??\PIPE\lsarpc
PID = [1148] NAME = [notepad.exe]
\??\C:\WINDOWS\system32\msctfime.ime
PID = [1148] NAME = [notepad.exe]
\??\C:\WINDOWS\system32\msctfime.ime
```

Figure 7.3: Monitoring notepad.exe.

We modify some plugins in DECAF, namely we use API_TRACER and HOOKAPITESTS, with some modification through the code we generate a list of all system calls called by certain process using the first plugins API_TRACER, using the second plugins we were able to catch all the process used to call a certain system call, from both plugins we can generate the reachability graph, which could used in later.

The call stack contains the addresses of all of these parameters. The third attribute (OBJECT_ATTRIBUTES) is a structure that contains an ObjectName struct, as shown in Figure 7.4. We can then use the ObjectName struct to retrieve the full path of the file.



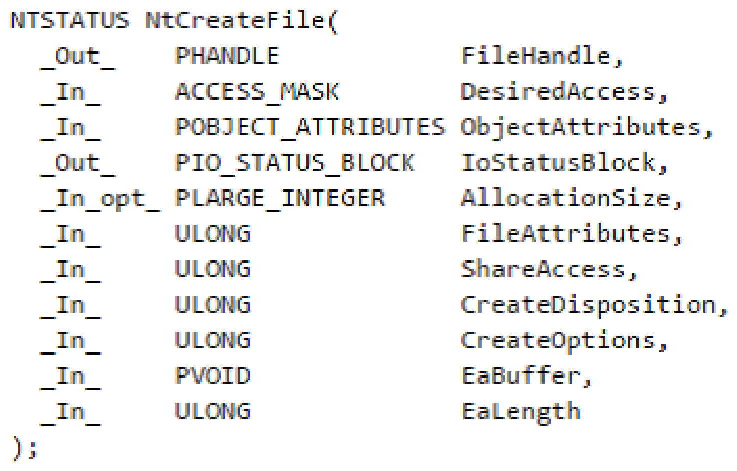
```
typedef struct _OBJECT_ATTRIBUTES {
  ULONG           Length;
  HANDLE          RootDirectory;
  PUNICODE_STRING ObjectName;
  ULONG           Attributes;
  PVOID           SecurityDescriptor;
  PVOID           SecurityQualityOfService;
}  OBJECT_ATTRIBUTES, *POBJECT_ATTRIBUTES;
```

Figure 7.4: OBJECT_ATTRIBUTES class.

We conclude from that we have to do an extra work to retrieve extra information about the hook system call. This will be highly system dependent and relies directly on the signature of the system call and the data structures and data types used to store parameters.

7.3 DRAKVUF Integration

Instead of developing our security system from scratch, we utilized DRAKVUF and developed our security system around it. As explained earlier, DRAKVUF provides a suitable environment for malware analysis. This is established by



```
NTSTATUS NtCreateFile(
  _Out_     PHANDLE            FileHandle,
  _In_      ACCESS_MASK        DesiredAccess,
  _In_      POBJECT_ATTRIBUTES ObjectAttributes,
  _Out_     PIO_STATUS_BLOCK   IoStatusBlock,
  _In_opt_  PLARGE_INTEGER     AllocationSize,
  _In_      ULONG              FileAttributes,
  _In_      ULONG              ShareAccess,
  _In_      ULONG              CreateDisposition,
  _In_      ULONG              CreateOptions,
  _In_      PVOID              EaBuffer,
  _In_      ULONG              EaLength
);
```

Figure 7.5: NtCreateFile parameters.

capturing system calls. Therefore, the idea behind developing DRAKVUF was behavior-analysis and the captured system calls ignored anything to do with assets. In order for us to utilize DRAKVUF in our security system, we need to get asset information when capturing system calls. We started modifying DRAKVUF to include filename as a parameter in the system calls. searching for the file name in the parameter of the system calls requires a lot of sting comparisons as the name of the file stored randomly regarding the system call, so it could be the second parameter or the last one, one the other hand it requires tracing efforts, as it could be a pointer rather than a String.

In the 30^{th} of June 2017, a new version of DRAKVUF was released (DRAKVUF 0.5), that includes the filename as a parameter in the system call. Furthermore, the new release of AVG Internet Security - Unlimited [135] and Bitdefender 2017 [136], contains an option to protect some folders from ransom attacks, by preventing the untrusted application to access these folders, and for sure the user himself can customize the trusted application list and the protected folders. This was a motivation for us as the assets importance is gain attention from the research community.

Modifying DRAKVUF

To develop our security system, we have to do the following changes to DRAKVUF: It should be noted that all these changes are included in Appendix A.

1. **Asset Identification Plugin:** We added this plugin to have the Critical Assets Identification capability.
2. **Reachability Graph Plugin:** We did this by utilizing some existing DRAKVUF Python plugins to generate the reachability graph.
3. **Monitoring Plugin:** We took advantage of DRAKVUF breakpoints to selectively modify call functions in the SYSCALLS plugin.
4. **Decision Plugin:** We added this plugin to DRAKVUF to enable the prevention as well as the feedback capabilities of our security system.

In a nutshell, we modified the SYSCALLS plugin within the DRAKVUF system and injected breakpoints only to selected system calls that could breach files security requirements instead of injecting breakpoints to all NT system calls. The reason of this modification is enhancing DRAKVUF performance. Furthermore, we modified the callback functions within the SYSCALLS plugin and corrupted the system call arguments. This measure was taken if the system call is in violation of as asset's security requirement. We modified arguments in four registers, namely RCX, RDX, R8, and R9. This ensures that the system call will never access the asset.

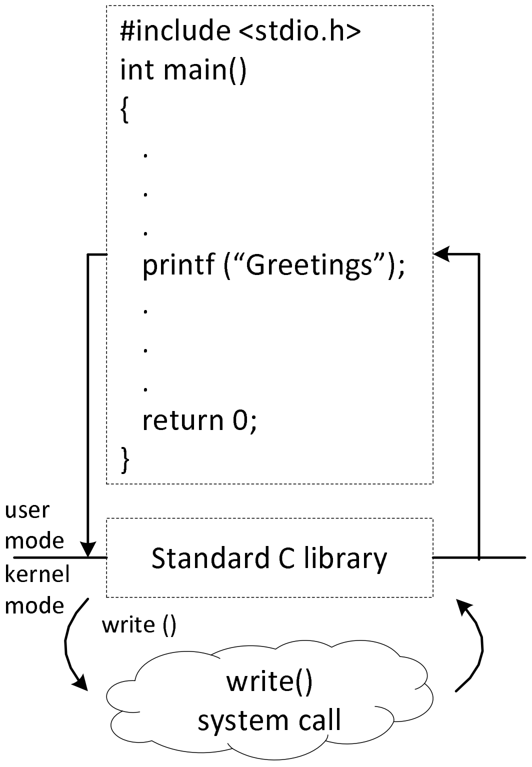
7.4 System Calls Mapping

Our security system can analyze all or a subset of the captured system calls. To improve performance in [6], monitored system calls were minimized to 29. Authors in [6] monitored system calls related to malware behavior . They started monitoring NtOpenFile and NtCreateFile. These two system calls affect file renaming and copying. Later, they added other network related system calls for a total of 29 system calls.

In our experiments, we want to determine the system calls associated with "integrity", "availability", and "confidentiality". As such, "writing, appending" statements are associated with "integrity". Similarly, "reading, opening" and "deleting, renaming" are associated with "confidentiality" and availability, respectively.

In Figure 7.6 adopted from [2], the C user code invokes printf() statement. This statement is intercepted by the C library, which interacts with the kernel on behalf of the user program. Eventually, the printf() statement is mapped as write() system call in kernel mode. Once the kernel executes the write() system call, the returned value is passed to the user program.

In our evaluation, the user program invokes a statement that needs to executed



```
#include <stdio.h>
int main()
{
  .
  .
  .
  printf ("Greetings");
  .
  .
  .
  return 0;
}
```

Figure 7.6: Example of standard library. Adapted from [2].

by the kernel on behalf of the user program. Such statements are referred to as privileged statements [2]. At user space these statements call an interface library. The interface library does mode switching and give the command to the kernel which executes the system call. In order for our security system to capture the system calls, we need first to know which system calls correspond to the privileged statements (i.e. I/O privileged statements) invoked at user space.

As such, we need to map I/O privileged statements at the user space to the system call at the kernel space. We conducted several experiments and examined I/O privileged statements, namely open, view, delete, rename, write, read, and append. It should be noted that in all of these experiments, our security system is running. The following subsections outline our findings.

7.4.1 Open

To get the system calls invoked when we open a file, we conducted the following experiment: While hovering over a file, right-click and open the file with "Notepad". Examining the system call log file, we notice the following: (1) The name of the file appeared in the log 11 times, (2) The system calls are invoked by two processes namely, "explorer.exe" and "notepad.exe". (3) The system calls are NtQueryAttributesFile, NtQueryDirectoryFile, NtCreateFile, NtQueryVolumeInformationFile, NtQueryInformationFile, NtCreateSection. The results of this experiment are summarized in Table 7.1.

Table 7.1: Mapping open statement to system calls: without double click.

System Call	Repetition	Process Name
NtQueryAttributesFile	2	explorer.exe
NtQueryDirectoryFile	3	
NtQueryDirectoryFile	2	notepad.exe
NtCreateFile	1	
NtQueryVolumeInformationFile	1	
NtQueryInformationFile	1	
NtCreateSection	1	

Table 7.2: Mapping open statement to system calls: with double click.

System Call	Repetition	Process Name
NtQueryAttributesFile	2	explorer.exe
NtQueryDirectoryFile	3	
NtCreateFile	1	
NtQueryVolumeInformationFile	2	
NtQueryInformationFile	1	
NtFsControlFile	1	
NtQueryDirectoryFile	2	notepad.exe
NtCreateFile	1	
NtQueryVolumeInformationFile	1	
NtQueryInformationFile	1	
NtCreateSection	1	

7.4.2 View

In MS Windows, we can view the content of a file without opening it. This is doable by opening the folder containing the file in Windows Explorer then selecting the file. The file content will appear in the preview pane. Doing this, we notice the following: (1) The name of the file appeared in the log 34 times. (2) The system calls invoked are NtOpenFile, NtQueryAttributesFile, NtCreateFile, NtFsControlFile, NtReadFile, and NtSetInformationFile. (3) All of them are generated by "explorer.exe". The results of this experiment are summarized in Table 7.3.

Table 7.3: Mapping view statement to system calls.

System Call	Repetition	Process Name
NtOpenFile	12	explorer.exe
NtQueryAttributesFile	14	
NtCreateFile	4	
NtFsControlFile	1	
NtReadFile	2	
NtSetInformationFile	1	

7.4.3 Delete

Here, we want to capture the system calls generated when file is indirectly deleted (i.e., sent to the recycle bin) or directly deleted (i.e., press the shift key with the delete key). Pressing the delete key on the keyboard will send the file to the recycle bin. On the other hand, pressing the shift key with the delete key will delete the file immediately without sending it to the recycle bin. Doing this experiment, we notice the following: (1) The name of the file appeared in the log

11 times, (2) There are 5 system calls appeared to interact with the file. These system calls are NtCreateFile, NtQueryDirectoryFile, NtQueryInformationFile, NtSetInformationFile, and NtOpenFile. (3) All of the system calls are generated by “explorer.exe”. The results of this experiment are summarized in Table 7.4.

Table 7.4: Mapping delete statement to system calls.

System Call	Repetition	Process Name
NtCreateFile	2	explorer.exe
NtQueryDirectoryFile	2	
NtQueryInformationFile	2	
NtSetInformationFile	3	
NtOpenFile	2	

7.4.4 Rename

Renaming a file is done by hovering over the file, right-click, and then choosing rename from the pop-up menu. Doing this experiment, we notice the following for the original file: (1) The name of the original file appeared in the log 8 times. (2) These system calls are NtQueryDirectoryFile, NtOpenFile, NtQueryInformationFile, NtSetInformationFile, and NtCreateFile. (3) All of the system calls generated by “explorer.exe”.

For the new file, we notice the following: (1) The name of the file appeared in the log 28 times. (2) The name of the file repeated 8 times were called by “exploror.exe” and 20 times by “SearchProtocol”. (3) The system calls generated generated by “explorer.exe” were NtQueryDirectoryFile, NtOpenFile, and NtQueryAttributesFile. (4) The system calls generated by “SearchProtocol” are NtCreateFile, NtFsControlFile, NtQueryInformationFile, NtOpenFile, NtSetInformation-

File, and NtReadFile.

The results of this experiment are summarized in Table 7.5. The Table shows both of the results, namely results concerning the original file and results concerning the new file.

Table 7.5: Mapping rename statement to system calls.

File Type	System Call	Repetition	Process Name
Original	NtQueryDirectoryFile	3	explorer.exe
	NtOpenFile	1	
	NtQueryInformationFile	1	
	NtSetInformationFile	1	
	NtCreateFile	2	
New	NtQueryDirectoryFile	3	explorer.exe
	NtOpenFile	3	
	NtQueryAttributesFile	2	
	NtCreateFile	4	SearchProtocol
	NtFsControlFile	4	
	NtQueryInformationFile	6	
	NtOpenFile	2	
	NtSetInformationFile	2	
	NtReadFile	2	

7.4.5 Write and Save

This experiment is done in three steps:

- Step #1: Opening a file and modifying its content then check the log.
- Step #2: Opening a file, modifying its content, and then clicking the save button. Then, check the log.
- Step #3: Opening a file, modifying its content, the click the exit button. A dialogue box pop will ask the user to save the file. We press the save option and check the log.

For step #1, we notice the following: (1) The name of the file appeared in the log 26 times. (2) The system calls are generated by "explorer.exe", "notepad.exe", and "SearchProtocol".

For step #2: we notice the same behavior as in step #1. In addition, we notice the following: (1) The name of the file appeared 38 times in the log in total and 12 of them as new entry. (2) The system calls are generated by "explorer.exe", "notepad.exe", and "SearchProtocol".

As of step #3, the behavior was exactly as the one discussed in step #2. This can be explained by the fact that the actions taken by the user process towards the asset (i.e., the file) are the same. In step #3, we only delay the saving of the file by closing it abnormally, which affect the sequence of the system calls.

The results of this experiment are summarized in Table 7.6. The Table shows the three results, namely results concerning step #1, step #2, and step #3. Since step #2 and step #3 have the same results, they are combined in the same row of Table 7.6.

7.4.6 Append

The objective of this experiment is to explore the different between appending rather than writing to a file. In this experiment, we develop a C program "appendtotext.c" to append to an existing file and to a non-existing file. When the file was not there, the system creates the file then append to it. For the case where the file exits, the system directly append to the file.

Table 7.6: Mapping write and save statements to system calls.

File Type	System Call	Repetition	Process Name
Step #1	NtQueryAttributesFile	2	explorer.exe
	NtQueryDirectoryFile	3	
	NtCreateFile	1	
	NtQueryVolumeInformationFile	2	
	NtQueryInformationFile	1	
	NtFsControlFile	1	
	NtQueryDirectoryFile	2	notepad.exe
	NtCreateFile	1	
	NtQueryVolumeInformationFile	1	
	NtQueryInformationFile	1	
	NtCreateSection	1	
	NtCreateFile	2	SearchProtocol
	NtFsControlFile	2	
	NtQueryInformationFile	3	
	NtOpenFile	1	
	NtSetInformationFile	1	
	NtReadFile	1	
Step #2 and Step #3	NtQueryAttributesFile	4	explorer.exe
	NtQueryDirectoryFile	4	
	NtCreateFile	1	
	NtQueryVolumeInformationFile	2	
	NtQueryInformationFile	1	
	NtFsControlFile	1	
	NtOpenFile	3	
	NtQueryDirectoryFile	3	notepad.exe
	NtCreateFile	2	
	NtQueryVolumeInformationFile	1	
	NtQueryInformationFile	2	
	NtCreateSection	1	
	NtWriteFile	1	
	NtSetInformationFile	2	
	NtCreateFile	2	SearchProtocol
	NtFsControlFile	2	
	NtQueryInformationFile	3	
	NtOpenFile	1	
	NtSetInformationFile	1	
	NtReadFile	1	

In the case of the existing file, we notice the following: (1) The name of the file appeared in the log 21 times. (2) These system calls are NtCreateFile, NtQueryVolumeInformationFile, NtQueryInformationFile,NtSetInformationFile, NtWriteFile, NtFsControlFile, NtOpenFile, NtReadFile, NtQueryDirectoryFile, and NtQueryAttributesFile. (3) The processes “appendtotxt.exe”, “SearchProtocol”, and “explorer.exe” are responsible for generating the system calls. The results of this experiment are summarized in Table 7.7. The Table shows the two results, namely results concerning existing files and results concerning non-existing files.

Table 7.7: Mapping append statement to system calls.

File Status	System Call	Repetition	Process Name
Existing	NtCreateFile	1	appendtotext.exe
	NtQueryVolumeInformationFile	1	
	NtQueryInformationFile	1	
	NtSetInformationFile	1	
	NtWriteFile	1	
	NtCreateFile	2	SearchProtocol
	NtFsControlFile	2	
	NtQueryInformationFile	3	
	NtOpenFile	1	
	NtSetInformationFile	1	
	NtReadFile	1	
	NtQueryDirectoryFile	1	explorer.exe
	NtOpenFile	3	
	NtQueryAttributesFile	2	
Non-Existing	NtCreateFile	1	appendtotext.exe
	NtQueryDirectoryFile	1	
	NtQueryVolumeInformationFile	1	
	NtQueryInformationFile	1	
	NtSetInformationFile	1	
	NtWriteFile	1	
	NtCreateFile	2	SearchProtocol
	NtFsControlFile	2	
	NtQueryInformationFile	3	
	NtOpenFile	1	
	NtSetInformationFile	1	
	NtReadFile	1	
	NtQueryDirectoryFile	2	explorer.exe
	NtOpenFile	3	
	NtQueryAttributesFile	2	

7.4.7 Selected System Calls

The security requirements need to be mapped to system calls. This mapping is done 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, NtReadFile, NtOpenFile, NtCreateFile, and NtSetInformationFile system calls must be prevented to preserve "confidentiality". The meaning of these system calls are shown in Table 7.8.

7.5 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. 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 Figure 7.7, the running processes captured by the task manager within the virtual machine are also captured by our security system.

In our next experiment, we exposed our prototype to an academic crypto-ransomware identical to the famous jigsaw crypto-ransomware of which new variants just appeared in January 2018 [137]. A crypto-ransomware traverses interest-

Table 7.8: Meaning of system calls.

System Call	Meaning
NtCreateFile	Creates a new file or directory, or opens an existing file, device, directory, or volume.
NtCreateSection	The ZwCreateSection routine creates a section object.
NtDeleteFile	The ZwDeleteFile routine deletes the specified file.
NtFsControlFile	The ZwFsControlFile routine sends a control code directly to a specified file system or file system filter driver, causing the corresponding driver to perform the specified action.
NtOpenFile	Opens an existing file, device, directory, or volume, and returns a handle for the file object.
NtQueryAttributesFile	Retrieves basic attributes for the specified file object.
NtQueryDirectoryFile	The ZwQueryDirectoryFile routine returns various kinds of information about files in the directory specified by a given file handle.
NtQueryFullAttributesFile	The ZwQueryFullAttributesFile routine supplies network open information for the specified file.
NtQueryInformationFile	The ZwQueryInformationFile routine returns various kinds of information about a file object.
NtQueryVolumeInformationFile	The ZwQueryVolumeInformationFile routine retrieves information about the volume associated with a given file, directory, storage device, or volume.
NtReadFile	The ZwReadFile routine reads data from an open file.
NtSetInformationFile	The ZwSetInformationFile routine changes various kinds of information about a file object.
NtWriteFile	The ZwWriteFile routine writes data to an open file.

ing 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

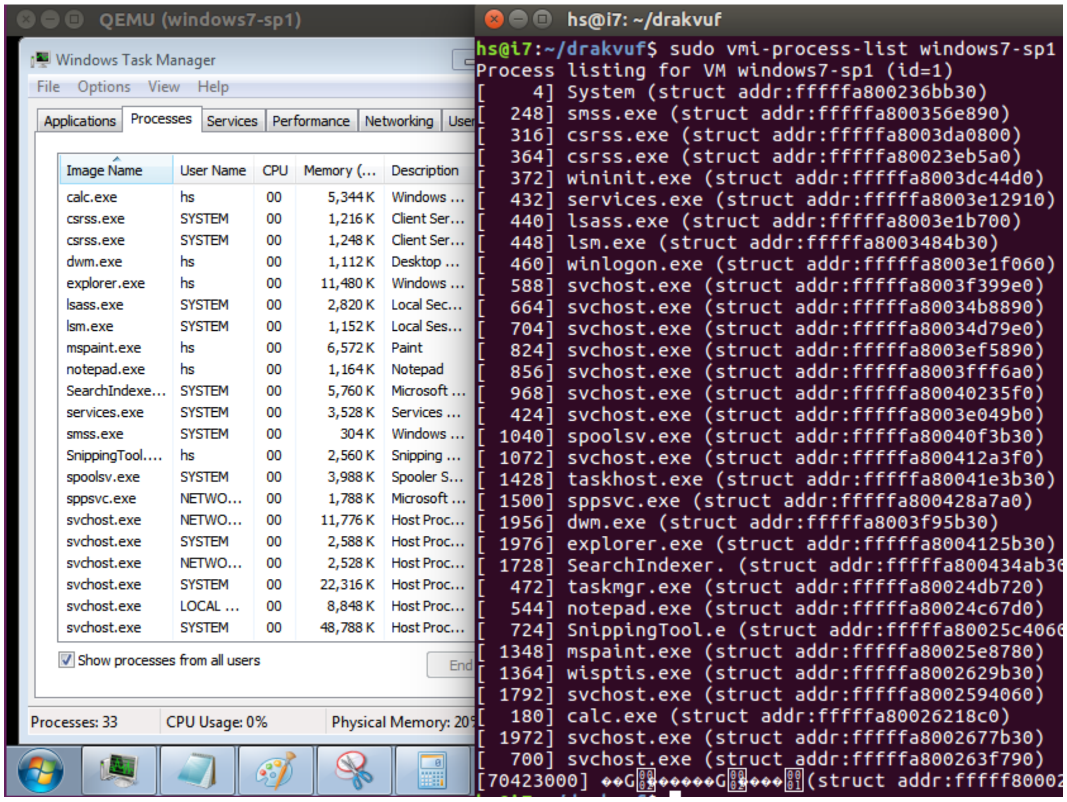


Image Name	User Name	CPU	Memory (...	Description
calc.exe	hs	00	5,344 K	Windows ...
csrss.exe	SYSTEM	00	1,216 K	Client Ser...
csrss.exe	SYSTEM	00	1,248 K	Client Ser...
dwm.exe	hs	00	1,112 K	Desktop ...
explorer.exe	hs	00	11,480 K	Windows ...
lsass.exe	SYSTEM	00	2,820 K	Local Sec...
lsm.exe	SYSTEM	00	1,152 K	Local Ses...
mspaint.exe	hs	00	6,572 K	Paint
notepad.exe	hs	00	1,164 K	Notepad
SearchIndexe...	SYSTEM	00	5,760 K	Microsoft ...
services.exe	SYSTEM	00	3,528 K	Services ...
smss.exe	SYSTEM	00	304 K	Windows ...
SnippingTool....	hs	00	2,560 K	Snipping ...
spoolsv.exe	SYSTEM	00	3,988 K	Spooler S...
sppsvc.exe	NETWO...	00	1,788 K	Microsoft ...
svchost.exe	NETWO...	00	11,776 K	Host Proc...
svchost.exe	SYSTEM	00	2,588 K	Host Proc...
svchost.exe	NETWO...	00	2,528 K	Host Proc...
svchost.exe	SYSTEM	00	22,316 K	Host Proc...
svchost.exe	LOCAL ...	00	8,848 K	Host Proc...
svchost.exe	SYSTEM	00	48,788 K	Host Proc...

☑ Show processes from all users

Processes: 33 CPU Usage: 0% Physical Memory: 20

```
hs@i7: ~/drakvuf
hs@i7:~/drakvuf$ sudo vmi-process-list windows7-sp1
Process listing for VM windows7-sp1 (id=1)
[    4] System (struct addr:fffffa800236bb30)
[  248] smss.exe (struct addr:fffffa800356e890)
[  316] csrss.exe (struct addr:fffffa8003da0800)
[  364] csrss.exe (struct addr:fffffa80023eb5a0)
[  372] wininit.exe (struct addr:fffffa8003dc44d0)
[  432] services.exe (struct addr:fffffa8003e12910)
[  440] lsass.exe (struct addr:fffffa8003e1b700)
[  448] lsm.exe (struct addr:fffffa8003484b30)
[  460] winlogon.exe (struct addr:fffffa8003e1f060)
[  588] svchost.exe (struct addr:fffffa8003f399e0)
[  664] svchost.exe (struct addr:fffffa80034b8890)
[  704] svchost.exe (struct addr:fffffa80034d79e0)
[  824] svchost.exe (struct addr:fffffa8003ef5890)
[  856] svchost.exe (struct addr:fffffa8003fff6a0)
[  968] svchost.exe (struct addr:fffffa80040235f0)
[  424] svchost.exe (struct addr:fffffa8003e049b0)
[ 1040] spoolsv.exe (struct addr:fffffa80040f3b30)
[ 1072] svchost.exe (struct addr:fffffa800412a3f0)
[ 1428] taskhost.exe (struct addr:fffffa80041e3b30)
[ 1500] sppsvc.exe (struct addr:fffffa800428a7a0)
[ 1956] dwm.exe (struct addr:fffffa8003f95b30)
[ 1976] explorer.exe (struct addr:fffffa8004125b30)
[ 1728] SearchIndexer. (struct addr:fffffa800434ab30
[  472] taskmgr.exe (struct addr:fffffa80024db720)
[  544] notepad.exe (struct addr:fffffa80024c67d0)
[  724] SnippingTool.e (struct addr:fffffa80025c4060
[ 1348] mspaint.exe (struct addr:fffffa80025e8780)
[ 1364] wisptis.exe (struct addr:fffffa8002629b30)
[ 1792] svchost.exe (struct addr:fffffa8002594060)
[  180] calc.exe (struct addr:fffffa80026218c0)
[ 1972] svchost.exe (struct addr:fffffa8002677b30)
[  700] svchost.exe (struct addr:fffffa800263f790)
[70423000] ��G��������G����(struct addr:fffff80002
```

Figure 7.7: Process list generated within and outside the virtual machine.

files. According to [137] and the analysis done by [4] [138] [139], 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 [140] for the meaning of the system calls and we show our findings in Table 7.8.

Table 7.9 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 Table 7.9, 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. 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 7.9: System calls initiated by ransomware.exe for issa.txt.

Captured System Call	File Path	Stage #
NtQueryFullAttributesFile	`\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
NtSetInformationFile	`\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
NtCreateFile	`\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
NtQueryInformationFile	`\Users\hs\Desktop\issa.txt`	7
NtSetInformationFile	`\Users\hs\Desktop\issa.txt`	7

Finally, we conducted an experiment to verify that "ransomware.exe" is working properly in our environment. Therefore, 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.

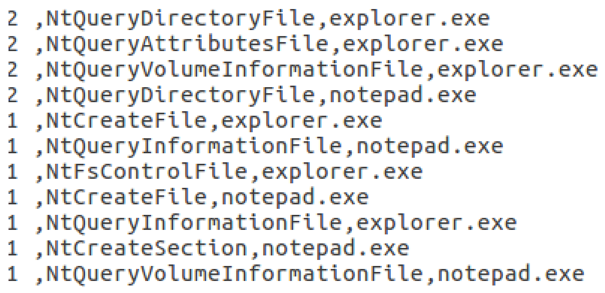
7.6 Generating Reachability Graph

7.6.1 Direct reachability graph

We run the system in a secure environment. This environment is considered to be safe and with no ongoing attacks. The reachability graph will capture low-level critical assets and does not require deep-knowledge expertise about the IT infrastructure. The goal here is to free the system administrator from providing low-level details about the organization because this is done automatically by the reachability graph. The generating of the reachability graph is as follows:

- We start the system that contains the critical assets.
- We start our security system.
- The monitored system calls are written to a log file.
- Our analysis phase parses the log file and organizes the monitored system calls as shown in Figure 7.8.
- The reachability graph will be generated as shown in Figure 7.9.

As shown in 7.9, the critical file is accessed by two processes "notepad.exe" and "explorer.exe". If one of these processes is modified by an unauthorized user, the security requirements of our critical file can be violated. Therefore, there is a need to include the indirect reachability to the critical assets.



```
2 ,NtQueryDirectoryFile,explorer.exe
2 ,NtQueryAttributesFile,explorer.exe
2 ,NtQueryVolumeInformationFile,explorer.exe
2 ,NtQueryDirectoryFile,notepad.exe
1 ,NtCreateFile,explorer.exe
1 ,NtQueryInformationFile,notepad.exe
1 ,NtFsControlFile,explorer.exe
1 ,NtCreateFile,notepad.exe
1 ,NtQueryInformationFile,explorer.exe
1 ,NtCreateSection,notepad.exe
1 ,NtQueryVolumeInformationFile,notepad.exe
```

Figure 7.8: Reading a text file by Notepad.

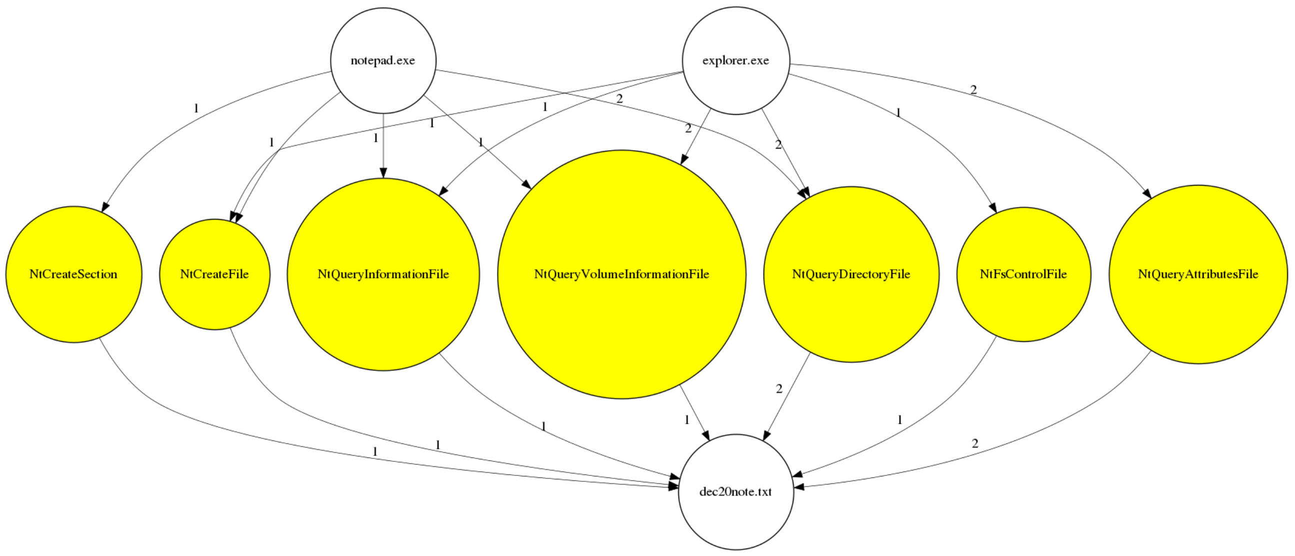
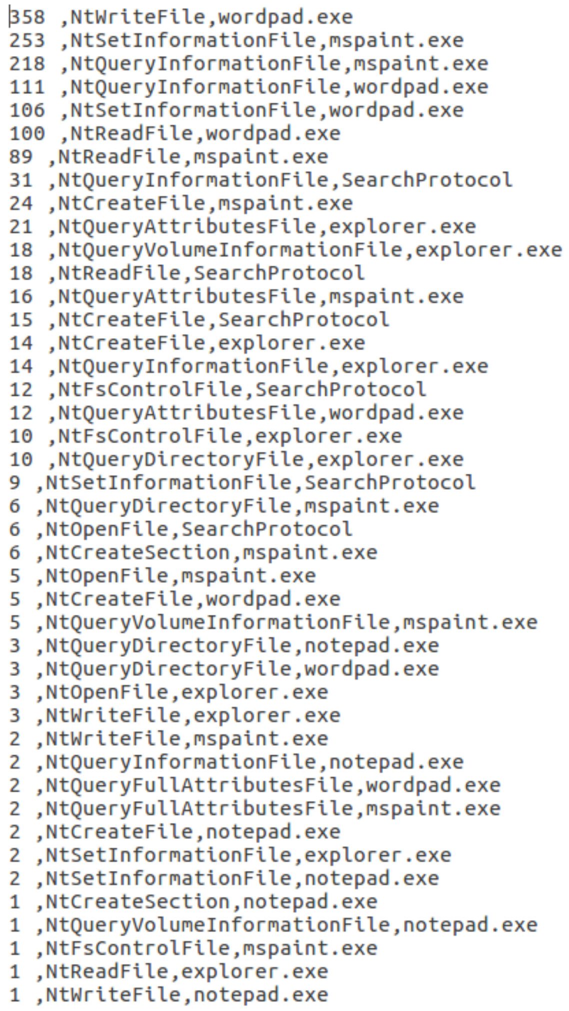


Figure 7.9: Direct reachability graph for reading a text file by Notepad.

7.6.2 Indirect reachability graph

Authors in [18] constructed what is known as "dependency graph" to simulate the interactions between system objects to estimate the probability that a critical assets is compromised if an attacked penetrated through some safe data paths. In our approach, we need to catch the indirect reachability to our critical assets.

The analysis phase get more complex in order to include the indirect low-level critical assets to the reachability graph. The log file is parsed starting the critical asset identified by the asset owner. The direct reachability graph already captured all the processes having direct access to the critical file. We explore further and find out the dependency of other system objects (processes or files) that have interactions with these processes. Any found object will be added as a critical asset to the reachability graph.



```
358 ,NtWriteFile,wordpad.exe
253 ,NtSetInformationFile,mspaint.exe
218 ,NtQueryInformationFile,mspaint.exe
111 ,NtQueryInformationFile,wordpad.exe
106 ,NtSetInformationFile,wordpad.exe
100 ,NtReadFile,wordpad.exe
89 ,NtReadFile,mspaint.exe
31 ,NtQueryInformationFile,SearchProtocol
24 ,NtCreateFile,mspaint.exe
21 ,NtQueryAttributesFile,explorer.exe
18 ,NtQueryVolumeInformationFile,explorer.exe
18 ,NtReadFile,SearchProtocol
16 ,NtQueryAttributesFile,mspaint.exe
15 ,NtCreateFile,SearchProtocol
14 ,NtCreateFile,explorer.exe
14 ,NtQueryInformationFile,explorer.exe
12 ,NtFsControlFile,SearchProtocol
12 ,NtQueryAttributesFile,wordpad.exe
10 ,NtFsControlFile,explorer.exe
10 ,NtQueryDirectoryFile,explorer.exe
9 ,NtSetInformationFile,SearchProtocol
6 ,NtQueryDirectoryFile,mspaint.exe
6 ,NtOpenFile,SearchProtocol
6 ,NtCreateSection,mspaint.exe
5 ,NtOpenFile,mspaint.exe
5 ,NtCreateFile,wordpad.exe
5 ,NtQueryVolumeInformationFile,mspaint.exe
3 ,NtQueryDirectoryFile,notepad.exe
3 ,NtQueryDirectoryFile,wordpad.exe
3 ,NtOpenFile,explorer.exe
3 ,NtWriteFile,explorer.exe
2 ,NtWriteFile,mspaint.exe
2 ,NtQueryInformationFile,notepad.exe
2 ,NtQueryFullAttributesFile,wordpad.exe
2 ,NtQueryFullAttributesFile,mspaint.exe
2 ,NtCreateFile,notepad.exe
2 ,NtSetInformationFile,explorer.exe
2 ,NtSetInformationFile,notepad.exe
1 ,NtCreateSection,notepad.exe
1 ,NtQueryVolumeInformationFile,notepad.exe
1 ,NtFsControlFile,mspaint.exe
1 ,NtReadFile,explorer.exe
1 ,NtWriteFile,notepad.exe
```

Figure 7.10: Accessing a file by Notepad, WordPad, and MS Paint.

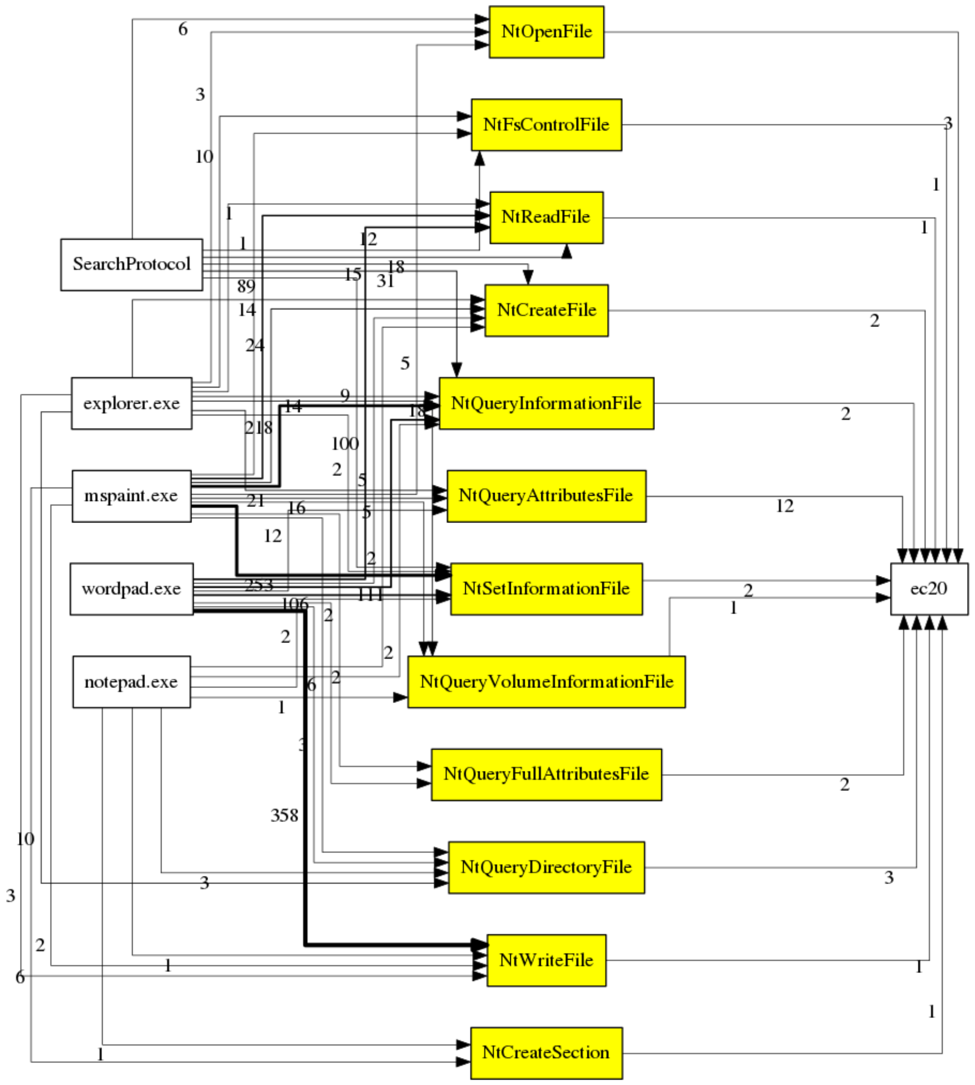


Figure 7.11: Direct reachability graph for accessing a file.

7.7 Agility of the Security System

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 Equations 1 to 3 explained in Section 4.5.1.2 as follows:

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

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

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 as shown in Figure 7.12 and indeed our security system enforced the security requirements for file *abc.txt* by not allowing any process to access *abc.txt*.

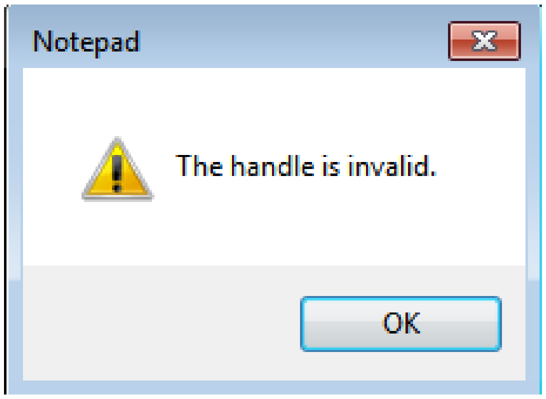
We conducted another experiment, where we identified three critical assets as *abc.txt*, *abd.txt*, and *abe.txt*. The security requirements were also identified for each file. The security requirement for *abc.txt* is confidentiality, integrity, and availability. Similarly, the security requirements for *abd.txt* are integrity and availability whereas only availability is required for *abe.txt*. Therefore and according to Equations 1 to 3 explained in Section 4.5.1.2, we have the followings:

$$C = \{(abc.txt, P_{abc}), (abd.txt, P_{abd}), (abe.txt, P_{abe})\} \tag{7.3}$$

$$P_{abc} = \{(confidentiality, \{Notepad\}), (availability, \{Notepad\})\} \tag{7.4}$$

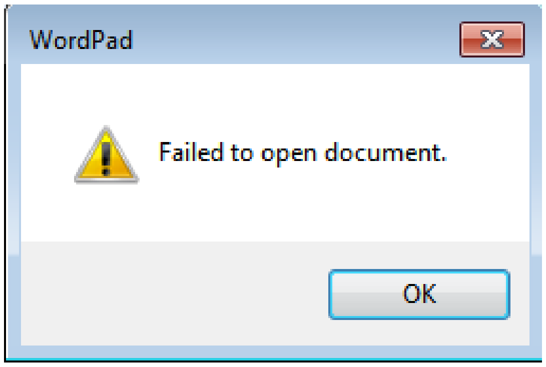
$$P_{abd} = \{(integrity, \{Notepad\}), (availability, \{Notepad\})\} \tag{7.5}$$

$$P_{abe} = \{(availability, \{Notepad\})\} \tag{7.6}$$


Notepad

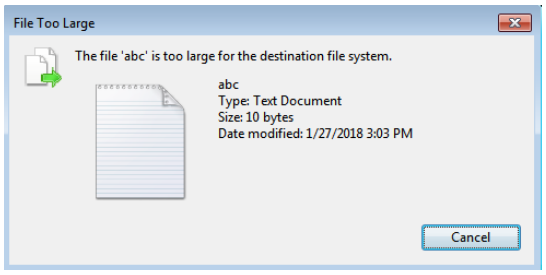
The handle is invalid.

(a) Could not modify abc.txt.


WordPad

Failed to open document.

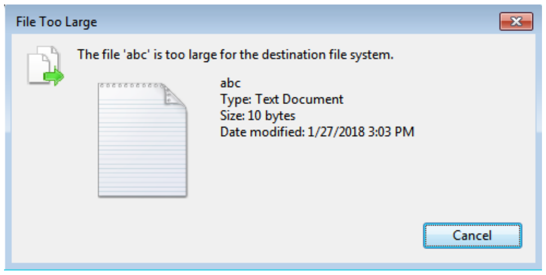
(b) Could not open abc.txt.


File Too Large

The file 'abc' is too large for the destination file system.

abc
Type: Text Document
Size: 10 bytes
Date modified: 1/27/2018 3:03 PM

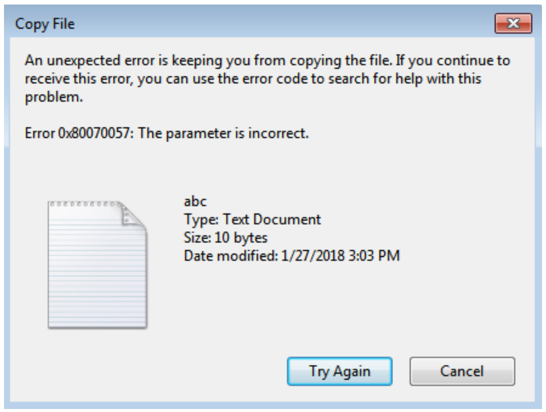
(c) Could not delete abc.txt.


File Too Large

The file 'abc' is too large for the destination file system.

abc
Type: Text Document
Size: 10 bytes
Date modified: 1/27/2018 3:03 PM

(d) Could not rename abc.txt.

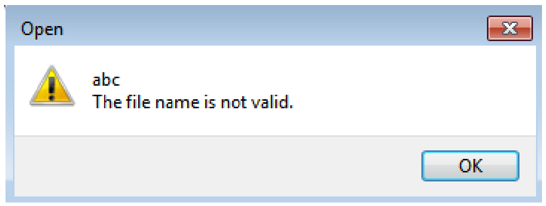

Copy File

An unexpected error is keeping you from copying the file. If you continue to receive this error, you can use the error code to search for help with this problem.

Error 0x80070057: The parameter is incorrect.

abc
Type: Text Document
Size: 10 bytes
Date modified: 1/27/2018 3:03 PM

(e) Could not copy abc.txt.


Open

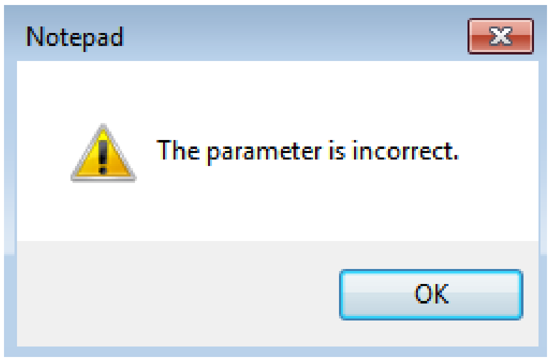
abc
The file name is not valid.

(f) Could not open abc.txt By WordPad from open menu.

Figure 7.12: File abc.txt is defended against any access from any process.

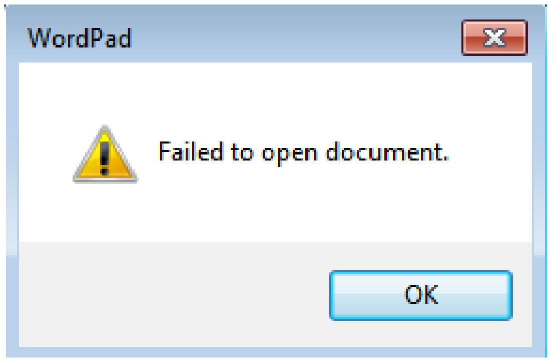
While our security system was running, we tried to access *abc.txt* within the virtual machine using various processes as shown in Figure 7.13 and indeed our security system enforced the security requirement for file *abc.txt* by not allowing any process to access *abc.txt*. We were successful in reading *abc.txt* using Notepad but we could not modify the file using Notepad as depicted in Figure 7.12(a).

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 as the results shown in Figures 7.13 and 7.12.



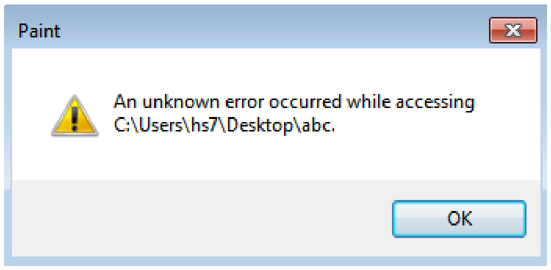
The parameter is incorrect.

(a) Notepad Could not open abc.txt.



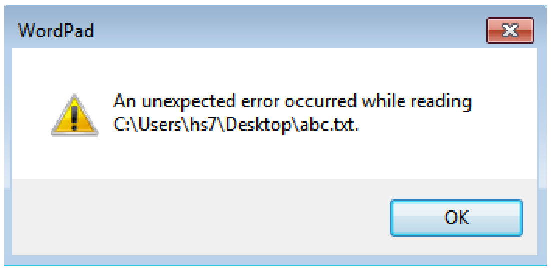
Failed to open document.

(b) WordPad could not Open abc.txt.



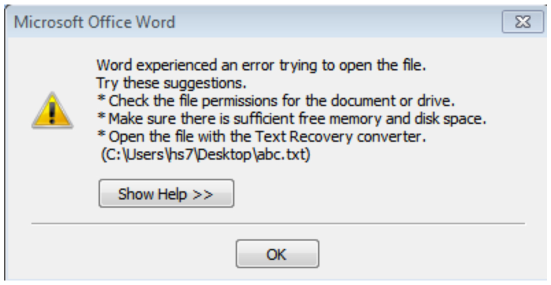
An unknown error occurred while accessing C:\Users\hs7\Desktop\abc.

(c) MS Paint could not Open abc.png



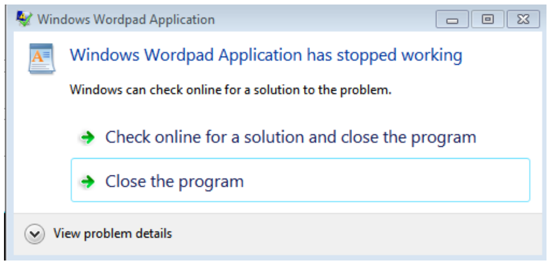
An unexpected error occurred while reading C:\Users\hs7\Desktop\abc.txt.

(d) WordPad could not Open abc.txt.



Word experienced an error trying to open the file.
Try these suggestions.
* Check the file permissions for the document or drive.
* Make sure there is sufficient free memory and disk space.
* Open the file with the Text Recovery converter.
(C:\Users\hs7\Desktop\abc.txt)

(e) Could not open abc.txt.


Windows Wordpad Application
Check online for a solution and close the program
Close the program
View problem details

Windows Wordpad Application has stopped working

Windows can check online for a solution to the problem.

(f) WordPad could not Open abc.txt.

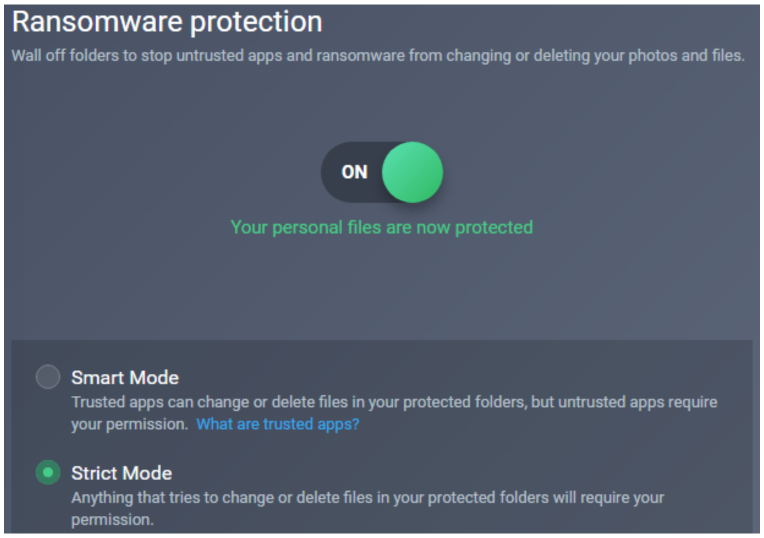
Figure 7.13: No process can access the critical asset abc.txt

7.8 Agility Against Real-World Ransomware

7.8.1 Anti Virus Guard

Environment Setup: We install Anti Virus Guard (AVG) on the guest machine [135] and activate ransomware protection option which is part of AVG to test the effect of real-world ransomware. After activation of ransomware protection, three main menus appeared to set the needed parameters for ransomware protection as in Figures 7.14, 7.15, and 7.17.

In Figure 7.14, you can choose between smart mode or strict mode. In smart mode, any untrusted application will require your permission to change or to delete any file inside your protected folders, while in strict mode all of the applications will ask for your permission.


Ransomware protection

Wall off folders to stop untrusted apps and ransomware from changing or deleting your photos and files.

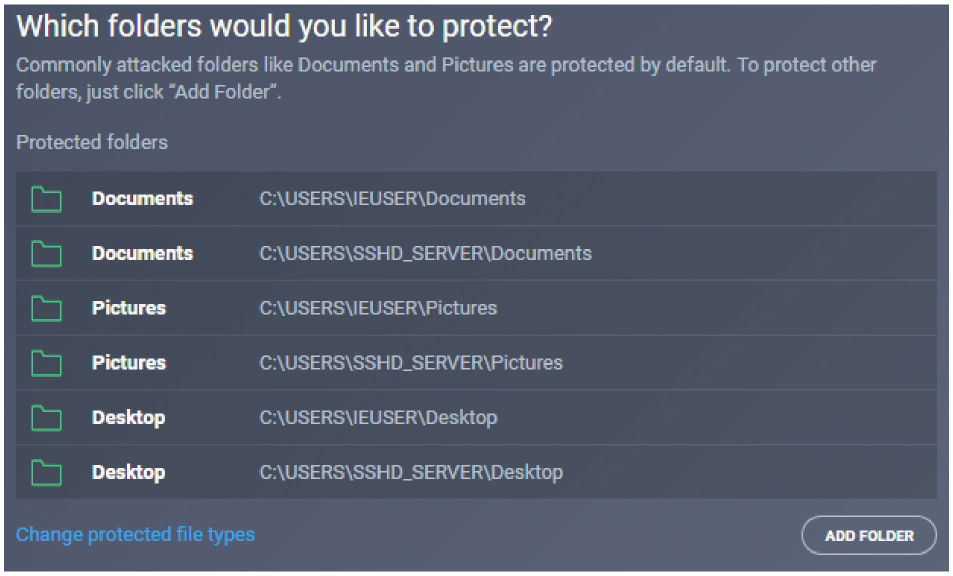
Your personal files are now protected

○ Smart Mode
Trusted apps can change or delete files in your protected folders, but untrusted apps require your permission. What are trusted apps?

● Strict Mode
Anything that tries to change or delete files in your protected folders will require your permission.

Figure 7.14: Protection modes in AVG.

In Figure 7.15, you can specify the folders to be protected. By default, AVG will set these folders as appeared in Figure 7.15. It is easy to add more or adjust the type of files to be secured inside the protected folder as shown in Figure 7.16. In Figure 7.17, you can specify the blocked or the allowed application so the allowed application will not any more wait for permission to access the files inside the selected folders while the blocked application will be blocked directly without notification.



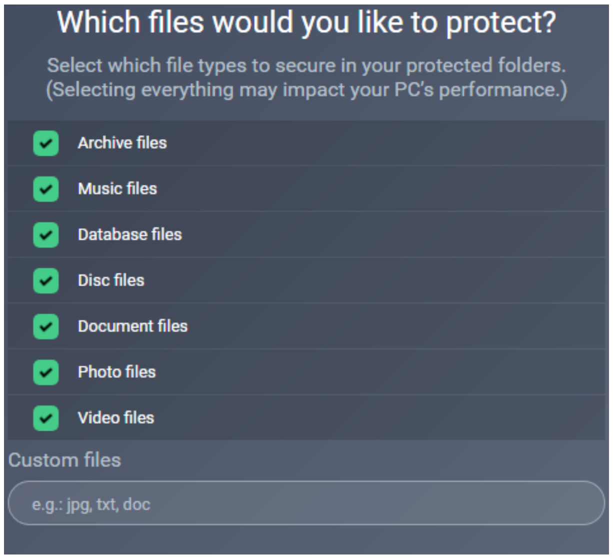
Which folders would you like to protect?

Commonly attacked folders like Documents and Pictures are protected by default. To protect other folders, just click "Add Folder".

Protected folders

Documents	C:\USERS\IEUSER\Documents
Documents	C:\USERS\SSHD_SERVER\Documents
Pictures	C:\USERS\IEUSER\Pictures
Pictures	C:\USERS\SSHD_SERVER\Pictures
Desktop	C:\USERS\IEUSER\Desktop
Desktop	C:\USERS\SSHD_SERVER\Desktop

Figure 7.15: Default protected folders in AVG.



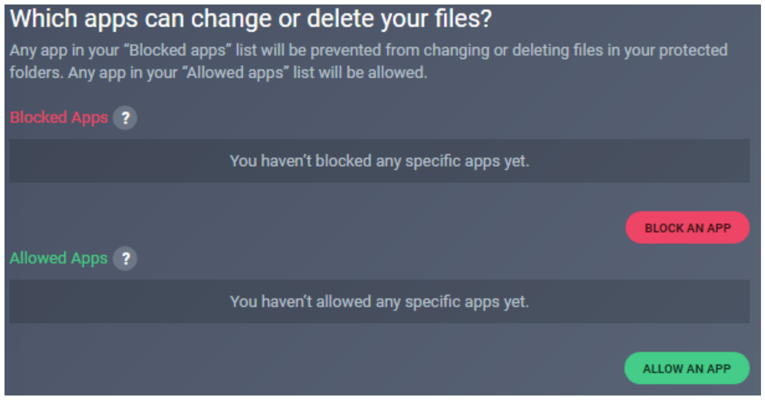
Which files would you like to protect?

Select which file types to secure in your protected folders. (Selecting everything may impact your PC's performance.)

- ✓ Archive files
- ✓ Music files
- ✓ Database files
- ✓ Disc files
- ✓ Document files
- ✓ Photo files
- ✓ Video files

Custom files

Figure 7.16: Files customization in AVG.


Which apps can change or delete your files?

Any app in your "Blocked apps" list will be prevented from changing or deleting files in your protected folders. Any app in your "Allowed apps" list will be allowed.

Blocked Apps

You haven't blocked any specific apps yet.

Allowed Apps

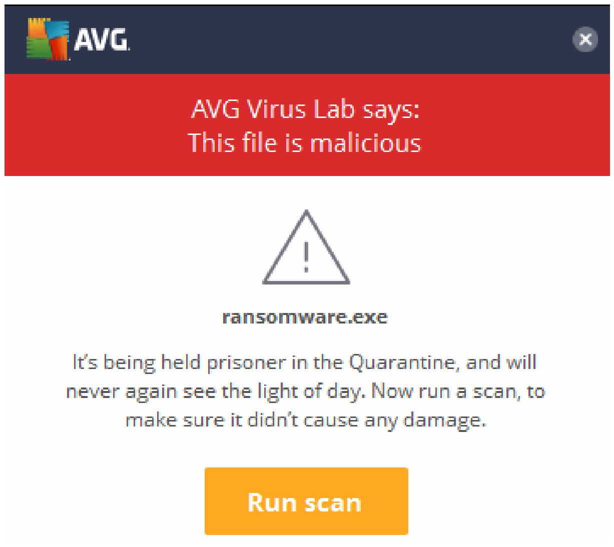
You haven't allowed any specific apps yet.

Figure 7.17: Blocked/ Non-Blocking in AVG.

Running Crypto-ransomware: We used the same crypto-ransomware [137] to test AVG ransomware protection environment. When we run ransomware.exe, a pop-up menu appears asking the user to either grant or deny permission for ransomware.exe to run. In the first experiment, we grant permission to ransomware.exe so it becomes one of the trusted applications. Therefore, all the files will be encrypted by ransomware.exe whether they are in protected or unprotected folder. In the second experiment, we deny permission to ransomware.exe and therefore AVG blocked ransomware.exe and a report is generated by AVG anti virus, as shown in Figures 7.18, 7.19, and 7.20.

In Figures 7.18 and 7.19, AVG anti virus generates a message detailing the possible malicious action of encryption generated by ransomware.exe. AVG anti virus also specifies the particular part of malicious action, which results in putting ransomware.exe in the quarantine.

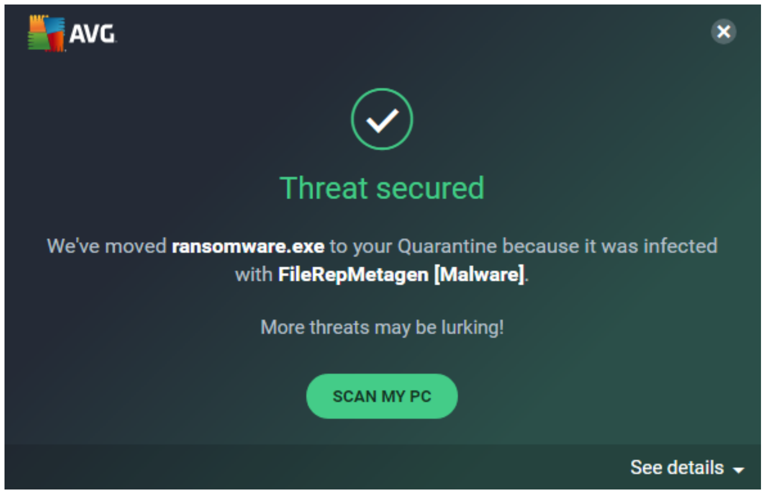
Later on when we try to download other files similar to ransomware.exe, AVG anti virus will instantly quarantined these files. By double clicking on ransomware.exe a dialogue box is shown as in Figure 7.20. This dialogue box is


AVG Virus Lab says:
This file is malicious

ransomware.exe

It's being held prisoner in the Quarantine, and will never again see the light of day. Now run a scan, to make sure it didn't cause any damage.

Figure 7.18: AVG pop-up menu.


Threat secured

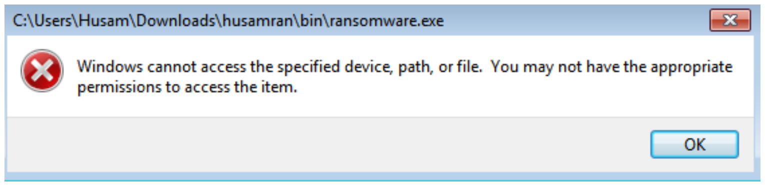
We've moved **ransomware.exe** to your Quarantine because it was infected with **FileRepMetagen [Malware]**.

More threats may be lurking!

Figure 7.19: AVG anti virus response.

generated by Windows assuring that ransomware.exe has been blocked and having no permission to run any more.

In the third experiment, we disable AVG anti virus, which is working as signature based, but not the ransomware protection option. By doing this, we ensure that the ransomware.exe will not be deleted nor quarantined. After that, we run ransomware.exe and a dialogue box pops up asking the user to block or allow the

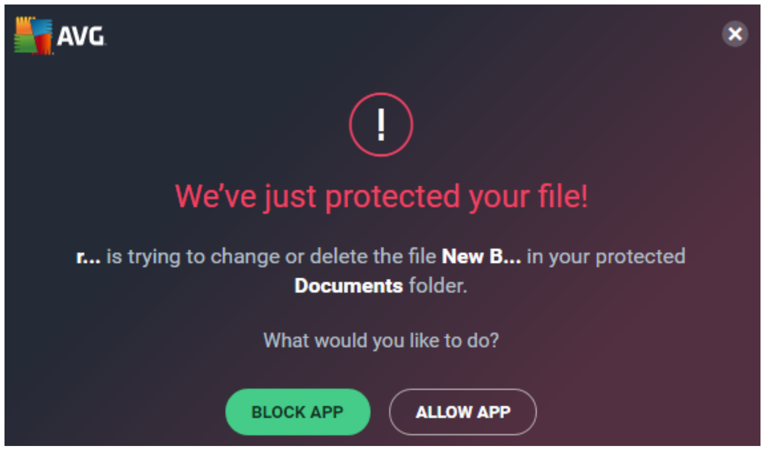


C:\Users\Husam\Downloads\husamran\bin\ransomware.exe

Windows cannot access the specified device, path, or file. You may not have the appropriate permissions to access the item.

Figure 7.20: Windows response.

ransomware.exe as in Figure 7.21. When we choose allow application, all the files will be encrypted whether they are in protected or unprotected folder. On the other hand, choosing block will allow ransomware.exe to encrypt all files expect those files in the protected folder. This is the desired and expected action of the ransomware protection part in AVG.



We've just protected your file!

r... is trying to change or delete the file **New B...** in your protected **Documents** folder.

What would you like to do?

Figure 7.21: AVG ransomware protection response.

7.8.2 Asset-Based Security System

We exposed our prototype to crypto-ransomware [137] 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 Equations 4, 6, and 7. The crypto-ransomware was able to encrypt all files, as shown in Figure 7.22, 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 NtDeleteFile. These are exactly the system calls mapped to integrity and availability as explained in Section 4.5.1.2. Similarly, the systems calls blocked from accessing “abe.txt” are exactly the ones corresponding to NtReadFile and NtSetInformationFile. These are exactly the system calls mapped to availability as also explained in Section 4.5.1.2. This shows that our security system defends identified critical assets by insuring 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.

7.8.3 AVG versus Our Security System

Our security system and AVG ransomware protection offer the protection to critical assets with some differences:

More Control: In our security system, we handle CIA which means if a process wants to read a critical file, then this will be denied by our security system but not by AVG as AVG protects against only modification and deletion.

More Safe: Compared with AVG, our security system works at the hypervisor level having all the VMI advantages. An attacker can know and can bypass AVG as shown in Figure 7.21. Furthermore, AVG information collection phase is

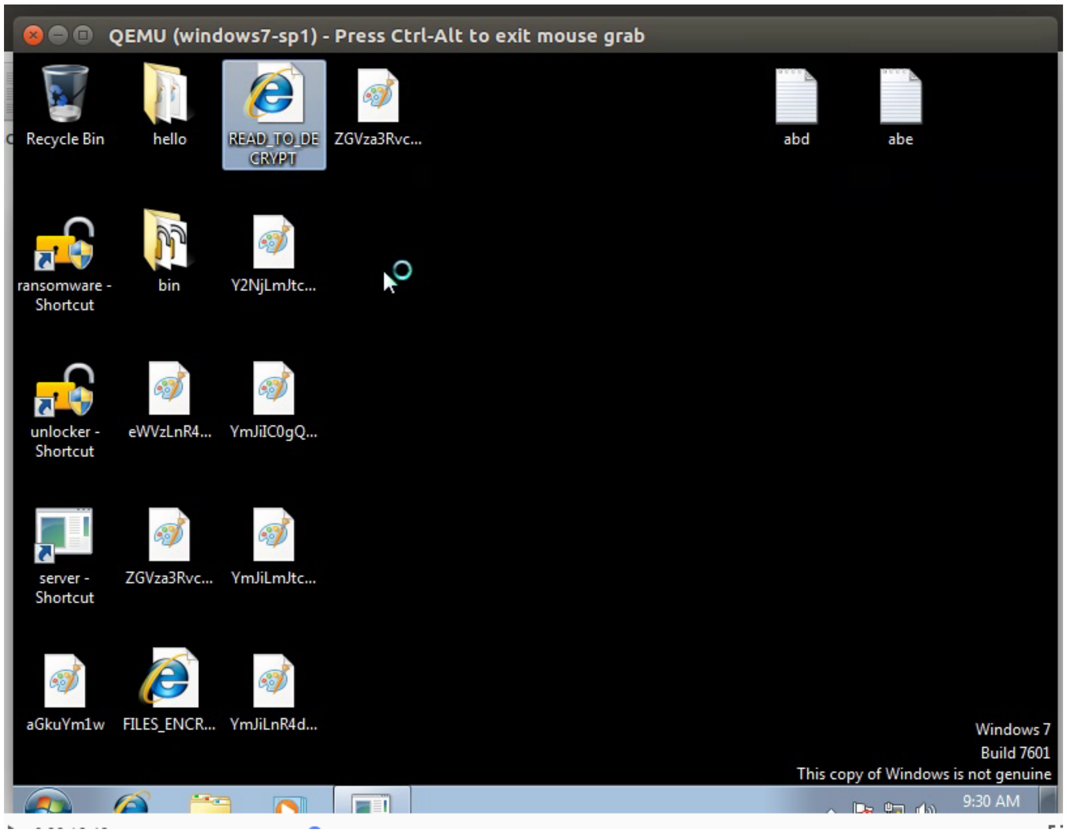


Figure 7.22: Crypto-ransomware encrypting all but the two critical files.

vulnerable to attacks as compared to our offline information collection phase as explained in section 4.5.1.

7.9 Security System Overhead

The performance penalty that comes with any new security approach is a crucial measure of the viability of that approach. If the security approach hinders performance to a degree where the system becomes unusable, then it's nonviable. Therefore, we dedicate this section to the overhead measurement of our security system.

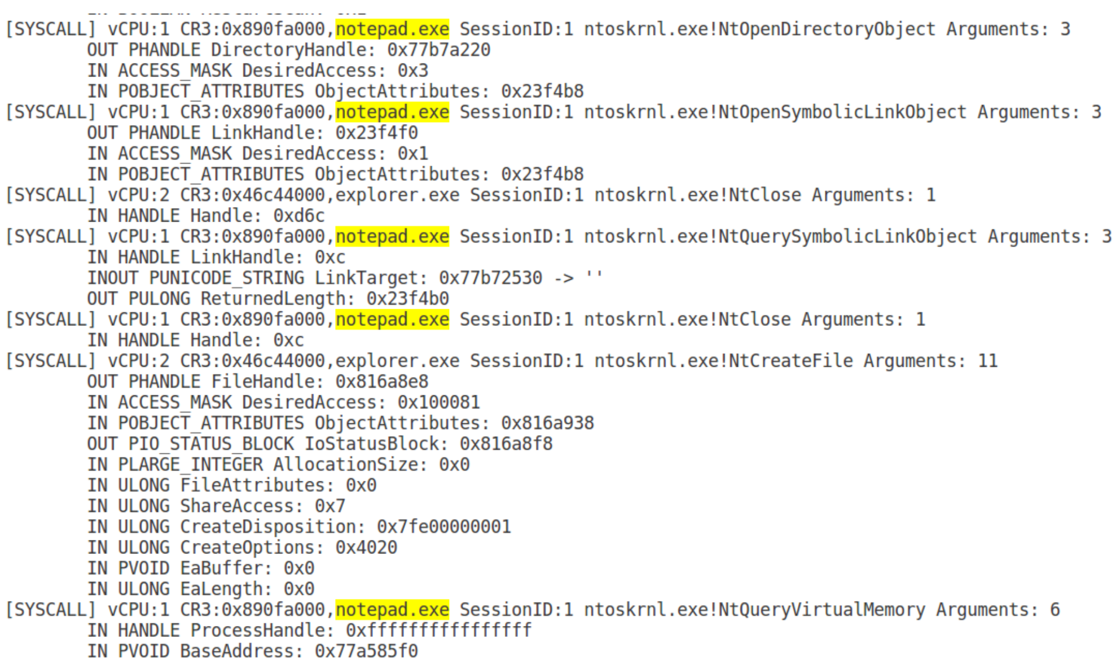
7.9.1 Generated System Calls

We conducted these sets of experiments to measure how many system calls will be captured by the monitoring phase. Subsequently, these system calls will be passed to the analysis phase for further processing. We monitored four applications, namely Notepad, WordPad, WinWord, and MS Paint. We count the system calls generated by these four applications using two different scenarios.

In the first scenario, we configure AppTimer to open the application and close the application. Then, a count of the system calls generated by the application will be provided to us by our monitoring phase as shown in Figure 7.23. Table 7.10 shows the number of system calls generated from this scenario under the "No Asset" column. This column is found under two categories: "Application Only" and "System Related". "Application Only" means the system calls generated from the application itself (i.e., the system call contains the application name). "System Related" means the system calls generated from the application itself as well as other system calls generated because of other system applications. For example, in Figure 7.23 there are two system calls generated by "explorer.exe" as a result of executing "Notepad.exe" or system applications.

Table 7.10: Number of NT system calls: monitoring phase.

Application	System Calls			
	Application Only		System Related	
	No Asset	With Asset	No Asset	With Asset
Notepad	1419	2069	3349	4855
WordPad	10226	13470	12306	17104
WinWord	24096	31929	31094	51276
MS Paint	11043	14993	18152	19420



```
[SYSCALL] vCPU:1 CR3:0x890fa000,notepad.exe SessionID:1 ntoskrnl.exe!NtOpenDirectoryObject Arguments: 3
        OUT PHANDLE DirectoryHandle: 0x77b7a220
        IN ACCESS_MASK DesiredAccess: 0x3
        IN POBJECT_ATTRIBUTES ObjectAttributes: 0x23f4b8
[SYSCALL] vCPU:1 CR3:0x890fa000,notepad.exe SessionID:1 ntoskrnl.exe!NtOpenSymbolicLinkObject Arguments: 3
        OUT PHANDLE LinkHandle: 0x23f4f0
        IN ACCESS_MASK DesiredAccess: 0x1
        IN POBJECT_ATTRIBUTES ObjectAttributes: 0x23f4b8
[SYSCALL] vCPU:2 CR3:0x46c44000,explorer.exe SessionID:1 ntoskrnl.exe!NtClose Arguments: 1
        IN HANDLE Handle: 0xd6c
[SYSCALL] vCPU:1 CR3:0x890fa000,notepad.exe SessionID:1 ntoskrnl.exe!NtQuerySymbolicLinkObject Arguments: 3
        IN HANDLE LinkHandle: 0xc
        INOUT PUNICODE_STRING LinkTarget: 0x77b72530 -> ''
        OUT PULONG ReturnedLength: 0x23f4b0
[SYSCALL] vCPU:1 CR3:0x890fa000,notepad.exe SessionID:1 ntoskrnl.exe!NtClose Arguments: 1
        IN HANDLE Handle: 0xc
[SYSCALL] vCPU:2 CR3:0x46c44000,explorer.exe SessionID:1 ntoskrnl.exe!NtCreateFile Arguments: 11
        OUT PHANDLE FileHandle: 0x816a8e8
        IN ACCESS_MASK DesiredAccess: 0x100081
        IN POBJECT_ATTRIBUTES ObjectAttributes: 0x816a938
        OUT PIO_STATUS_BLOCK IoStatusBlock: 0x816a8f8
        IN PLARGE_INTEGER AllocationSize: 0x0
        IN ULONG FileAttributes: 0x0
        IN ULONG ShareAccess: 0x7
        IN ULONG CreateDisposition: 0x7fe00000001
        IN ULONG CreateOptions: 0x4020
        IN PVOID EaBuffer: 0x0
        IN ULONG EaLength: 0x0
[SYSCALL] vCPU:1 CR3:0x890fa000,notepad.exe SessionID:1 ntoskrnl.exe!NtQueryVirtualMemory Arguments: 6
        IN HANDLE ProcessHandle: 0xffffffffffffffff
        IN PVOID BaseAddress: 0x77a585f0
```

Figure 7.23: Sample of generated system calls.

In the second scenario, we configure AppTimer to open the application, open a file within the application, and then close the application. The count of system calls generated by the application in this scenario will be provided to us by our monitoring phase similar to the ones provided in Figure 7.23.

We calculated the numbers shown in Table 7.10 by eliminating all system calls before the first system call generated by the application and also eliminating all system calls appeared after the last system call generated by the application. We then count the system calls in between. It should be also noted that each number shown in Table 7.10 is the average of 5 runs. Examining Figure 7.24, we observe the following:

- The number of system calls vary according to the application. For example, "Notepad" generated the least number of system calls since it is a basic text editor meant for basic plain text entry [141]. The article [141] verifies our findings in Table 7.10. The article states that WordPad is more advanced than Notepad and is meant for formatting and printing documents like WinWord, but not quite as advanced as WinWord.

- The number of system calls needed to only open the application is less than the system calls generated when the application used to open a file. This increase is due to extra privileged file-related I/O operations.

- Our monitoring phase is not only affected by the system calls generated by the monitored application but also by the system calls generated by other application during the monitoring period.

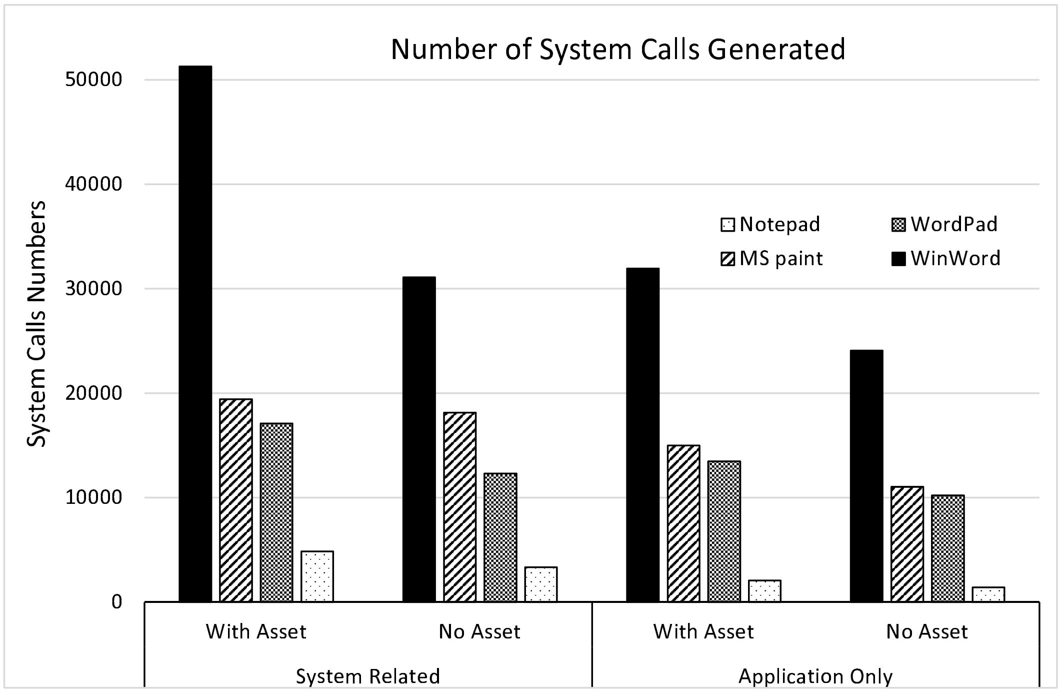


Figure 7.24: System calls generated by an application in different scenarios.

7.9.2 System Call Filtering

This section deals with filtering and passing only the needed systems calls to our analysis phase to insure the protection of assets. In Table 7.11, we count the number of system calls generated by specific application for specific NT system call. As shown in the table's first row, 11 NtCreateFile, 25 NtOpenFile, 2 NtReadFile, and 54 NtSetInformationFile system calls were generated by Notepad for a total of 92 system calls. These specific NT system calls are the ones that our analysis phase will process. That is, for the case of "Notepad", our analysis phase will process only 92 system calls out of 4855 systems calls generated by "Notepad". Similarly, the same explanation can be applied to the rest of the applications.

Table 7.11: Number of specific NT system calls: monitoring phase.

Application	System Calls			
	NtCreateFile	NtOpenFile	NtReadFile	NtSetInformationFile
Notepad	11	25	2	54
WordPad	67	138	34	116
WinWord	208	244	156	306
MS Paint	49	100	37	151

In Figure 7.25, we capture the number of selected system calls for certain application. We filter these captured system more and more to dig for certain system calls. We find that the number of these system calls related to the application itself, this result is expected as is, but a noticeable issue regarding the number of system calls in WordPad and MS Paint where they look like already the same in number when NtReadFile called but different when NtSetInformationFile generated. It is obvious from Figure 7.25 that MS Paint has more number of calls of type NtSetInformationFile rather than WordPad, while NtOpenFile and NtCreateFile generated more in WordPad compared to MS Paint.

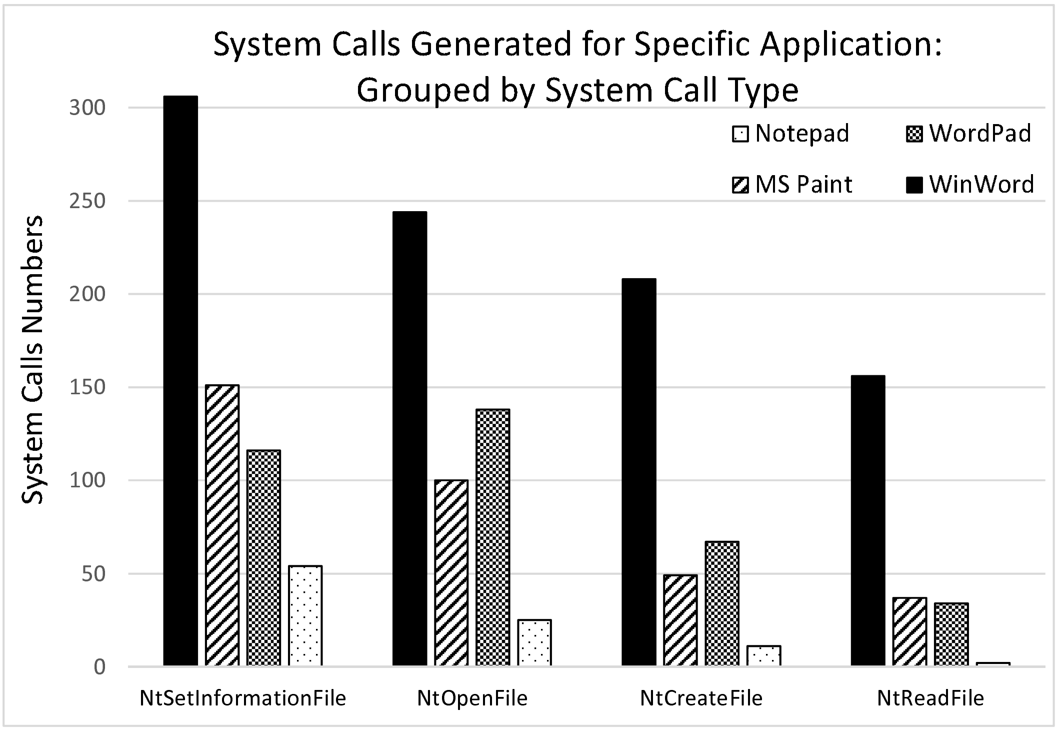


Figure 7.25: Selected system calls grouped by system call.

In Figure 7.26, we grouped the number of system calls by the application and not as in Figure 7.25 which grouped by system call. We can notice from Figure 7.26 that application WordPad looks differently than other applications when the number of NtSetInformationFile is not the most called system call.

7.9.3 System Security Response Time

In these set of experiments, we configure AppTimer to open and close an application. AppTimer will then generate a log file containing the response time as shown in Figure7.27

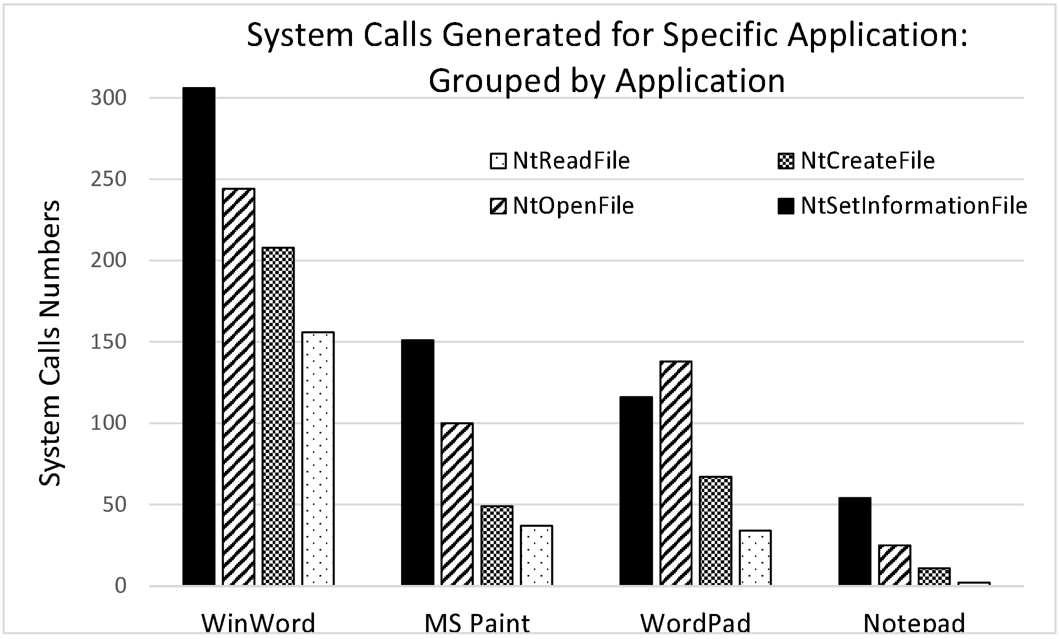
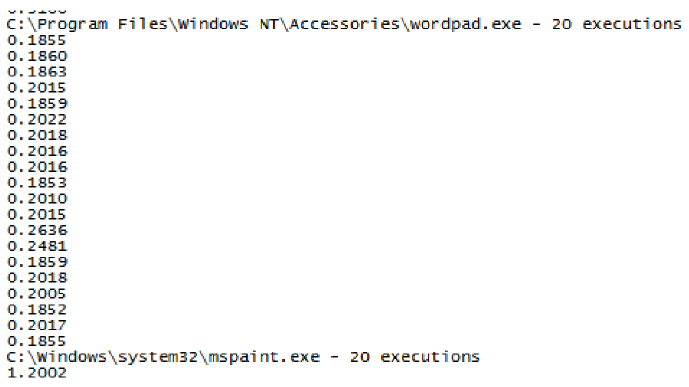


Figure 7.26: Selected system calls grouped by application.



```
C:\Program Files\Windows NT\Accessories\wordpad.exe - 20 executions
0.1855
0.1860
0.1863
0.2015
0.1859
0.2022
0.2018
0.2016
0.2016
0.1853
0.2010
0.2015
0.2636
0.2481
0.1859
0.2018
0.2005
0.1852
0.2017
0.1855
C:\Windows\system32\mspaint.exe - 20 executions
1.2002
```

Figure 7.27: Sample of AppTimer generated log file.

We measured the response time as shown in Table 7.12. The response time is the average of 20 runs and is measured in seconds. The table also shows the time measured when there is no asset accessed at all whether critical or non-critical. Table 7.12 contains the response time measured by “AppTimer” when specific application is opened. There are 5 columns, where all of them show the response time using the corresponding application without opening a file. Reference to Figure 4.10, the response time measured in Table 7.12 branches to

“Normal Operation” after step #2. The experiments shown in Table 7.12 are conducted as follows:

- Start AppTimer.
- Instruct AppTimer to open the application and record the start time.
- Once the application window opens, instruct AppTimer to record the end time.
- The difference between these two recorded times is the response time.

The first column shows the response time when there is no VMI (i.e., the monitoring phase is off). The second column contains the response time when the monitoring phase is active and the monitoring is done to the 6 specific system calls namely, NtWriteFile, NtReadFile, NtDeleteFile, NtSetInformationFile, NtOpenFile, and NtCreateFile. The third column measures the response time when all the NT system calls are monitored, while the fourth column shows the response time when the monitoring and analysis phases are active and done to the 6 specific system calls. The fifth column is taken when all the NT system calls are monitored and analyzed.

We measured the response time as shown in Table 7.13. The response time is the average of 20 runs and it is measured in seconds. The table also shows the response time measured when there is non-critical asset accessed. Table 7.13 contains the response time measured by “AppTimer” when specific application used to open non-critical asset. There are 5 columns, where all of them show

Table 7.12: Response time without file access.

Application	No Asset				
		VMI		VMI With Analysis	
	No VMI	Specific System Calls	NT System Calls	Specific System Calls	NT System Calls
Notepad	0.0447	0.0611	0.1380	0.0620	0.1624
WordPad	0.0461	0.0770	0.1851	0.0616	0.2326
WinWord	0.0622	0.1233	0.6527	0.1232	0.6527
MS Paint	0.0696	0.1236	0.5437	0.1244	0.6769

the response time using the corresponding application opening a file. Reference to Figure 4.10, the response time measured in Table 7.13 branches to "Normal Operation" after step #4. The experiments shown in Table 7.13 are conducted as follows:

- Start AppTimer.
- Instruct AppTimer to open the non-critical file using the application and record the start time.
- Once the non-critical file is opened, instruct AppTimer to record the end time.
- The difference between these two recorded times is the response time.

The first column shows the response time when there is no VMI (i.e., the monitoring phase is off). The second column contains the response time when the monitoring phase is active and the monitoring is done to the 6 specific system calls namely, NtWriteFile, NtReadFile, NtDeleteFile, NtSetInformationFile, NtOpenFile, and NtCreateFile. The third column measures the response time when all the NT system calls are monitored, while the fourth column shows the response time when the monitoring and analysis phases are active and done to the 6 specific system calls. The fifth column is taken when all the NT system calls are monitored and analyzed.

Table 7.13: Response time with non-critical file access.

Application	Non-Critical Asset				
		VMI		VMI With Analysis	
	No VMI	Specific System Calls	NT System Calls	Specific System Calls	NT System Calls
Notepad	0.0462	0.0595	0.1378	0.0626	0.1930
WordPad	0.0462	0.0770	0.2013	0.0777	0.2483
WinWord	0.0622	0.1234	0.6682	0.1216	0.6298
MS Paint	0.0774	0.1238	0.5516	0.1244	0.7162

We measured the response time as shown in Table 7.14. The response time is the average of 20 runs and it is measured in seconds. The table also shows the response time measured when there is critical asset. Table 7.14 contains the response time measured by "AppTimer" when specific application wants to open the critical asset. There are 4 columns, where all of them show the response time using the corresponding application trying to open the critical file. Reference to Figure 4.10, the response time measured in Table 7.14 either branches to "Normal Operation" or "Prevent system calls from continuity" after step #6. We will branch to "Normal Operation" when the access to the critical file is not prevented as shown in Table 7.14 column 3 and 4. If the access to the critical file is prevented, then the response time is shown in columns 1 and 2. The experiments shown in Table 7.14 are conducted as follows:

- Start AppTimer.

- Instruct AppTimer to open the critical file using the application and record the start time.
- The application may or may not open the file
  - if the application is allowed to open the critical file, then Once the critical file is opened, instruct AppTimer to record the end time.
  - if the application is not allowed to open the critical file, then Once the dialogue box, as shown Figure 7.13(c) appears, instruct AppTimer to record the end time.
- The difference between these two recorded times is the response time.

In Table 7.14, our security system (including the monitoring phase, analysis phase, and the decision phase) is active. The first column shows the response time when the monitoring is done to the 6 specific system calls namely, NtWriteFile, NtReadFile, NtDeleteFile, NtSetInformationFile, NtOpenFile, and NtCreateFile. The second column measures the response time when all the NT system calls are monitored. It should be noted that the response time shown in columns 1 and 2 is taken when access to the file is prevented.

The third column shows the response time when the monitoring is done to the 6 specific system calls while the fourth column is taken when all the NT system calls are monitored. It should be noted that the response time shown in columns 3 and 4 is taken when access to the file is allowed.

Table 7.14: Response time with critical file access.

Application	Critical Asset			
	Prevented		Not Prevented	
	Specific System Calls	NT System Calls	Specific System Calls	NT System Calls
Notepad	0.0613	0.1303	0.0612	0.1370
WordPad	0.0768	0.1854	0.0770	0.2011
WinWord	0.1211	0.6215	0.1230	0.6287
MS Paint	0.1237	0.4972	0.1240	0.5285

7.10 Performance Ratio

Table 7.15 shows the performance ratio for the 6 specific system calls. It should be noted that the performance ratio in:

- The first column is the result of dividing the corresponding entries from the fourth column in Table 7.12 by the first column in the same Table.
- The second column is the result of dividing the corresponding entries from the fourth column in Table 7.13 by the first column in the same Table.
- The third column is the result of dividing the corresponding entries from the first column in Table 7.14 by the first column in Table 7.13.
- The fourth column is the result of dividing the corresponding entries from the third column in Table 7.14 by the first column in Table 7.13.

Table 7.16 shows the performance ratio for NT system calls. It should be noted that the performance ratio in:

- The first column is the result of dividing the corresponding entries from the fifth column in Table 7.12 by the first column in the same Table.

- The second column is the result of dividing the corresponding entries from the fifth column in Table 7.13 by the first column in the same Table.
- The third column is the result of dividing the corresponding entries from the second column in Table 7.14 by the first column in Table 7.13.
- The fourth column is the result of dividing the corresponding entries from the fourth column in Table 7.14 by the first column in Table 7.13.

Table 7.15: Performance ratio for the 6 specific system calls.

	No Asset	Non-Critical Asset	Prevented	Not Prevented
Notepad	1.385	1.354	1.326	1.323
WordPad	1.336	1.681	1.662	1.665
WinWord	1.980	1.954	1.946	1.977
MS Paint	1.786	1.608	1.598	1.603

Table 7.16: Performance ratio for the NT system calls.

	No Asset	Non-Critical Asset	Prevented	Not Prevented
Notepad	3.631	4.177	2.820	2.965
WordPad	5.044	5.373	4.012	4.351
WinWord	10.493	10.126	9.991	10.106
MS Paint	9.725	9.259	6.427	6.831

From Figure 7.28 we notice that the latency here is not more than 10X in worst case and around 2X in best case.

From Figure 7.29 we notice that analyzing only specific system calls generates latency not more than 2X in worst case and around 1.3X in best case, which is a bigger improvement over the results shown in Figure 7.28.

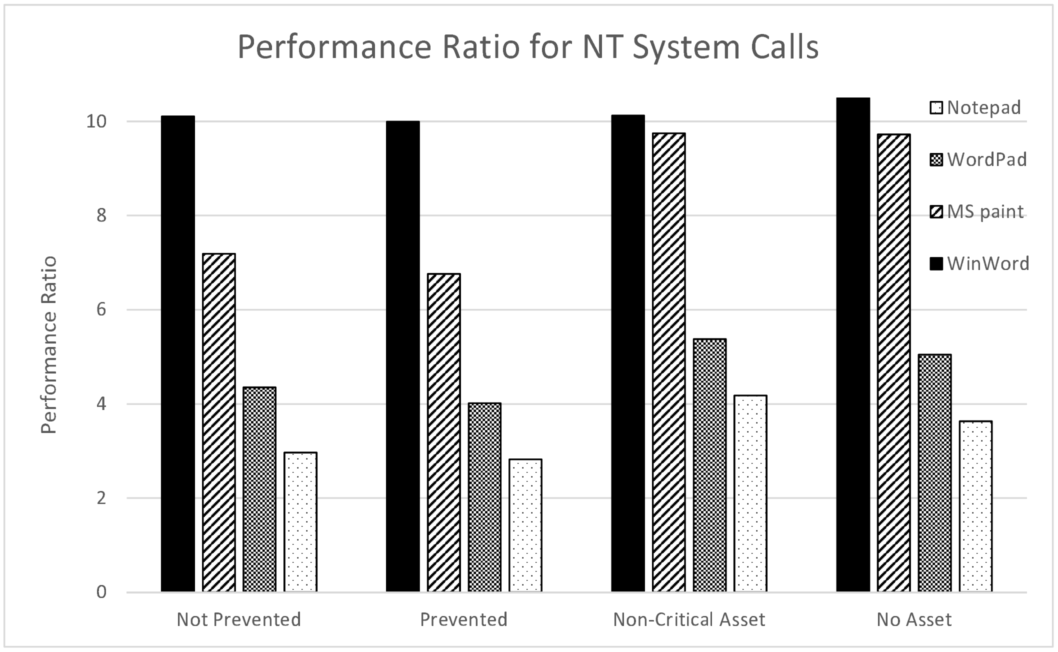


Figure 7.28: Performance ratio for the NT system calls.

We have to mention that our security system overhead using DRAKVUF 0.4 is approximately 38X in best case and up to 62X in worst case. This is very costly and further details of these results can be found in Appendix B. We are thankful to DRAKVUF version 0.5 [142] released in 30th June 2017 and Xen 4.9 [143] released in 28th June 2017. The new release includes core modifications to the in-depth execution tracing of arbitrary binaries as well as modifications to DRAKVUF system call plugin. These modifications enable DRAKVUF to print detailed arguments for Windows guest. As such, our security system is improved significantly and this is shown by examining the results obtained using DRAKVUF 0.4 and comparing them to the results obtained using DRAKVUF 0.5.

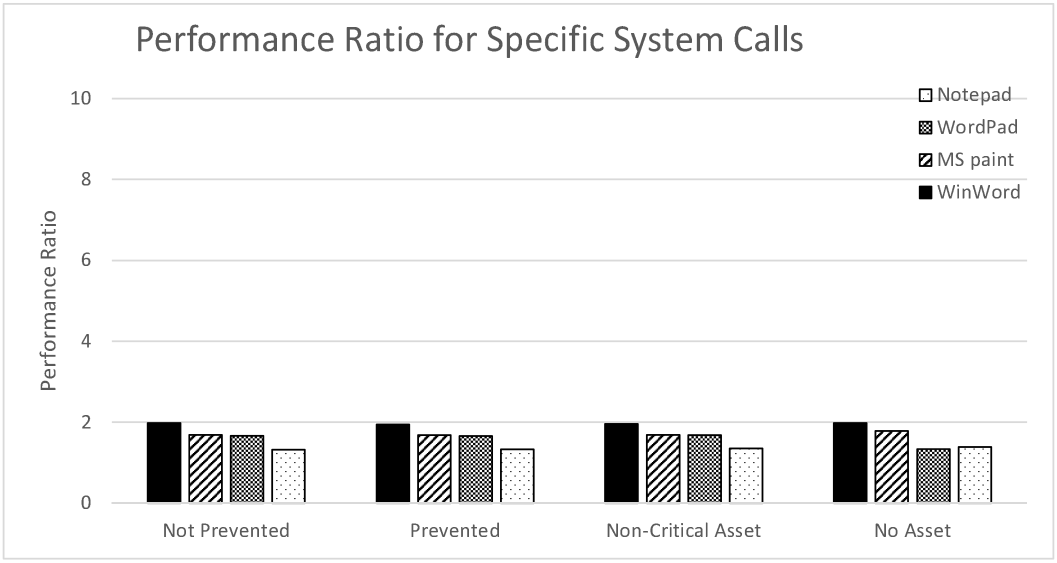
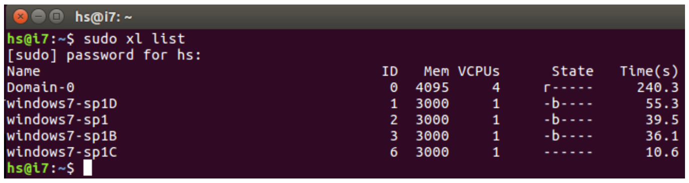


Figure 7.29: Performance ratio for the 6 specific system calls.

7.11 On the Scalability of the Security System

As previously mentioned that the tool used to conduct VMI is DRAKVUF. One major reason behind using this binary analysis tool is scalability. DRAKVUF achieves scalability as mentioned by [144] [130] and hence it is capable of analyzing large corpus of data with minimum overhead [145] [130]. Furthermore, critical assets are few can be counted on the fingers of one hand [18], so the analyzed data is not large.

Therefore, the scalability of our security system is inherited from DRAKVUF. In addition, we conducted an experiment where we run 4 different VMs on top of our host machine as shown in Figures 7.30 and 7.31. The objective of this set of experiments is to consider CPU utilization measurements for the proposed system and how they vary with an increase in the number of VMs served, and the number of attacks per VM. That is, we want to characterize how our security system scales with more VMs and more attacks per VM. Minimal consumption of the CPU and memory usage of the 4 VMs is shown in 7.32. That is, the CPU utilization when having 4 VMs is very similar to the CPU utilization when we run only one VM. Since VMs are seen as user processes by the hypervisor, the real scalability issue is at the hypervisor level and this has been proved by many published papers [144] [130] [145] [18].



```
hs@i7: ~
hs@i7:~$ sudo xl list
[sudo] password for hs:
Name                                        ID   Mem VCPUs      State   Time(s)
Domain-0                                     0  4095     4     r-----     240.3
windows7-sp1D                                1  3000     1     -b----      55.3
windows7-sp1                                 2  3000     1     -b----      39.5
windows7-sp1B                                3  3000     1     -b----      36.1
windows7-sp1C                                6  3000     1     ------      10.6
hs@i7:~$
```

Figure 7.30: Linux Dom0 and the 4 Windows VMs.

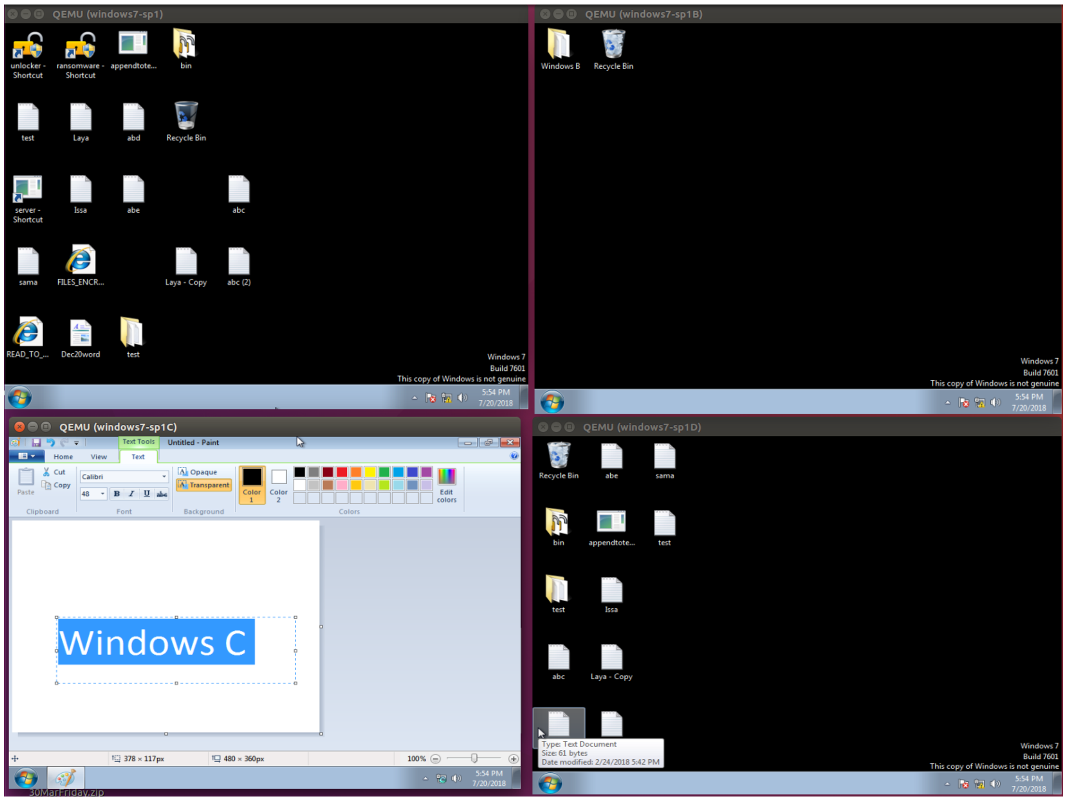
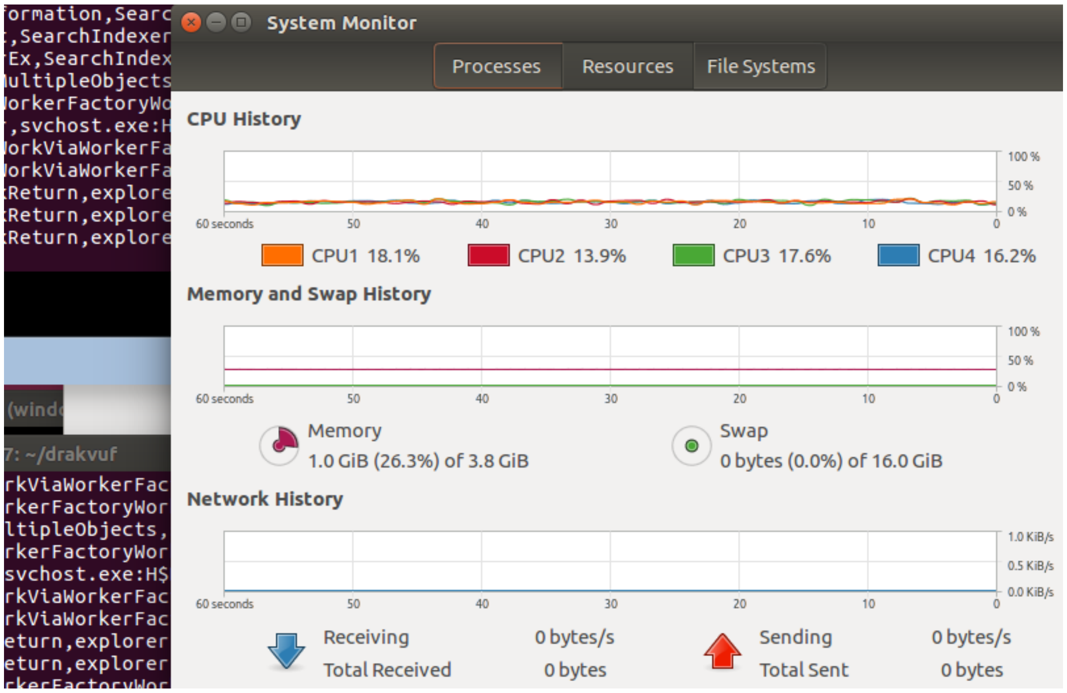


Figure 7.31: Running Multiple VMs.


```
ormation,Searc
,SearchIndexer
Ex,SearchIndex
ultipleObjects
orkerFactoryWo
,svchost.exe:H
orkViaWorkerFa
orkViaWorkerFa
Return,explore
Return,explore
Return,explore
```

```
(windo
7: ~/drakvuf
rkViaWorkerFac
rkerFactoryWor
ltipleObjects,
rkerFactoryWor
svchost.exe:H$
rkViaWorkerFac
rkViaWorkerFac
eturn,explorer
eturn,explorer
```

System Monitor

Processes | Resources | File Systems

CPU History

🟧 CPU1 18.1% 🟥 CPU2 13.9% 🟩 CPU3 17.6% 🟦 CPU4 16.2%

Memory and Swap History

Memory 1.0 GiB (26.3%) of 3.8 GiB

Swap 0 bytes (0.0%) of 16.0 GiB

Network History

Receiving 0 bytes/s
Total Received 0 bytes

Sending 0 bytes/s
Total Sent 0 bytes

Figure 7.32: Utilization of guests CPUs.

7.12 Performance Remarks

As outlined in [146], security systems follow an iterative process that can be broken down into four phases: predict, prevent, detect, and respond. Prediction assesses attack surface and prevention try to reduce such surface. Therefore, both of prediction and prevention rely on attacks. In our security system, we rely only on the defender side and therefore our security system does detection. The detection process is done based on pre-determined rules obtained from the reachability graph. This is very similar to file permission in any operating system.

As such, techniques such as precision and recall are unsuitable performance measures for our security model. Precision and recall consider actual and predicted instances. Since our security system does not consider the attacker side, we will never know the actual instances. Furthermore, we will never know the predicted instances since our security system is proactive as outlined in table 4.1.

We also studied some datasets used in security analysis for intrusion detection such as NSL-KDD, which contains datasets labeled as either normal or abnormal. If the dataset is labelled abnormal, then it contains one of 24 different kinds of attacks. These 24 attacks are grouped into 4 classes: Probe, DoS, R2L (Remote-to-Local), and U2R (User-to-Root) attacks [147] [148].

None of these classes target our assets file or process as the following:

- Probing or information collection is done before our security system is deployed. After that, our security system relies on capturing and monitoring system calls based on the collected information. If an attacker probes a VM, the collected information is safe since it is stored at the hypervisor level. Therefore, probing attacks are outside the scope of our security system.

- Availability as defined in [59] has three attributes namely, response time, expiration, and resource allocation. As long as the asset (a physical file in our system) exists, a resource is allocated. If the physical file exists but can not be accessed because the service (process) used to reach the file is unavailable, then the response time is affected. In this case, only the reachability to the file is affected.

  Therefore, in our security system, the availability is limited to only physical files. Our security system is not concerned with the availability of processes or services (DoS attacks) because our focus is to protect physical files and insure that these physical files are not detected and not available to unauthorized users.

- Both of U2R and R2L are considered as gaining unauthorized access to assets. Since our security system sits at the hypervisor level, it will not be affected by gaining unauthorized access in the guest OS. Furthermore, our security system starts defending from the system calls level which is a step before gaining unauthorized access because every action done by root/ordinary user generates a system call.

CHAPTER 8

CONCLUSION AND FUTURE DIRECTION

8.1 Overview

There is no doubt that there is a vicious battle between attackers and defenders. Researchers as well as security practitioners have developed defense systems. These defense systems are built to defend against certain attack(s). To design such defense systems, attack vectors need to be examined. For example, if we want to design a signature-based defense system, then we have to collect previous attack vectors and develop signatures for these attacks. Similarly, behavior-based security systems need to study the behavior of attack vectors to try and predict future attacks.

As such, a vital input parameter to these defense systems is the attack vector. The problem here is that the attack vector is designed by the attacker. If the attacker changes the attack vector, then the defense system becomes obsolete. In our work we are proposing to design a defense system that has no input parameters from attackers.

This led us to think differently by proposing an asset-based security system which is not inheriting the weaknesses in previous defense systems. Our security system depends only on the defender which leaves attackers in a learning phase regarding our security system. Our security system reacts prior to an attack.

This thesis argues that the trend of constantly chasing changing attack vectors is contributing to the continuity of attackers-led security vicious cycle. Attackers are leading and defenders are learning. This paradigm needs to be shifted in a way that defenders are leading and attackers are learning.

We started this thesis by studying the need for security in by collecting information about the destructive effects of attacks through cyber space in addition to the financial losses due to these attacks. Then, we surveyed the existing security solutions for such attacks. We also explored the reasons behind the success of these destructive attacks and the fail of the defense lines.

This thesis proposes an asset-based security system where security practitioners build their security systems based on information they own. The idea is to completely rely of ourselves in building security systems and require nothing from attackers. This way, attackers chase defenders which will not just level the security playing field but will give advantage to defenders.

Our security system consists of 4 phases namely, information collection, monitoring, decision, and feedback. Information collection phase prepares and collects the information needed for the security system to start its functionality by taking information from the asset owner about the critical assets and by building the reachability graph to reach those assets. The monitoring phase includes collecting system calls and parsing them to be processed in the decision phase which compares pre-collected information from the information collection phase with ongoing collected information from the monitoring phase to put the final decision for the captured system calls to be either terminated or passed. Finally, the feedback phase is important in applying and accommodating changes in the pre-collected information.

In this thesis, we propose a proactive asset based defense scheme using policies in a virtualized environment that can prevent illegal access to assets. Furthermore, we implemented the proposed scheme using Xen as a hypervisor and DRAKVUF as hypervisor level monitoring agent to monitor and prevent illegal access to assets within a guest operating system running windows.

As a proof of concept, we evaluated our security model using ransomware real-world attacks. The obtained results show that we achieved promising results with acceptable degradation in performance. Finally, we evaluated the performance of the solution and found it to be promising with some issues. As such, the response time overhead of our security system and the design of our security system can be accomplished as operating-system-independent.

We outlined 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.

8.2 Concluding Remarks

We started our security system prototype evaluation by a verification and validation step. In this step, we started by running the Task Manager within the virtual machine to provide the list of the running processes. Indeed, our security system prototype provided the same list of running processes. We also exposed our prototype to a real-world ransomware virus. Again our prototype was successful in protecting critical assets from the ransomware effect and encryption.

To measure our prototype overhead, we conducted a set of experiments to count the number of system calls to estimate the efforts done by our prototype. We also measured the time needed to open an application in different scenarios. The overhead of our security system prototype was acceptable.

We leave the reader with a comparison conclusion as shown in Table 8.1. The Table compares our security system to a non-asset-based security system. Our security system's strength comes from its asset-awareness property which enables proactive prevention security measures.

General attacks hit as many as possible targets and these attacks will not be effective when our proposed security system is deployed. This is because of

Table 8.1: Comparing Asset-Based and Non-Asset-Based Security Systems.

Phase		Security System	
		Non-Asset-Based	Asset-Based
Collector		Collects all	Collects all
Parser	System calls	Sequence of System calls	Subset
	Content	Content-unaware	Content-aware
Decision Maker		After the fact	Before the fact
Generator		Attack-Based	Defender-Based
Tuner		MTD-impossible	MTD-possible

customization to every standalone system will make the same attack vector useless against our system. For example, if the attack process is named “notepad.exe” and we used that process name as an allowed process to access our critical asset files in one system. The other systems can customize the name of the allowed process to be for example “mynotepad56.exe”. Therefore, “notepad.exe” malicious code process will not harm the critical assets. Moreover, if the attacker reconnaissance phase is done on our security system to know the process name we used as an allowed process, then the attacker will need to change his process name to the discovered one and must do that for every standalone defense system to hit as many targets as possible.

Also this mechanism can work in the same system. If we have 2 files using the same program or used by 2 users, we can provide different name for every user so we can make 2 processes such as renaming “notepad.exe” as “hisnotepad.exe” and “yournotepad.exe”. Then, we assign each one of these processes to different critical asset file. In this case, if the attacker succeeds in penetrating one of the asset files, the attacker has to initiate a new attack vector with new tuning to get the other file. this has to be done even if the 2 critical files are at the same system using the same program.

One main disadvantage of our security system is the following. If the attacker knows the name of the asset file and the process name accessing that file, then the attacker can utilize this information to bypass our system defense mechanism to gain access to the asset file. But this penetration is still defender-based game not attacker-based game even though that weakness can be waived by using MTD.

8.3 Asset-Based Security System Assumptions

We designed and implemented a prototype of our security system assuming the following:

- **Asset scope:** As shown in Table 2.5, assets can be physical, t-ICT, e-ICT, or e-information. Since our proposed asset-based security system is asset-based, the scope of assets that our proposed system is based on needs to be clearly defined. Assets span over a wide range of entities as defined in [18] and [53]. Files, processes, sockets, physical entities can be considered as assets. In this thesis, we are focusing on protecting information which is one of the main five assets mentioned in Table 2.5. Specifically, we are concerned with protecting files. Files can be protected by controlling the physical files themselves and protecting reaching those files through processes. As such, this thesis considers asset as file, and monitors the system calls to reach those files through processes. So, the scope of our asset is mainly starts from a file, then other processes and files could be added as assets through the reachability graph.

- **Information Collection Phase Immunity:** The information collection phase as discussed in section 4.5.1 is assumed to be done when the system is offline. That is, the information collection phase needs to be done before system deployment. This is done and assumed also in [18]. Once the information is collected, this information will be used to determine the critical assets (including files and processes) that the user needs to protect. Since, later security decisions will depend on the information collected, immunity to this phase is necessary for validity of the system. Therefore, each time information phase needs to be executed, it must be done offline.

- **Virtualization Environment:** Our security system monitors system calls utilizing VMI, which enables our security system to be agentless and does in-depth execution tracing of arbitrary binaries. Therefore, a virtualization environment is assumed. As such, our system system will be able to catch and analyze every system call generated by the guest OS by interacting with the hypervisor.

8.4 Asset-Based Security System Limitations

- **Availability scope:** Availability as defined in [59] has three attributes namely, response time, expiration, and resource allocation. As long as the asset (a physical file in our system) exists, a resource is allocated. If the physical file exists but can not be accessed because the service (process) used to reach the file is unavailable, then the response time is affected. In this case, only the reachability to the file is affected.

Therefore, in our security system, the availability is limited to only physical files. Our security system is not concerned with the availability of processes (services) because our focus is to protect physical files and insure that these physical files are not detected and not available to unauthorized users.

- **Guest Operating System Compromise:** If the attacker can compromise the guest OS, the system calls table can also can compromised. Therefore, an attacker can manipulate the system call table and then our system will not be able to catch critical system calls related to critical assets. As we are depending mainly on monitoring system calls in our system,then a compromise to the guest OS will harm our proposed security system.

- **Hypervisor Compromise:** Although seldom and complicated, attacks can reach hypervisors [149]. These attacks need sophisticated tools and skills but attacks knows as hyperjacking [149] could be done by (a) adding a rogue hypervisor on the top or beneath the original hypervisor and (b) directly controlling the original hypervisor. As such, if an attacker can reach the hypervisor by hyperjacking, then our system can be disabled making critical asset vulnerable to threats. System call interception techniques can not monitor the malicious drivers and rootkits which could be monitored using DRAKVUF at hypervisor level [150].

8.5 Directions for Future Work

Future directions for this work include, improving the performance, implementing facilities for more fine grained access control policies with MTD, and expanding this implementation to support Linux, Android or any other OS.

A fundamental paradigm shift is MTD systems where attack surfaces are always dynamically moving. MTD systems are only leveling the game between defenders are attackers and still require information on the attack scenarios in order to design proper adaptive and dynamic security systems. A key challenge to MTD systems is how to control such a dynamic environment so that defenders are not confused. As such, MTD systems need to impose this dynamic change from the perspective of attackers and not defenders.

To enhance the performance ratio in our security system, the concept of read-only file which used in Windows can be applied here to reduce the response time overhead. That is, using access mask bits [151] supported by the Operating System can relief the defender. Also if we customize our monitor system it can improve the performance significantly as it consume most of the time.

To expand this implementation to support Linux, Android or any other OS. For example in the case of Linux we need to install any Linux version as guest OS, then we need to monitor the system calls were generated from it, analyze them with their parameters. Also a definition for every system call is needed to be familiar with them. Finally we can apply our operational architecture to Linux.

REFERENCES

[1] R. van der Meulen, "Build Adaptive Security Architecture Into Your Organization," 30 Jun. 2017, Accessed: 14 Mar. 2018. [Online]. Available: https://www.gartner.com/smarterwithgartner/build-adaptive-security-architecture-into-your-organization/

[2] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating system concepts essentials.* John Wiley & Sons, Inc., 2014.

[3] I. Eusgeld, F. Freiling, and R. H. Reussner, *Dependability Metrics: GI-Dagstuhl Research Seminar, Dagstuhl Castle, Germany, October 5-November 1, 2005, Advanced Lectures.* Springer, 2008, vol. 4909.

[4] Q. Chen and R. A. Bridges, "Automated Behavioral Analysis of Malware A Case Study of WannaCry Ransomware," *arXiv preprint arXiv:1709.08753*, 2017.

[5] R. Canzanese, S. Mancoridis, and M. Kam, "System call-based detection of malicious processes," in *Software Quality, Reliability and Security (QRS), 2015 IEEE International Conference on.* IEEE, 2015, pp. 119–124.

[6] M. Kührer, J. Hoffmann, and T. Holz, "CloudSylla: Detecting Suspicious System Calls in the Cloud," in *Symposium on Self-Stabilizing Systems*. Springer, 2014, pp. 63–77.

[7] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid: behavior-based malware detection system for android," in *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*. ACM, 2011, pp. 15–26.

[8] S. Gupta and P. Kumar, "Taxonomy of cloud security," *International Journal of computer science, engineering and applications*, vol. 3, no. 5, p. 47, 2013.

[9] J. Leyden, "Hack on Saudi Aramco hit 30,000 workstations, oil firm admits," 2012, Accessed: 27 Jan. 2016. [Online]. Available: http://www.theregister.co.uk/2012/08/29/saudi_aramco_malware_attack_analysis/

[10] J. Fingas, "Stuxnet worm entered iran's nuclear facilities through hacked suppliers," 2014. [Online]. Available: http://www.engadget.com/2014/11/13/stuxnet-worm-targeted-companies-first/

[11] K. Zetter, "An Unprecedented Look at Stuxnet, the Worlds First Digital Weapon," 3 Nov. 2014, Accessed: 27 Feb. 2018. [Online]. Available: http://www.wired.com/2014/11/countdown-to-zero-day-stuxnet/

[12] Wikipedia, "Stuxnet," 5 Sep. 2017, Accessed: 27 Jan. 2018. [Online]. Available: https://en.wikipedia.org/wiki/Stuxnet

[13] A. Hamilton, “Top 10 security scandals of 2014,” 16 Dec. 2014, Accessed: 27 Jan. 2016. [Online]. Available: http://www.itproportal.com/2014/12/16/top-10-security-scandals-2014/

[14] Z. Whittaker, “These companies lost your data in 2015’s biggest hacks, breaches,” 2016. [Online]. Available: http://www.zdnet.com/pictures/biggest-hacks-security-data-breaches-2015/3/

[15] S. Kuranda, “The 10 biggest data breaches of 2015 (so far),” 27 Jul. 2015, Accessed: 27 Jan. 2016. [Online]. Available: http://www.crn.com/slide-shows/security/300077563/the-10-biggest-data-breaches-of-2015-so-far.htm/pgno/0/5

[16] H. Berghel, “On the problem of (cyber) attribution.” *IEEE Computer*, vol. 50, no. 3, pp. 84–89, 2017.

[17] L. Ponemon, “2015 cost of cyber crime study: Global,” 9 Oct. 2015, Accessed: 10 Aug. 2016. [Online]. Available: http://www8.hp.com/us/en/software-solutions/ponemon-cyber-security-report/

[18] S. A. Zonouz, R. Berthier, H. Khurana, W. H. Sanders, and T. Yardley, “Seclius: An information flow-based, consequence-centric security metric,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 2, pp. 562–573, 2015.

[19] C. Karr, "The IT security vicious cycle of "Assuming Compromise"," 10 Feb. 2015, Accessed: 25 Apr. 2017. [Online]. Available: http://www.itproportal.com/2015/02/10/security-vicious-cycle-assuming-compromise

[20] B. Christopher and T.-R. Eneken, "The Cyber Attack on Saudi Aramco," 1 Apr. 2013, Accessed: 27 Jan. 2016. [Online]. Available: https://www.iiss.org/en/publications/survival/sections/2013-94b0/survival--global-politics-and-strategy-april-may-2013-b2cc/55-2-08-bronk-and-tikk-ringas-e272

[21] A. Young and M. Yung, "Cryptovirology: Extortion-based security threats and countermeasures," in *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on.* IEEE, 1996, pp. 129–140.

[22] AlArabiya, "What is the Shamoon virus that has returned to hack Saudi networks?" 24 Jan. 2017, Accessed: 20 Apr. 2017. [Online]. Available: http://english.alarabiya.net/en/media/digital/2017/01/24/What-is-the-Shamoon-virus-that-has-returned-to-hack-Saudi-networks-.html

[23] U. Emmnauel and T. Mohammed, "Cyber security, threat intelligence: Defending the digital platform," *Journal of International Technology and Information Management*, vol. 26, no. 1, pp. 138–160, 2017.

[24] R. Brewer, "Advanced persistent threats: minimising the damage," *Network Security*, vol. 2014, no. 4, pp. 5–9, 2014.

[25] Norton, “The 8 Most Famous Computer Viruses of All Time,” 22 Feb. 2016, Accessed: 15 Feb. 2018. [Online]. Available: https://uk.norton.com/norton-blog/2016/02/the_8_most_famousco.html

[26] D. Jeffers, “Crime pays very well: Cryptolocker grosses up to $30 million in ransom,” 20 Dec. 2013, Accessed: 15 Feb. 2018. [Online]. Available: https://www.pcworld.com/article/2082204/crime-pays-very-well-cryptolocker-grosses-up-to-30-million-in-ransom.html

[27] S. Larson, “Massive cyberattack targeting 99 countries causes sweeping havoc,” 13 May 2017, Accessed: 25 May 2017. [Online]. Available: http://money.cnn.com/2017/05/12/technology/ransomware-attack-nsa-microsoft/

[28] J. BERR, “WannaCry ransomware attack losses could reach $4 billion,” 16 May 2017, Accessed: 15 Feb. 2018. [Online]. Available: https://www.cbsnews.com/news/wannacry-ransomware-attacks-wannacry-virus-losses/

[29] “Corporate Insider Threat Detection: Cyber Security Inside and Out,” 31 March 2015, https://www.cybersecurity.ox.ac.uk/research/projects. [Online]. Available: http://www.cs.ox.ac.uk/projects/CITD/

[30] Raytheon, “Triton apx suite,” 2016. [Online]. Available: https://www.forcepoint.com/products

[31] Kaspersky, “Ideas for the Future: the Best Projects of ‘CyberSecurity for the Next Generation - Russia & CIS Round 2014’ Chosen in Moscow,” Feb 2014.

[Online]. Available: http://www.kaspersky.com/about/news/business/2014/Ideas-for-the-Future-CyberSecurity-for-the-Next-Generation

[32] "Prevent prying eyes from seeing your critical data," 2015. [Online]. Available: http://www.optiolabs.com/product-overview/optioaware/

[33] DHS, "CSD Projects," 2016, Accessed: 2 Feb. 2017. [Online]. Available: http://www.dhs.gov/science-and-technology/csd-projects

[34] NoMoreRansom, "The No More Ransom Project," 2018, Accessed: 17 Feb. 2018. [Online]. Available: https://www.nomoreransom.org/en/index.html

[35] 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.

[36] 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 Conference on.* IEEE, 2015, pp. 738–746.

[37] L. Bilge and T. Dumitras, "Before we knew it: an empirical study of zero-day attacks in the real world," in *Proceedings of the 2012 ACM conference on Computer and communications security.* ACM, 2012, pp. 833–844.

[38] S. G. Salvatore Sinno, Franco Negri, "Designing an Adaptive Security Architecture with Unisys Stealth and LogRhythm," 2017, Accessed:

14 Mar. 2018. [Online]. Available: https://logrhythm.com/pdfs/3rd-party-whitepaper/designing-an-adaptive-security-architecture-with%-unisys-stealth-micro-and-logrhythm-3rd-party-white-paper.pdf

[39] M. Sajjad, A. A. Abbasi, A. Malik, A. B. Altamimi, and I. M. Alseadoon, "Classification and mapping of adaptive security for mobile computing," *IEEE Transactions on Emerging Topics in Computing*, 2018.

[40] R. Kissel, "Glossary of key information security terms," *NIST Interagency Reports NIST IR*, vol. 7298, no. 3, 2013.

[41] D. Schneider, "The state of network security," *Network Security*, vol. 2012, no. 2, pp. 14–20, 2012.

[42] R. W. Shirey, "Internet security glossary, version 2," 2007, [accessed Jan 27 2016]. [Online]. Available: https://tools.ietf.org/pdf/rfc4949.pdf

[43] CISCO, "Cybersecurity: Build Trust, Visibility and Resilience," 2011, Accessed: 22 Mar. 2016. [Online]. Available: http://www.cisco.com/c/dam/en_us/solutions/industries/docs/gov/cybersecurity_bvr_wp.pdf

[44] H. Goldman, R. McQuaid, and J. Picciotto, "Cyber resilience for mission assurance," in *Technologies for Homeland Security (HST), 2011 IEEE International Conference on.* IEEE, 2011, pp. 236–241.

[45] J. Rouse, "Mobile devices–the most hostile environment for security?" *Network Security*, vol. 2012, no. 3, pp. 11–13, 2012.

[46] C. Tankard, "Advanced persistent threats and how to monitor and deter them," *Network security*, vol. 2011, no. 8, pp. 16–19, 2011.

[47] "Resilience and Cyber Security of Technology in the Built Environment," 2013. [Online]. Available: https://www.cpni.gov.uk/documents/publications/2013/2013063-resilience_cyber_security_technology_built_environment.pdf?epslanguage=en-gb

[48] International Organization for Standardization, "ISO/IEC 13335-1:2004," Accessed: 1 Mar. 2018. [Online]. Available: https://www.iso.org/standard/39066.html

[49] Information Technology Laboratory, "NIST Glossary of Information Security Terms | CSRC," 2018, Accessed: 8 Mar. 2018. [Online]. Available: https://csrc.nist.gov/Glossary

[50] "Physical security, From Wikipedia, the free encyclopedia." [Online]. Available: https://en.wikipedia.org/wiki/Physical_security

[51] S. William, *Computer Security: Principles And Practice*. Pearson Education India, 2008.

[52] J. Pike, "GlobalSecurity.Org Physical-Security Challenges." [Online]. Available: http://www.globalsecurity.org/military/library/policy/army/fm/3-19-30/ch1.htm

[53] R. Von Solms and J. Van Niekerk, "From information security to cyber security," *computers & security*, vol. 38, pp. 97–102, 2013.

[54] C. Simmons, C. Ellis, S. Shiva, D. Dasgupta, and Q. Wu, “AVOIDIT: A cyber attack taxonomy,” 2009.

[55] A. Abbas, A. El Saddik, and A. Miri, “A comprehensive approach to designing internet security taxonomy,” in *Electrical and Computer Engineering, 2006. CCECE’06. Canadian Conference on.* IEEE, 2006, pp. 1316–1319.

[56] G. Hagan, “Glossary of defense acquisition acronyms & terms,” *Defense Acquisition University*, 2009.

[57] M. Uma and G. Padmavathi, “A Survey on Various Cyber Attacks and their Classification,” *Network Security*, vol. 15, no. 5, pp. 390–396, 2013.

[58] J. Mirkovic and P. Reiher, “A taxonomy of DDoS attack and DDoS defense mechanisms,” *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004.

[59] N. Rjaibi and L. B. A. Rabai, “Developing a Novel Holistic Taxonomy of Security Requirements,” *Procedia Computer Science*, vol. 62, pp. 213–220, 2015.

[60] T. Christian, “Security requirements reusability and the SQUARE methodology,” CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, Tech. Rep., 2010.

[61] K. Beckers, M. Heisel, and D. Hatebur, *Pattern and Security Requirements.* Springer, 2016.

[62] J. Andress, *The basics of information security: understanding the fundamentals of InfoSec in theory and practice.* Syngress, 2014.

[63] P. Savolainen, E. Niemela, and R. Savola, "A taxonomy of information security for service-centric systems," in *Software Engineering and Advanced Applications, 2007. 33rd EUROMICRO Conference on.* IEEE, 2007, pp. 5–12.

[64] W. V. Maconachy, C. D. Schou, D. Ragsdale, and D. Welch, "A model for information assurance: An integrated approach," in *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security*, vol. 310. United States Military Academy, West Point. IEEE, 2001.

[65] "Glossary of Security Terms, Definitions, and Acronyms," 8 Mar 2016 Nov 2012. [Online]. Available: http://www.cdse.edu/documents/cdse/Glossary_Handbook.pdf

[66] B. A. Forouzan and D. Mukhopadhyay, *Cryptography And Network Security (Sie).* McGraw-Hill Education, 2011.

[67] Hitachi, "Physical security solution," 2015, Accessed: 21 Feb. 2016. [Online]. Available: http://www.hitachi.com/businesses/infrastructure/product_site/pss/index.html

[68] "How can I set password control or access control on my webpages?" [Online]. Available: http://www.its.hku.hk/faq/infosys/web/personalweb/pwd-control

[69] R. S. Sandhu and P. Samarati, "Access control: principle and practice," *IEEE communications magazine*, vol. 32, no. 9, pp. 40–48, 1994.

[70] J.-H. Hwang, Y. Xing, U. Cetintemel, and S. Zdonik, "A cooperative, self-configuring high-availability solution for stream processing," in *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on.* IEEE, 2007, pp. 176–185.

[71] F. Wang, J. Qiu, J. Yang, B. Dong, X. Li, and Y. Li, "Hadoop high availability through metadata replication," in *Proceedings of the first international workshop on Cloud data management.* ACM, 2009, pp. 37–44.

[72] A. Juels and A. Oprea, "New approaches to security and availability for cloud data," *Communications of the ACM*, vol. 56, no. 2, pp. 64–73, 2013.

[73] D. R. Gibson, "Confidentiality, Integrity, Availability - Security Triad," 2015, Accessed: 22 Feb. 2016. [Online]. Available: http://blogs.getcertifiedgetahead.com/confidentiality-integrity-availability-security-triad/

[74] D. Gibson, *CompTIA Security+: Get Certified Get Ahead SY0-201 Study Guide.* BookSurge Publishing, 2009.

[75] G. K. Saha, "A single-version algorithmic approach to fault tolerant computing using static redundancy," *CLEI Electronic Journal*, vol. 9, no. 2, 2006.

[76] V. B. Balasubramanyn, G. Thamilarasu, and R. Sridhar, "Security Solution For Data Integrity In Wireless BioSensor Networks," in *Distributed Computing Systems Workshops, 2007. ICDCSW'07. 27th International Conference on.* IEEE, 2007, pp. 79–79.

[77] M. Hussain and M. Hussain, "A survey of image steganography techniques," 2013.

[78] A. Boukerche, K. El-Khatib, L. Xu, and L. Korba, "A Novel Solution for Achieving Anonymity in Wireless Ad hoc Networks," in *Proceedings of the 1st ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks.* ACM, 2004, pp. 30–38.

[79] M. Rouse, "Nonrepudiation Definition," Accessed: 3 Mar. 2016. [Online]. Available: http://searchsecurity.techtarget.com/definition/nonrepudiation

[80] J. Zhou and D. Gollman, "A fair non-repudiation protocol," in *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on.* IEEE, 1996, pp. 55–61.

[81] "Citi business online," 2016, Accessed: 24 Feb. 2016. [Online]. Available: https://businessaccess.citibank.citigroup.com/cbusol/landing/security.do

[82] C. Perrin, "The CIA triad," *Dostopno na: http://www. techrepublic. com/blog/security/the-cia-triad/488*, 2008.

[83] G. Stoneburner, C. Hayden, and A. Feringa, "Engineering principles for information technology security (a baseline for achieving security)," BOOZ-ALLEN AND HAMILTON INC MCLEAN VA, Tech. Rep., 2001.

[84] G. Pender-Bey, "The parkerian hexad." [Online]. Available: http://cs.lewisu.edu/mathcs/msisprojects/papers/georgiependerbey.pdf

[85] M. Chapple, D. Littlejohn Shinder, and E. Tittel, "TICSA Certification: Information Security Basics," *Pearson IT Certification*, 2002.

[86] D. G. Firesmith, "A taxonomy of security-related requirements," in *International Workshop on High Assurance Systems (RHAS'05)*, 2005, pp. 29–30.

[87] N. R. Mead and T. Stehney, *Security quality requirements engineering (SQUARE) methodology.* ACM, 2005, vol. 30, no. 4.

[88] C. Calderón and E. Marta, "A taxonomy of software security requirements," *Revista Avances en Sistemas e Informática*, vol. 4, no. 3, 2007.

[89] J. D. Howard and T. A. Longstaff, "A common language for computer security incidents," *Sandia National Laboratories*, vol. 10, p. 751004, 1998.

[90] R. P. van Heerden, B. Irwin, and I. Burke, "Classifying network attack scenarios using an Ontology," in *Proceedings of the 7th International Conference on Information-Warfare & Security (ICIW 2012)*, 2012, pp. 311–324.

[91] G. Loukas, D. Gan, and T. Vuong, "A taxonomy of cyber attack and defence mechanisms for emergency management networks," in *Pervasive Computing*

and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on. IEEE, 2013, pp. 534–539.

[92] K. S. Killourhy, R. A. Maxion, and K. M. Tan, "A defense-centric taxonomy based on attack manifestations," in *Dependable Systems and Networks, 2004 International Conference on.* IEEE, 2004, pp. 102–111.

[93] A. Shameli-Sendi, H. Louafi, W. He, and M. Cheriet, "A defense-centric model for multi-step attack damage cost evaluation," in *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on.* IEEE, 2015, pp. 145–149.

[94] A. Shameli-Sendi, M. Cheriet, and A. Hamou-Lhadj, "Taxonomy of intrusion risk assessment and response system," *Computers & Security*, vol. 45, pp. 1–16, 2014.

[95] A. Shameli-Sendi, R. Aghababaei-Barzegar, and M. Cheriet, "Taxonomy of information security risk assessment (ISRA)," *Computers & Security*, vol. 57, pp. 14–30, 2016.

[96] M. Alam, "Software Security Requirements Checklist," *International Journal of Software Engineering, IJSE*, vol. 3, no. 1, pp. 53–62, 2010.

[97] J. Jürjens, *Secure systems development with UML.* Springer Science & Business Media, 2005.

[98] A. Roy, D. S. Kim, and K. S. Trivedi, "Attack countermeasure trees (ACT): towards unifying the constructs of attack and defense trees," *Security and Communication Networks*, vol. 5, no. 8, pp. 929–943, 2012.

[99] M. Alazab, S. Venkatraman, P. Watters, and M. Alazab, "Zero-day malware detection based on supervised learning algorithms of API call signatures," in *Proceedings of the Ninth Australasian Data Mining Conference-Volume 121*. Australian Computer Society, Inc., 2011, pp. 171–182.

[100] M. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang, "Riskranker: scalable and accurate zero-day android malware detection," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*. ACM, 2012, pp. 281–294.

[101] P. M. Comar, L. Liu, S. Saha, P.-N. Tan, and A. Nucci, "Combining supervised and unsupervised learning for zero-day malware detection," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 2022–2030.

[102] 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.

[103] 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.

[104] A. Kharraz, S. Arshad, C. Mulliner, W. K. Robertson, and E. Kirda, "UN-VEIL: A Large-Scale, Automated Approach to Detecting Ransomware," in *USENIX Security Symposium*, 2016, pp. 757–772.

[105] 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*, 2018.

[106] K. Ganame, M. A. Allaire, G. Zagdene, and O. Boudar, "Network Behavioral Analysis for Zero-Day Malware Detection–A Case Study," in *International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments*. Springer, 2017, pp. 169–181.

[107] J. Chen, C. Wang, Z. Zhao, K. Chen, R. Du, and G.-J. Ahn, "Uncovering the face of android ransomware: Characterization and real-time detection," *IEEE Transactions on Information Forensics and Security*, 2017.

[108] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "An effective address mutation approach for disrupting reconnaissance attacks," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 12, pp. 2562–2577, 2015.

[109] X. Feng, Z. Zheng, D. Cansever, A. Swami, and P. Mohapatra, "A signaling game model for moving target defense," in *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*. IEEE, 2017, pp. 1–9.

[110] A. G. Bardas, S. C. Sundaramurthy, X. Ou, and S. A. DeLoach, "MTD CBITS: Moving Target Defense for Cloud-Based IT Systems," in *European Symposium on Research in Computer Security*. Springer, 2017, pp. 167–186.

[111] 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.

[112] J. Tian, R. Tan, X. Guan, and T. Liu, "Enhanced Hidden Moving Target Defense in Smart Grids," *IEEE Transactions on Smart Grid*, 2018.

[113] Will Brandon, "European Information Security Summit," 2016, Accessed: 10 Mar. 2018. [Online]. Available: https://biztechevents.co.uk/teiss/

[114] Dan Raywood, "TEISS - Consider Vulnerabilities, Assets and Threats to Understand Risk," 23 Feb. 2016, Accessed: 10 Mar. 2018. [Online]. Available: https://www.infosecurity-magazine.com/news/teiss-vulnerabilities-assets-risk/

[115] Microsoft, "Using nt and zw versions of the native system services routines," 2017. [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/hardware/ff565438(v=vs.85).aspx

[116] F. Rodríguez-Haro, F. Freitag, L. Navarro, E. Hernánchez-sánchez, N. Farías-Mendoza, J. A. Guerrero-Ibáñez, and A. González-Potes, "A summary of virtualization techniques," *Procedia Technology*, vol. 3, pp. 267–272, 2012.

[117] F. Bellard, "Qemu, a fast and portable dynamic translator." in *USENIX Annual Technical Conference, FREENIX Track*, vol. 41, 2005, p. 46.

[118] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *ACM SIGOPS operating systems review*, vol. 37, no. 5. ACM, 2003, pp. 164–177.

[119] C. D. Graziano, "A performance analysis of Xen and KVM hypervisors for hosting the Xen Worlds Project," 2011.

[120] WIKI, "Xen Project Software Overview," 24 Jan. 2017, Accessed: 20 Apr. 2017. [Online]. Available: https://wiki.xenproject.org/wiki/Xen_Project_Software_Overview

[121] Q. Jia, Z. Shen, W. Song, R. Van Renesse, and H. Weatherspoon, "Supercloud: Opportunities and challenges," *ACM SIGOPS Operating Systems Review*, vol. 49, no. 1, pp. 137–141, 2015.

[122] H. Fayyad-Kazan, L. Perneel, and M. Timmerman, "Benchmarking the performance of Microsoft Hyper-V server, VMware ESXi and Xen hypervisors," *Journal of Emerging Trends in Computing and Information Sciences*, vol. 4, no. 12, pp. 922–933, 2013.

[123] Stackoverflow, "virtualization - What's the differences between Xen, QEMU and KVM? - Stack Overflow," 24 Apr. 2012, Accessed: 18 Mar. 2018. [Online]. Available: https://stackoverflow.com/questions/10307323/whats-the-differences-between-xen-qemu-and-kvm

[124] A. Bulazel and B. Yener, “A Survey On Automated Dynamic Malware Analysis Evasion and Counter-Evasion: PC, Mobile, and Web,” in *Proceedings of the 1st Reversing and Offensive-oriented Trends Symposium.* ACM, 2017, p. 2.

[125] D. Song, D. Brumley, H. Yin, J. Caballero, I. Jager, M. G. Kang, Z. Liang, J. Newsome, P. Poosankam, and P. Saxena, “BitBlaze: A new approach to computer security via binary analysis,” in *International Conference on Information Systems Security.* Springer, 2008, pp. 1–25.

[126] BitBlaze Team, “TEMU: The BitBlaze Dynamic Analysis Component,” 2008, Accessed: 20 Mar. 2018. [Online]. Available: http://bitblaze.cs.berkeley.edu/

[127] A. Henderson, A. Prakash, L. K. Yan, X. Hu, X. Wang, R. Zhou, and H. Yin, “Make it work, make it right, make it fast: building a platform-neutral whole-system dynamic binary analysis platform,” in *Proceedings of the 2014 International Symposium on Software Testing and Analysis.* ACM, 2014, pp. 248–258.

[128] A. Henderson, L. K. Yan, X. Hu, A. Prakash, H. Yin, and S. McCamant, “Decaf: A platform-neutral whole-system dynamic binary analysis platform,” *IEEE Transactions on Software Engineering*, vol. 43, no. 2, pp. 164–184, 2017.

[129] L.-K. Yan and H. Yin, “Droidscope: Seamlessly reconstructing the os and dalvik semantic views for dynamic android malware analysis.” in *USENIX security symposium*, 2012, pp. 569–584.

[130] T. K. Lengyel, S. Maresca, B. D. Payne, G. D. Webster, S. Vogl, and A. Kiayias, “Scalability, fidelity and stealth in the DRAKVUF dynamic malware analysis system,” in *Proceedings of the 30th Annual Computer Security Applications Conference*. ACM, 2014, pp. 386–395.

[131] LibVMI Project, “LibVMI Project,” 2015, Accessed: 19 Mar. 2018. [Online]. Available: http://libvmi.com/

[132] Rekall Team, “Rekall Forensic,” 2016, Accessed: 19 Mar. 2018. [Online]. Available: http://www.rekall-forensic.com/

[133] M. Cohen, “Forensic analysis of windows user space applications through heap allocations,” in *Computers and Communication (ISCC), 2015 IEEE Symposium on*. IEEE, 2015, pp. 237–244.

[134] A. Socała and M. Cohen, “Automatic profile generation for live linux memory analysis,” *Digital Investigation*, vol. 16, pp. S11–S24, 2016.

[135] AVG Support Team, “Ransomware Protection - FAQs,” 2016, Accessed: 22 Mar. 2018. [Online]. Available: https://support.avg.com/SupportArticleView?l=en&urlname=Ransomware-Protection-FAQs

[136] Bitdefender Support Team, “How ransomware protection works in Bitdefender 2017,” 2017, Accessed: 22 Mar.

2018. [Online]. Available: https://www.bitdefender.com/support/how-ransomware-protection-works-in-bitdefender-2017-1733.html

[137] Mauri de Souza Nunes, “A POC Windows crypto-ransomware (Academic),” 5 Sep. 2016, Accessed: 09 Nov. 2017. [Online]. Available: https://github.com/mauri870/ransomware

[138] A. Akkas, C. N. Chachamis, and L. Fetahu, “Malware Analysis of WanaCry Ransomware,” 2017.

[139] 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 International Conference on*. IEEE, 2016, pp. 303–312.

[140] Microsoft, “Microsoft API and reference catalog,” 2018, Accessed: 02 Feb. 2018. [Online]. Available: https://msdn.microsoft.com/en-us/library/ms123401.aspx

[141] M. Crider, “What’s the Difference Between Notepad and WordPad in Windows?” 22 Mar. 2017, Accessed: 27 Mar. 2018. [Online]. Available: https://www.howtogeek.com/299490/whats-the-difference-between-notepad-and-wordpad-in-windows/

[142] Tklengyel, “Releases of DRAKVUF,” 2017. [Online]. Available: https://github.com/tklengyel/drakvuf/releases

[143] Xenproject.org, "Xen Project Release Features," 2017, Accessed: 14 Aug. 2017. [Online]. Available: https://wiki.xenproject.org/wiki/Xen_Project_Release_Features

[144] M. Bushouse and D. Reeves, "Hyperagents: Migrating host agents to the hypervisor," in *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy.* ACM, 2018, pp. 212–223.

[145] G. D. Webster, Z. D. Hanif, A. L. Ludwig, T. K. Lengyel, A. Zarras, and C. Eckert, "Skald: a scalable architecture for feature extraction, multi-user analysis, and real-time information sharing," in *International Conference on Information Security.* Springer, 2016, pp. 231–249.

[146] F-Secure, "RANSOMWARE HOW TO PREDICT, PREVENT, DETECT & RESPOND," 5 Nov. 2016, Accessed: 20 July. 2018. [Online]. Available: https://www.f-secure.com/documents/996508/1030745/Ransomware_how_to_ppdr.pdf

[147] S. Revathi and A. Malathi, "A detailed analysis on nsl-kdd dataset using various machine learning techniques for intrusion detection," *International Journal of Engineering Research and Technology. ESRSA Publications*, 2013.

[148] L. Dhanabal and S. Shantharajah, "A study on nsl-kdd dataset for intrusion detection system based on classification algorithms," *International Journal*

of Advanced Research in Computer and Communication Engineering, vol. 4, no. 6, pp. 446–452, 2015.

[149] A. Sancheti, “Resolving security issues in the virtual machine file system,” *International Journal of Engineering Research and General Science*, vol. 3, no. 5, pp. 366–370, 2015.

[150] P. Mishra, E. S. Pilli, V. Varadharajan, and U. Tupakula, “Intrusion detection techniques in cloud environment: A survey,” *Journal of Network and Computer Applications*, vol. 77, pp. 18–47, 2017.

[151] Microsoft, “How permissions work,” 2013. [Online]. Available: https://technet.microsoft.com/en-us/library/cc783530(v=ws.10).aspx

APPENDIX A

INSTALLATION GUIDE

UPDATED 7/10/2017 DRAKVUF now requires Xen 4.9. Please pay attention to the updated VM configuration specifying the altp2m option for a domain.

The following packages are normally required to build Xen and DRAKVUF on Debian based Linux distros. The system has been mainly tested on Debian Jessie and Ubuntu 14.04 LTS.

sudo apt-get install wget git bcc bin86 gawk bridge-utils iproute libcurl3 libcurl4-openssl-dev bzip2 pciutils-dev build-essential make gcc clang libc6-dev libc6-dev-i386 linux-libc-dev zlib1g-dev python-dev python-pip libncurses5-dev patch libvncserver-dev libssl-dev libsdl-dev iasl libbz2-dev e2fslibs-dev git-core uuid-dev ocaml libx11-dev bison flex ocaml-findlib xz-utils gettext libyajl-dev libpixman-1-dev libaio-dev libfdt-dev cabextract libglib2.0-dev autoconf automake libtool check libjson-c-dev libfuse-dev checkpolicy liblzma-dev autoconf-archive kpartx python-capstone

We will be installing a slightly modified version of Xen 4.8 that includes a built-in XSM policy required for DRAKVUF.

```
cd

git clone https://github.com/tklengyel/drakvuf

cd drakvuf

git submodule init

git submodule update

cd xen

./configure –enable-githttp

make -j4 dist-xen

make -j4 dist-tools
```

APPENDIX B

DRAKVUF VERSION 0.4

Here we added the experiment result from the previous version of DRAKVUF version 0.4 which is related to system security response time, unfortunately this result are no more valid also we could not complete some of the table as DRAKVUF version 0.4 is not working anymore.

Table B.1 time shown in seconds consumed in Guest OS when application opened alone, shows time is seconds needed to run a specific program in different situations and scenarios we run each one of them about 100 time, then we took the average, we have three columns, all of them shows the time for all application solo i.e. without opening a file using that application, the steps will be done according to the approach figure are step one and step two, we are using a program called APPTIMER to record the time, it works like the following procedure: Start Apptimer. Instruct Apptimer to open the application and record the START time. Once the application window opens, instruct Apptimer to record the END time. The difference between these columns is that the time recorded in the first column is when there is no monitoring at all i.e. the plugins are off, the second column contains the time when the monitoring done to specific system calls namely they are NtWriteFile, NtReadFile, NtDeleteFile, NtSetInformationFile, NtOpenFile, and NtCreateFile. The third column time is taken when all the NT system calls are monitored.

Table B.1: Response time Without File Access.

Application	Without File Access		
	Code off	Specific System Call	NT System Calls
Notepad	0.042	1.591	1.767
WordPad	0.045	2.134	2.306
MS Paint	0.070	3.220	3.884

In Table B.2 time is shown in seconds consumed in Guest OS when noncritical file accessed by an application, there are three columns which record the time needed to access a noncritical file by an application, the step done in this table starts from step one to up to step four, following the next procedure: Start Apptimer. Instruct Apptimer to open the application and record the start time. Once the application window opens the noncritical file, instruct Apptimer to record the END time.

Table B.2: Response time when noncritical file accessed.

Application	Without File Access		
	Code off	Specific System Call	NT System Calls
Notepad	0.040	1.669	1.693
WordPad	0.050	2.627	2.821
MS Paint	0.067	3.315	3.951

In Table B.3 time is shown in seconds consumed in Guest OS when critical file accessed the time shown for critical files when prevented from access and when allowed, for prevented case there is no access to the file at all, to ensure this criterion we need the seven steps from the first one to the last one with the following procedure: Start Apptimer. Instruct Apptimer to open the application and record the start time. Once the application window opens, the application tries to access the critical file (i.e., open the critical file). Once a message is displayed to indicate access denial, Apptimer records the END time, for level three case where the process has given full access to the file, six steps is enough to ensure the criteria with the following procedure: Start Apptimer. Instruct Apptimer to open the application and record the start time. Once the application window (if allowed) opens the critical file, instruct Apptimer to record the END time.

Table B.3: Response time when critical file accessed.

App.	Prevent		No Prevent	
	Specific System Calls	NT System Calls	Specific System Calls	NT System Calls
Notepad	1.707	1.783	1.659	1.801
WordPad	2.751	4.054	2.595	2.888
MS Paint	3.389	4.192		

Vitae

- Name: Husam Issa Mohammad Suwad
- Nationality: Jordanian
- Date of Birth: 24 Aug 1980
- Email: *hsuwad@gmail.com*
- Permanent Address: Halhul - Hebron - Palestine
- Education:
  - Master of Scientific computational, V. Good accumulative average of 85%, College of Higher education, BirZeit University (BZU).
  - Bachelor of Computer System Engineering, with a distinctive accumulative average of 86.6%, College of Engineering & Technology, Palestine Polytechnic University (PPU).
- Current Work: Assistant Professor - Palestinian Technical University - Kadoorie - Al-Aroub branch (PTU).