Cover sheet for submission of work for assessment

SWINBURNE UNIVERSITY OF TECHNOLOGY

UNIT DETAILS

Unit name	IT Security			Class day/time	
Unit code	COS30015	Assignment no.	1	Due date	4/7/2021
Name of lecturer/teacher	Dr. Nguyen Dai Tho				
Tutor/marker's name					

Office use only

Faculty or school date stamp

STUDENT(S)

	Family Name(s)	Given Name(s)	Student ID Number(s)
(1)	Nguyen	Minh Hoang	103295032
(2)			
(3)			
(4)			
(5)			
(6)			

DECLARATION AND STATEMENT OF AUTHORSHIP

1. I/we have not impersonated, or allowed myself/ourselves to be impersonated by any person for the purposes of this assessment.
2. This assessment is my/our original work and no part of it has been copied from any other source except where due acknowledgement is made.
3. No part of this assessment has been written for me/us by any other person except where such collaboration has been authorised by the lecturer/teacher concerned.
4. I/we have not previously submitted this work for this or any other course/unit.
5. I/we give permission for my/our assessment response to be reproduced, communicated, compared and archived for plagiarism detection, benchmarking or educational purposes.

I/we understand that:

6. Plagiarism is the presentation of the work, idea or creation of another person as though it is your own. It is a form of cheating and is a very serious academic offence that may lead to exclusion from the University. Plagiarised material can be drawn from, and presented in, written, graphic and visual form, including electronic data and oral presentations. Plagiarism occurs when the origin of the material used is not appropriately cited.

Student signature/s

I/we declare that I/we have read and understood the declaration and statement of authorship.

(1)	Minh Hoang Nguyen	(4)	
(2)		(5)	
(3)		(6)	

Further information relating to the penalties for plagiarism, which range from a formal caution to expulsion from the University is contained on the Current Students website at **www.swin.edu.au/student/**

Copies of this form can be downloaded from the Student Forms web page at **www.swinburne.edu.au/studentforms/**

ASSIGNMENT 1 – MODELS AND SYSTEMS TO PREVENT SECURITY VULNERABILITY

Swinburne University of Technology (Vietnam)

ABSTRACT

Nowadays, Security in Information Technology is become one of the most global concerns due to the many new technology is presented and applied to the daily life. As a result, the number of vulnerabilities existed in the internet is increased rapidly day-by-day, and many security professionals throughout the world are working to find out more effective ways to manage and reduce both quantity and severity of the vulnerability's impact in cyberspace. This literature review will focus on the topics related to the security vulnerability, summarize and evaluate the methods about the way to assess and reduce vulnerability holes, in order to point out the key limitations and challenges that still existed. This literature review will dive into many categories and aspects of vulnerability such as Operating Systems, Network, Software, Cloud and Virtualization. Each method for each vulnerability's category has their own strengths and weakness, we will mention about it more specific in this assignment.

Minh Hoang Nguyen – Student ID: 103295032

COS30015 – IT Security

Table of Contents

I. Introduction: Security Vulnerability and Impact

Nowadays, more and more Internet users, domain administrators and software engineers tend to concern about the security assessment and security level of their information and communication systems as well as the software and web application which they are developing. From many recent security incidents happened to US companies such as the cyberattack to the largest US pipeline company "Colonial Pipeline" which made the company to fork over 4.4 million dollars to end and take back the control permission for their system, it raises the emergency alarm to many organizations about the cyber security level of their own system, and proper security measures need to be applied in order to mitigate risks and vulnerabilities. One of the main reasons to answer the question why intruders can always exploit and gain malicious intrusions into the information systems, is that the security vulnerabilities existed in software, services, smartphones, communication system and operating system.

Vulnerability is one of the most weaknesses that can be leveraged by the attacker to break the information assurance of the corporation's system. According to the ISACA (Information Systems Audit and Control Association), security vulnerability can be defined as the logic fault in the algorithm while developing OS, software and service which can be exposed and used directly by a hacker to gain unauthorized access to company's network and system. By scanning and exploiting security vulnerability, attackers can gain the initial foothold to the system, and do many post-exploitation steps such as access unauthorized data, privilege escalation, generate backdoor shell with the administrator permission, cause the DDOS (Distributed Denial-Of-Service), or simply encrypt the data by ransomware virus to extort the company. Vulnerabilities are always existed because of the careless behavior of developers in security when designs and develops operating system and software. Even the misconfiguration of the services by system and network administrator can also make the vulnerabilities exist and available to be leveraged by attackers.

The good example which can demonstrate the importance in eliminating the vulnerability is the amounts of Microsoft vulnerabilities that being discovered over the last five years. In 2020, 1268 Microsoft vulnerabilities were discovered by security experts which is a very high number compared to the previous year (48% increase year-over-year), and the total number of Microsoft's product vulnerabilities has skyrocketed with a huge 180% increase since 2016. As a result of the increase in volume of vulnerabilities, attackers are now having more and more attack surfaces to exploit and gain access to the system. As we have known that "Elevation of Privilege" is indisputably one of the main vulnerabilities which has a serious impact to the system's security, and for the first time, this vulnerability accounted for the largest proportion (44%) of the total Microsoft vulnerabilities, raised 282% compared to the "Elevation of Privilege" vulnerability amount in 2019. Last but not least, ransomware attack nowadays can leverage and target a dozen of security vulnerability due to the significant increase of them year-over-year, compared to the past that ransomware attack would have just leveraged and used only one vulnerability. **[1]**

The security vulnerability is a big aspect in the security field, by that reason, this literature review will focus on the security vulnerability in various types of technology and the way how to mitigate them. Not only the security models and measures to eliminate the vulnerabilities that existed in operating systems, network, software, but also other emerging technologies such as Cloud Computing and Virtualization which play an important role in the future of technology. This literature review will include the assessment of each solution to find out both the strong points and weaknesses it still had, together with my own opinion for the future improvement.

II. Current state of Art

This literature review will summarize the approaches and methods about solutions for vulnerability mitigation and elimination in various types of technology, specifically in enterprise operating systems, Network, Software, Cloud Computing and Virtualization. To have the insight and comprehensive view about them, I have collected many books, papers published by Institute of Electrical and Electronics Engineers (IEEE), Springer and ScienceDirect to discuss and sought out key limitations along with the way how we can improve and bring better solutions for each of them.

a. Literature Review (10 papers)

1. Detecting cyberattacks that took advantages of vulnerabilities fixed in MS17-010 by using Event Log (IEEE). In this paper **[2]**, the Professor Mariko Fujimoto and her team from The University of Tokyo proposed the method to detect attacks that leverage the vulnerabilities fixed in MS17-010 by analyzing the records in Window's built-in Event Logs. By reviewing many existing solutions and methods related to the way of solving the mitigation tasks for the MS17-010 Windows vulnerability such as network packet analysis, processes analysis, registry or DNS query, the author and her team pointed out that those methods require to install security product (EDR, IDS), or need to alter the existing network structure which sometimes may be difficult in Industrial Control System (ICS) and its environment because of its long-time life cycle and severe requirement for availability. Therefore, the author had researched and sought out the way to solve those problems by taking advantages of Event Logs to reduce impact of the implementation process. Their solutions not only focused on analyzing records and Event IDs in Windows's built-in Event Logs, but also followed by developing many versions of detection algorithms to adapt with different types of MS17-010 attacks depending on the Windows versions. Last but not least, the researchers from The University of Tokyo also provided a solution to handle the cases that attackers delete all the logs to hide their attack traces after compromising the system, by using the Elastic Track which will automatically generate the record's copy of Windows's Event Logs in real-time and alert administrators immediately if the attacks are detected.**[2]**

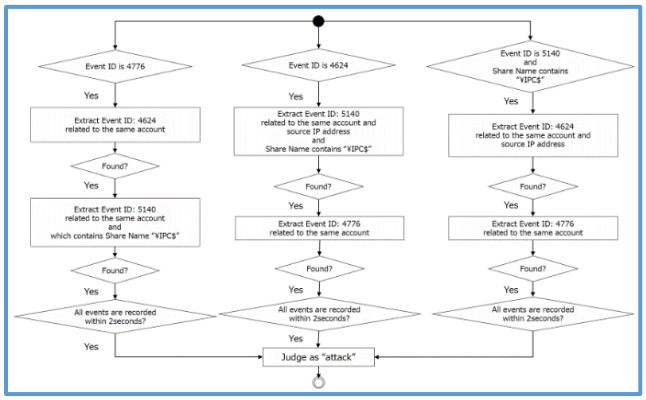

Fig. 1. Algorithm of the proposed method

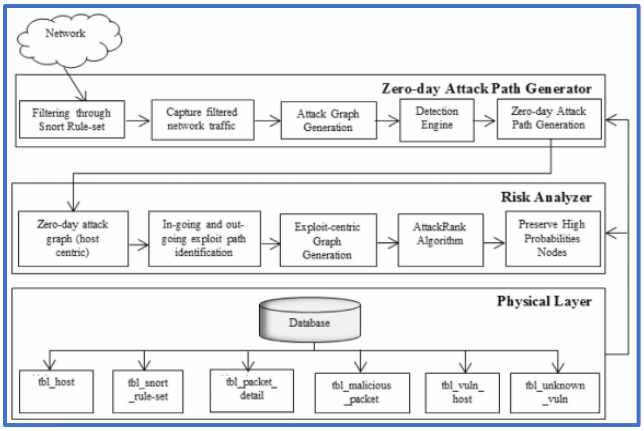

Fig. 2. Three Layers Architecture

2. A framework to deal with existing difficulties in zero-day vulnerability detection and prioritization (ScienceDirect). In this paper **[3]**, Dr. Umesh Kumar Singh with his team from Vikram University proposed a framework which focused on an integrated stragegy in detecting and priortizing zero-day attacks based on its likelihood. According to the paper, the 3-layer architecture which is preferred to use in the framework, is responsible for detecting and analyzing the zero-day attacks. This three layers architecture have assigned many specific functionalities to execute

and operate in parallel to raise the performance. The proposed framework was designed to block attackers by investigating and examining known samples of malware, and applied the probabilistic approach together with the hybrid detection-based technique in order to detect and identify the attack path of unknown vulnerabilities in network environment, as well to further rank the severity of newly discovered vulnerabilities**. [3]**

3. The robust intelligent detection measure for zero-day cyber-attack and vulnerability (Springer). In this paper **[4]**, Dr. Vikash Kumar and his team from National Institute of Technology Patna presented a novel approach to deal with unknown cyber-attacks and vulnerabilities by applying the concept of heavy-hitter and graph technique in this research paper. The proposed model was separated into two modules HVA and LVA, High Volume Attacks is responsible to derive high volume zero-day cyber-attacks using heavy-hitter, and Low Volume Attack has a mission to derive signatures for low volume zero-day cyber-attacks by making use of graph technique. The model was also developed to cover many types of zero-day attacks including variants of DDOS/DOS attacks, variants of data-theft attack, High Volume Attack (HVA), Low Volume Attack (LVA), scanning,… in which the paterns or signatures are unknown to the vendors. Moreover, two phases were consisted in this proposed work in order to assess the performance that are signature generation phase and evaluation phase. By that way, the signatures were generated at the training phase, then those signatures could be used for the process of evaluating performance by this model. **[4]**

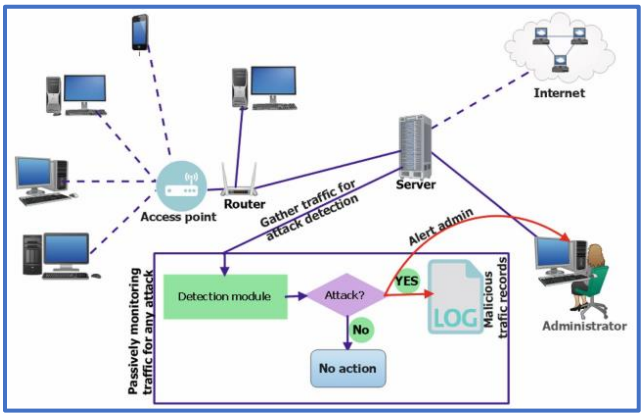

Fig. 3. Working procedure of proposed work

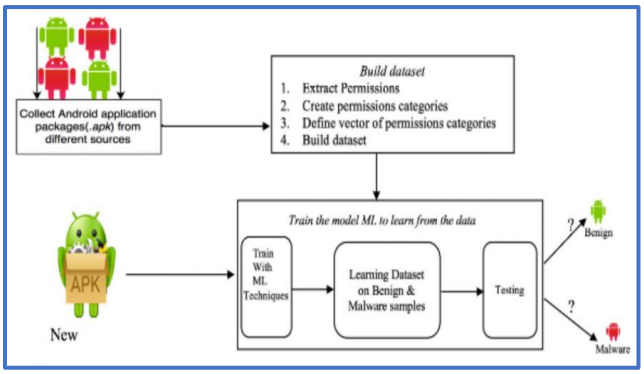

Fig. 4. Malware Detection Process with Machine Learning Techniques

4. A Machine Learning Model to detect vulnerabilities related to Permission Request for Android Applications (Springer). In this paper **[5]**, the author Didier Bassole and his researcher team proposed a novel model which used machine learning measures to enhance the process in detecting vulnerabilities automatically in the permission system of Android applications. This model not only allows users to expose the risk of vulnerabilities linked to an application through a combination of permissions or a permission, but also aims to perform assessments for vulnerability while enhancing the detection of those related to permissions with the goal to suggest mechanisms for preserving the data which is stored on the Android. Eight types of machine learning techniques have been applied into the framework to analyze the utilization of permissions in the detection of malware. Overall, this model focused on the analysis of potential risks relating to the utilization of these applications, and aimed to assess the vulnerabilities, conduct an informed evaluation of privacy risks and data security which Android applications incorporate, as well to raise the user's awareness throughout the practitioner and research communities. **[5]**

5. Applying the System Calls Analysis and Machine Learning in the process of detecting Android Security Vulnerabilities (IEEE). In this paper **[6]**, the author proposed an effective and robust approach in detecting vulnerabilities based on the detection of anomalies in system calls of both malicious and benign Android Application. Three categories were focused in the anomaly of their study that are sequence, frequency, and type of the system calls which may represent a vulnerability. The system would be configured to scan through the process of malicious and benign application, in order to detect the security vulnerabilities based on the combination of metrics as well as parameters. Moreover, sequence, frequency, and type of the system calls would be used to classify behavior of the process as malign or benign. The scoring threshold p and function f would be then defined and used by the detection algorithm with the purpose to detect the anomaly. The system improves the detection process by integrating many machine learning methods in order to identify a set of system call metrics and investigate the correlation between security issues and the pattern of system calls discovered. **[6]**

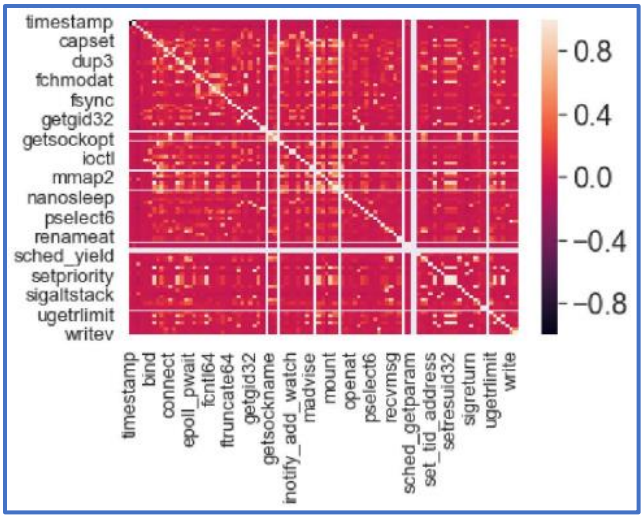

Fig. 5. System Calls Correlation Map

6. VulLocator – A System to detect vulnerable code automatically in Binary Programs (Springer).

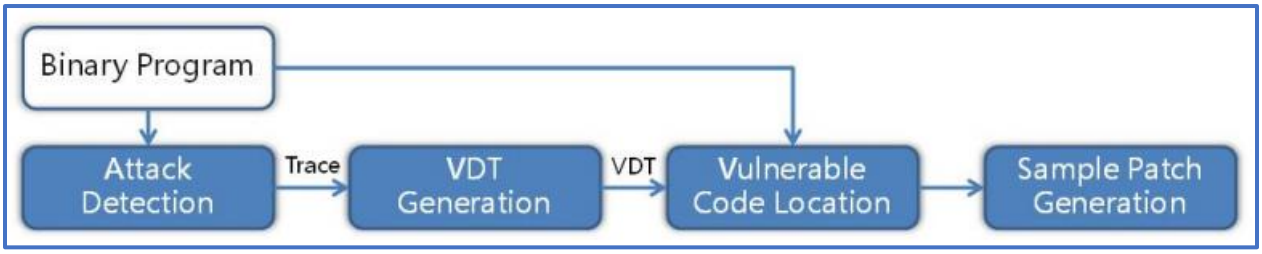

Fig. 6. The Process and Workflow of VulLocator

In this paper **[7]**, Professor Ying-Jun Zhang and his team from Chinese Academy of Sciences designed and developed a system based on a proof-of-concept prototype that can automatically detect vulnerable code in binary programs, which supported in the process of reducing the long delay in generating software patch. The system was presented with the name VulLocator that could analyze, locate vulnerable code and make evaluations from various types of vulnerabilities including memory corruption, format string, heap/integer/stack overflow, division by zero, and double free. It minimizes a great amount of instruction that need to be analyzed (from millions to dozens of instructions), by generating vulnerability dependency graph. The aim of the system VulLocator is to locate vulnerable code automatically in software without the need of any source code, which means no source code is required in the process of generating and analyzing dependence tree. By that way, many analysts and developers from the third-party could develop patches themselves. Moreover, VulLocator also has the ability to build a sample patch for short-term preventing against attacks. Although they may not function well as genuine patches, but they can temporarily prevent the attack through those vulnerabilities, and give more time for the analysts to create more fine-grained patches with information of vulnerable code given by VulLocator. **[7]**

7. VULCAN – A Framework to assess and evaluate vulnerability for Cloud Computing. In this paper **[8]**, the doctor Patrick Kamongi and his team presented VULCAN, a Vulnerability Assessment Framework to evaluate security vulnerability of Cloud Computing. They created a vulnerability taxonomy to classify security vulnerabilities as part of our effort to model them. Then, utilizing the data provided by NVD, they created an automated procedure to instantiate our ontology, which resulted in our ontology knowledge base – OKB. They may analyze and assess the security risk of individual or component portions of the cloud system's environment using this comprehensive OKB. They achieve this complete assessment through VULCAN components including System Classifiers and Indexer (SCI), Semantic Natural Language Process – SNLP, and so on... They envision that cloud computing security analysts, customer, and providers would be able to use VULCAN capabilities to examine their cloud environments in various ways. Furthermore, their framework is extensible, allowing developers to create and add new modules and components as needed. Users can also incorporate our VULCAN features into any other suitable desktop, cloud security assessment frameworks or mobile. **[8]**

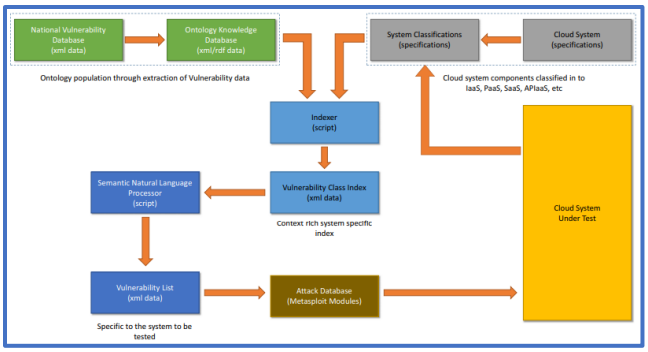

Fig. 7. VULCAN Architecture

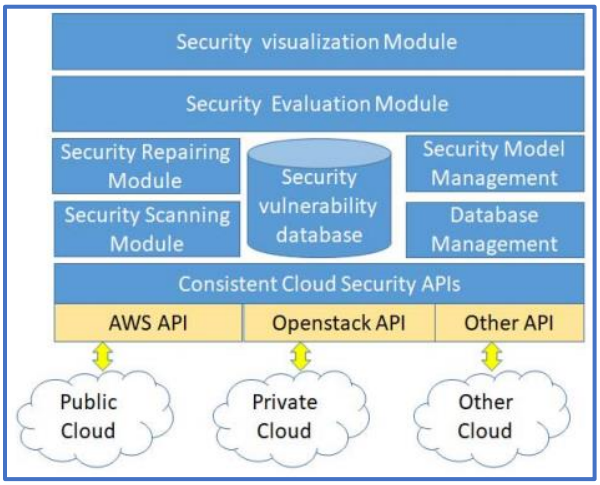

Fig. 8. Security Evaluation System for Cloud Computing Platform

8. The quantifiable security model to assess vulnerabilities in cloud computing platform (IEEE). In this paper **[9]**, the author Aobing Sun and his team from Chinese Academy of Sciences proposed a quantifiable security system to evaluate security vulnerabilities for single or multi-cloud environment which has the capability to be connected by using consistent API. This system included various types of engines and models such as security quantifiable evaluation model, security scanning engine, visual display module, security recovery engine, and etc. The security evaluation model is made up of a number of assessment elements that correspond to many different fields including storage, network, computing, application security, maintenance, and so on. Each element has a three-tuple of score, vulnerabilities, and repair technique assigned to it. The system applied a mechanism "One vote vetoed" for one field to add up the sum of its scores as the total score, and to generate a single security view. Through the security visual UI, the cloud users can obtain the security circumstance of their entire information infrastructure and architecture that was added to the resource library, then the security level can be rated by the score. Based on the G-Cloud platform, the authors can implement this quantifiable assessment system for different cloud users. It prepares and indicates the visual graphs along with the score of dynamic security scanning of one or multiple clouds, and guide the users to modify configuration, fix vulnerabilities, enhance the operation, as well to enhance the security of their cloud resources. **[9]**

9. Discovering specific vulnerabilities of virtualization in Cloud Computing Environment (IEEE). In this paper **[10]**, Guodong Zhu and his team member from University of Georgia researched and found out that the detection of these virtualization specific vulnerabilities generally needs domain-specific knowledge to be incorporated into the detection process. They had studied various types of known virtualization security vulnerabilities in different Virtual Machine Monitors (VMMs) which have been documented and published by National Vulnerability Database – NVD, Xen Security Advisory – XSA, and Common Vulnerabilities and Exposures – CVE. Their goal of this study is to analyze the differences and characteristics between vulnerabilities in traditional software and virtualization vulnerabilities, as well to categorize them into different groups of security vulnerabilities, in order to develop a systematic approach to expose them efficiently. The study conducted by them indicated that it is difficult to address the vulnerabilities using existing software validation and verification, due to the unique properties of cloud computing and virtualization. According to the observation of these documented vulnerabilities, the authors proposed three categories of vulnerabilities which are distinctive for the virtualization, and identified the challenges and proposed solutions of extending the traditional analysis measures of dynamic software – symbolic execution, in order to detect these vulnerabilities, as well to develop a framework to apply these detection methods to catch various kinds of vulnerabilities which are buried deep in the implementation of virtualization by the way of combining KLEE LLVM Execution Engine and Quick Emulator (QEMU) function test framework. **[10]**

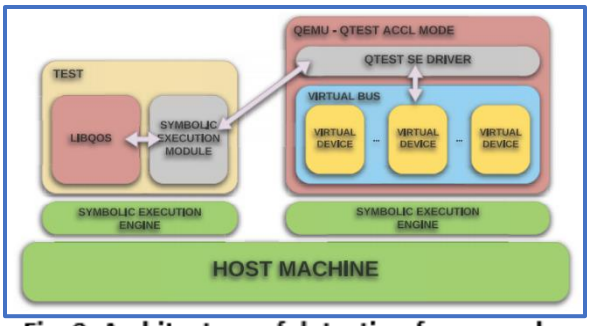

Fig. 9. Architecture of detection framework

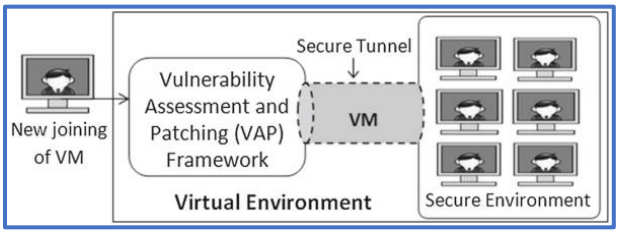

Fig. 10. Abstract model of proposed work

10. The effective and efficient framework for vulnerability assessment and patching (VAP) in cloud computing's virtual environment (Springer). In this paper **[11]**, the author Rejandra Patil and Chirag Modi researched and explored the feasibility in the process of implementing vulnerability assessment for a virtual machine by using proper analysis of vulnerabilities at the entry level with acceptable delay in provisioning Virtual Machine (VM). They proposed the VAP framework – Vulnerability Assessment and Patching which can automatically detect and prevent both network-level and system-level cyberattacks in virtual environment of cloud. The proposed framework will automatically perform the vulnerability scanning and detect open ports together with known vulnerabilities when new virtual machine (VM) is added to the virtual environment, in order to detect highly severe vulnerabilities, as well to perform the vulnerability evaluation by checking the exploitability and impact of those vulnerabilities. Therefore, they developed an algorithm integrated with the properties of scanned vulnerabilities to discover attacker sources. After functioning the framework, the severity of vulnerabilities would be calculated based on the impact, sources and exploitability of those vulnerabilities. If the patches of those vulnerabilities are available at the time when scanning finished, the framework would then patch them immediately, by contrast, VM risk analysis would be performed based on the unpatched vulnerabilities, and continuous monitoring will be given for those high-risk VMs. By that way, the newly joining VMs must go through the secure tunnel where framework would scan and patch for potential vulnerabilities of those VMs. After performing all the scanning and vulnerability patching for those VMs, the system will then add them into the running virtual environment. **[11]**

b. Methodology, strengths, key limitations/issues/challenges and own opinion

- **Vulnerability – Operating System (Windows)**

Nowadays, people have been still used the old versions of Windows, especially in Industrial Control System (ICS). In ICSs, it is common to see many up-to-date patches of Windows operating systems haven't been installed, and were deferred indefinitely. By that reason, a great amount of Windows vulnerability hasn't been patched until now, and it leads to one of the most severe case of vulnerability that is Remote Code Execution, which attackers can leverage to execute arbitrary code remotely on the target computer. MS17-010 is the Microsoft's security bulletin code which had been leveraged by the attackers to implement one of the worst cyberattacks in history – WannaCry ransomware cyberattack. The vulnerabilities fixed under that Microsoft's bulletin code have been still used and many attacks relating to that code have been conducted. By many reasons above, this paper will present a novel measure to counter the cyberattacks leveraging vulnerabilities fixed in MS17-010 by using Windows's Event Log.

b.1. Detecting cyberattacks that took advantages of vulnerabilities fixed in MS17-010 by using Event Log (IEEE)

According to author's research **[2]**, the attack measures and Event Logs recorded by attacks are varied based on the types and versions of the Windows Operating Systems. So that, they preferred to investigate the detection ways and countermeasures of the attacks for each version of the Windows to sought out several characteristics indicating the attacks. Thereby, the summary of the characteristics of attacks that leveraging the vulnerabilities fixed in MS17-010 found in the Event Logs will be conducted and listed. Moreover, by analyzing the Event Logs along with the Event IDs which is recorded in normal operations, they recognized that if one of the recorded Event IDs is used for detection alone, false positives are likely to happen. Following that way, they found that the security Event IDs are continuously recorded in a short time period, therefore, they decided to define the detection signature and develop the detection algorithm as the following steps. At first, the detection algorithm will filter the security Event IDs and export the Events in the CSV format, after that, it will put together the Events that being recorded in a certain time period. The final step is to check if each group included Events which matched with all detection signatures and characteristics, if its matched, the algorithm will judge these Events as the attacks and the alert will be activated, following with the incident response by administrators and blue team. Furthermore, because of the Windows OS specifications, it hasn't had any information that showing the relationship among Events which is used for the signature, hence, the researchers from University of Tokyo decided to put relevant Events based on the average time delay between event recorded of several attacks. By investigating them in an unbiased way that the attacks were carried out by several people in various contexts, the average would be taken, and the accuracy together with the objectivity of the result would be increased. Last but not least, the author also provided the way to counter the attackers, in case that they compromise the computer and delete all the event logs to hide their attack traces, by implementing the system using Elastic Stack which is used with the purposes to automatically store and preserve the logs from Windows's built-in Event Logs. Moreover, it can also be used to detect the attacks in real-time and alert immediately to the administrator for the quick incident response. **[2]**

Not only that, this method was design and developed to fix the difficulties of traditional techniques which also implemented to counter vulnerabilities fixed in MS17-010. In addition to the mentioned strengths above, this method can be easily integrated into a production environment such as Industrial Control System (ICS) because of its initial approach that can analyze Event Logs in offline. Moreover, as you can see from the papers **[12] [13]**, SolarWinds introduced a detection method for WannaCry by using various

features including DNS queries and registry, M.Satheesh Kumar and his team propose the detection rules of IDS for Eternal Blue by analyzing packets in network, while Da-Yu Kao presented a detection method for EternalBlue by analyzing packets and processes. The dificulty of those methods is that it only focused on exposing WannaCry malware which was delivered by using vulnerabilities fixed in MS17-010, and it didn't focus on other cyberattacks which also leverage vulnerabilties fixed in MS17-010. Another difficulty is that when we implement those above methods, we need to consider severe requirement for availability and long-time life cycle because it can be affected while we modify existing network structures or install security products such as IDS, Endpont Detection and Response. Because of those reasons, the technique proposed in this paper using Windows's built-in Event Logs, recorded all the Event IDs to analyze and process in algorithm which wouldn't affect the requirements of system.

Despite all of the strong points which I have discussed above, it still existed many weaknesses that need to be figured out and improved. This method still requires to install specific version for each type of Windows Operating Systems, and it can bring many limitations for the administrator to install this method to the whole network of their company. Adminstrators always need to have all versions of this technique on hand, and they need to install and implement this solution for each computer separately which required a great amount of time to deal with. Moreover, this method should enhance its feature to deal with other types of Microsoft's security code, to counter and deal with other security problem of windows.

From my opinion, this method should be upgraded and intergrated with the technique to recognize the type of Event ID in Event Log based on the Windows versions. By that way, this method wouldn't be seperated in many versions, and we can develop a system to cover all the versions and it can automatically recognize and select the version of method based on the version of Windows. Moreover, this technique should also integrate with the National Vulnerability Database (NVD) and Common Vulnerability and Exposures (CVE) to cover other types of vulnerability cyberattacks in Windows, not only focus on the vulnerabilities fixed in MS17-010. Last but not least, this method should add more attack tools using not only Event Logs but also network packet to detect the vulnerabilities and attacks, the requirement of long-term life cyle and severe requirement for availability should be remained the same as well as focus on when deal with the future work and improments.

- **Zero-day Vulnerability – Network & Infrastructure**

Nowadays, even the enterprises have focused on the security of their network, employed a great number of cyber security specialists to deal with the cyberattacks happened to the company, applied many techniques to check for the known vulnerabilities of the system, the cyberattacks still happened and caused heavy losses to the enterprises. Zero-day vulnerability has always been a difficult task for security engineers and specialists since it covered unknown vulnerabilities and haven't been published and patched. By that reason, the next 2 papers developed a framework and technique to deal with this special type of vulnerability by different approaches.

b.2. A framework to deal with existing difficulties in zero-day vulnerability detection and prioritization (ScienceDirect)

This framework **[3]** will cover a 3-layer architecture and this architecture would be used to detect and analyze the zero-day cyberattacks. The first layer is the generator of Zero-day attack path which has the mission to detect the vulnerabiltiies that haven't been discovered and published yet, the second layer is the Risk Analyzer that is assigned with the purpose to analyze the generated cyber attacks. And the final one in this architecture is the physical layer which comprises both the centralized server and database that are utilized with the goal to process the information of first two layers. **[3]**

The framework **[3]** was separated into 2 phases, the first one worked by leveraging favorable attack conditions obtained from various information sources which had been noticed and written up by IDS and security specialists. By that way, favorable attack conditions and known malicious packets would be collected in order to detect and filter at the initial stage. Furthermore, the framework would be used to examine the abnormal activities of our network and system, as a result, the attack graph could be built from the way of capturing the network scenario at any time stamp. This task also has the mission to avoid unnecessary payloads and handle heavy network traffic. Throughout the second phase, the higher risks of the zero-day vulnerabilities along with the nodes of produced attack graph were discovered. This is done by leveraging an intelligent algorithm which is called AttackRank that evaluating and ranking the severity of those identified zero-day vulnerabilities. AttackRank algorithm which was integrated inside the proposed framework, would measure the risks of unknown vulnerabilities and support the process of designing and preparing remediation plans. Finally, by using various standard parameters, the authors had prepared for the experimentation of their layered architecture and framework to verify the efficiency, and observed that the false positive rate was 0.3 percents along with the truest detection rate was about 96% which proved for the reliability of this approach. **[3]**

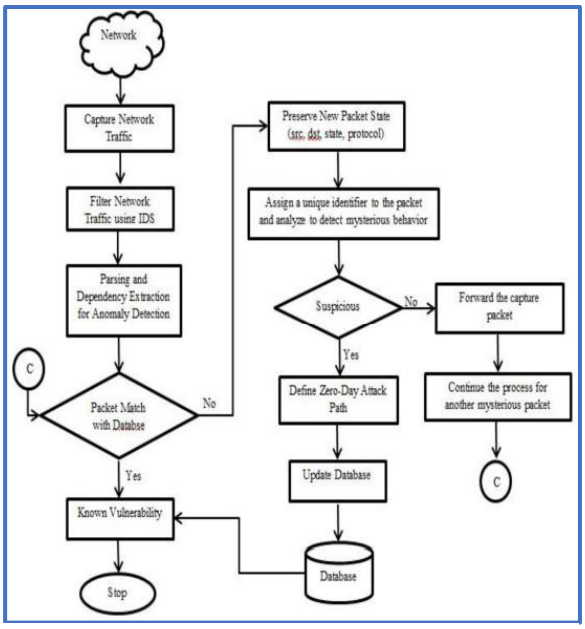

Fig. 11. zero-day attack path generator layer of proposed framework

Not only the strengths of this framework that I have discussed above, the core approach of this method is to solve the difficulty that traditional solution for zero-day vulnerability attack still had. The approach is to measure the risk level of vulnerabilities by applying the Hazard metric with the involvement of impact as well as frequency factors. Because we faced the issue here that the level of zero-day attacks risks is immeasurable, and we cannot calculate the severity of vulnerability while it is unknown, by that reason, this method considers the degree of exploitability to solve this challenge. Despite all of the strong points that I have mentioned and explained throughout this **b.2.** section, this framework still had many key limitations and challenges that should be improved in the future work. This framework can only analyze and detect the zero-day vulnerabilities in the network aspect and infrastructure, when the attacks lead to the unknown vulnerabilities inside the operating system or the services and software installed inside that OS, it's hard to prevent and take incident response immediately and the impact would be bigger based on the time that the cyberattacks haven't been recognized and detected. Sometimes, we shouldn't delete the zero-day vulnerabilities that we have discovered by using this framework, instead, we can create the honeypot from those unknown vulnerability, and when the attacker detects that unknown vulnerability and leverage it, we can use that honeypot to expose the source of attacker.

In my own opinion, the feature of this framework wouldn't be limited, and we should develop more tools inside this framework to not only analyze and detect in network and infrastructure aspect, but also extend the scope by applying the core of this zero-day vulnerability detection framework. We should find out other way to leverage the zero-day vulnerability to protect the system and catch the attacker, not only

focus on the way how to remove all of the unknown vulnerability inside the whole system. To reduce the time to apply this framework to other network and system, the database of this framework should be designed to globally accessed through a private line, and must be encrypted. When we implement this framework in other network, it will evaluate and save all of the discovered zero-day vulnerabilities into the online general zero-day vulnerability database, the frameworks which have been implemented will interact with each other through the database to reduce the time for exposing those vulnerabilities. This zero-day vulnerability database would be used only by this framework and must not be published to prevent case that those unknown vulnerabilities are leveraged by attacker to attack other system throughout the world.

b.3. The robust intelligent detection measure for zero-day cyber-attack and vulnerability (Springer)

Two phases were consisted in this proposed work **[4]** in order to assess the performance that are signature generation phase and evaluation phase. By that way, the signatures were generated at the training phase, then those signatures could be used for the process of evaluating performance by this model. After completing the signature generation phase, next stage is the attack detection phases. At this stage, a knowledge base consisting of attack signature will be generated by the system. These signatures represent the smallest unit, and they could be a key aspect in determining whether or not a flow is malicious. Attack signature knowledge bases, a data collection and preprocessing unit, and an attack detection unit make up this phase. The detection module accepts real-time traffic and converts it into the required format for detecting zero-day attack traces using knowledge bases. Even if only one signature fits the flow, the system sends an alert to security specialists for additional investigation and temporarily halts the flow. If the attack is confirmed by the expert, the model uses the traffic to find new signatures. For the working produre of this technique, the detection module will be placed on the server at first. This module receives a copy of every communication flow. The communication is preprocessed before being supplied into the detection module, which is in charge of detecting malicious traffic. The records relating to the detection of harmful actions are kept track of. In addition, an alert is sent to the administrator to confirm it. **[4]**

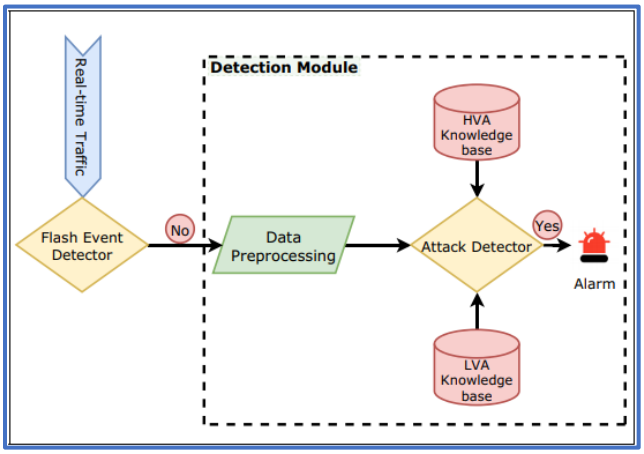

Fig. 12. Detecting ZA by applying signature knowledge base

By setting up a virtual environment which comprised 10 HVA nodes, 10 genuine and 3 LVA nodes, the data which had been used in this paper can be captured, as a result, the research indicated that this approach worked well on the format of raw hexadecimal byte, and can detect the zero-day cyber-attacks successfully. The suggested zero-day attack detection has a greater accuracy of 91.33 percent for binary classification and 90.35 percent for multi-class classification on real-time attack data, according to the results analysis. For binary-class classification in this model, the performance against benchmark data set CICIDS18 indicates a result of 91.62 percent which is quite reliable.

Not only the strengths of this model that I have discussed above, the core approach of this model is to solve the difficulty that traditional solution for zero-day vulnerability attack still exist. In papers **[14] [15],** we can see that various proposed works are restricted to the High Volume ZAs (HVA), because of that reason, the model of this paper was developed with the goals to solve those restrictions, and included the function to not only detect the Low Volume ZAs, but also detect the High Volume ZAs. Furthermore, other challenges still existed in the traditional solutions such as independence of source and destination specific information (IP Address, Ports, and so on…) have been solved from the approach of this model, and can be configured to cover broader variants of zero-day attacks.

On the other hand, this model still had several limitations that need to enhance and improve in the future works. The robustness of this approach is still low when applying to detect zero-day attacks, moreover, the exact category of Low Volume ZAs and High Volume ZAs variants cannot be detected because of the approach of its implementation.

In my opinion, the robustness of this model can be improved and enhance by the way of detecting several types of zero-day attacks whose behaviors are independent of traditional and existing attacks. Tests should be performed regularly to enhance the accuracy for the case of multi-class classification. We need to scan for intrusive patterns and reduce the temporal complexity of Low Volume Attacks signature generation.

- **Vulnerability – Software**

Presently, more and more software engineer pay attention to the vulnerability in logic of algorithm and code which they have been developed to create a software for windows, android, or even linux. The vulnerability existed in software can lead to many dangerous case that attacker can leverage and exploit it to gain the foothold to the system, and prepare for other post-exploitation step. For example, when android application is installed in your smartphone, you need to read the permission and give those permission for your android application to be installed into your phone, at that step, you gave several permissions for the software such as permission to read your messages, or permission to upload and share your data, and so on… In case that this software had known or unknown vulnerabilities, the software has capability to be exploited and attacker can use the software to do malicious action to your android phone. By those reasons, it is important to detect vulnerabilities in Software and remove them as soon as possible, before attacker can discover and leverage it to do malicious actions.

b.4. A Machine Learning Model to detect vulnerabilities related to Permission Request for Android Applications (Springer)

The methodology of this model **[5]** is to examine the relationship between the vulnerabilities existed in Android applications and the authorization request, for the purpose to detect the malware by applying the machine learning techniques. Their approach of this proposed model is to develop the criteria of risks based on the impact of Android authorizations in the process of accessing data. In this approach, the author used a set of harmful and benign data to derive the characteristics for malware detection. After that, the security rules can then be established based on the request templates of the risky permissions in the dataset. The machine learning algorithm of this model is developed with the goal to learn from the data. LR, MLP, KNN, LDA, RForest, DTree-Cart, NB and SVM are all the classification basis used in this Machine Learning, which are applied to predict the type of vulnerability. There are 2 cases of the class would be specified when the prediction completed, that are ‘Malware’ and ‘Benign’. ‘Malware’ class in this machine learning algorithm indicated that it is the vulnerability with high severity and impact. On the

other hand, the 'Benign' class indicated it is the vulnerability with low severity and impact. Last but not least, according to the experimentation, Random Forest Algorithm shows better score of precision, recall and F1-score compared to each other: 0.98 for malware – precision, 1.00 for malware – recall, 0.99 for malware – F1-score, 0.94 for benign apps – precision, 0.81 for benign apps – recall and 0.87 for benign apps – F1-score. **[5]**

The strong point of this technique is that it can use the machine learning to calculate the severity of the vulnerability, and predict the type of vulnerability based on the characteristics, permission and type which have been categorized in the database, in order to see what type of vulnerability has the highest correlation with this exposed vulnerability. The system will check the type of vulnerability which has the highest correlation score, and consider it as the type of the exposed vulnerability. Compared to the traditional prediction model to recognize the type and calculate the severity of vulnerability, this model applied 8 types of prediction models to evaluate the vulnerability, while the traditional prediction model only used 2 or 3 detection models. By applying 8 types of models, the system wouldn't just rely on one prediction models to recognize and calculate the correlation and severity score, the system can also compare those scores of each model to make the process of recognizing and calculating more accurate.

Although it had many strengths when apply this model into the process of detecting vulnerability in software, but it still had many restrictions that should be improved in the future works. After detecting the type and calculating the severity of vulnerability, the models specify only 2 cases for vulnerability that are "Malware" or "Benign", the models should categorize them into various dangerous level.

In my opinion, the model should develop a system that calculate the severity score from 0 to 10, while 0 means that this vulnerability has lowest severity and impact to the system, and 10 means that this vulnerability has highest severity and impact, in order to categorize the severity of vulnerability more clearly for user. Moreover, the model should also check and categorize the purpose of each application into the group, and calculate the severity of permission that have been given to each application, as well to calculate the dependency and relevancy between permission and purpose of application, in order to alert the user if weird permission has been given to android application.

b.5. Applying the Machine Learning and System Calls Analysis in the process of detecting Android Security Vulnerabilities (IEEE)

In a controlled environment **[6]**, the author ran the Android Application with known vulnerabilities in order to collect the system calls caused by the applications, afterwards, they designed the sequence of system calls as the anomalies in their research to generate the baseline for this system. The base of their analysis based on the time-series of system calls log data. They consolidated the information by focusing solely on the system calls, and removing any irrelevant or unnecessary parameters along with other parts of the information. Their next step is to integrate the machine learning technique that need to be used in the process of detecting anomalies and measuring the performance of detection system. Three different supervised machine learning algorithms were applied to compute the recall, F-Score and precision, as well to compare the results. The F-Score is considered as the most relevant based on its capability to consider both the false negative and false positive. However, through the research and experimentation process, the author concluded that an algorithm which shows up the balanced performance in three metrics, should be considered to be the excellent performance. After evaluating each algorithm for anomaly detection, they discovered that the Genetic Algorithm – LSTM (GA-LSTM) was able to find a result that was slightly better than the Long short-term memory (LSTM), but the K–Nearest Neighbors (KNN) was

able to offer a balanced performance in all three metrics, which can be considered as excellent performance. According to the results, machine learning approaches can predict flaws and vulnerabilities with an F-score of 85%, and it is feasible to detect and mine security vulnerabilities using various machine learning techniques based on the evaluation of KNN, LSTM, and GA-LSTM. **[6]**

Despite of using permission request in the paper **[5]** to detect the type and calculate the severity of android application vulnerability, the paper **[6]** paid attention to apply machine learning with system call to do that. Compared to the paper **[5]**, this method hasn't had any strong points since it only used 3 of the machine learning technique to forecast and detect bugs and vulnerability. The percentages of precision, recall, and F1 score experimented by this paper is completely lower compared to those scored made by Random Forest algorithm in the paper **[5].** While the authors in paper **[6]** just focused on proving the possibility to detect security vulnerability by using different machine learning models, the authors in paper **[4]** not only proved that theory, but also used various machine learning models to sought out what type of ML model offers the most robustness in detecting security vulnerability for android application.

In my opinion, the prediction system published in paper **[6]** should include more vulnerability data to train and raise the capability and probability of the models in the process of detecting different types of vulnerability. Moreover, the author should conduct the experimentation with the increased complexity to verify if the results taken from F-Score remains the same level, in order to check the quality and enhance the prediction system.

b.6. VulLocator – System to detect vulnerable code automatically in Binary Programs (Springer)

There are two steps in the attack detection phase **[7]**. The program begins by providing attack input data into it. VulLocator records the execution trace as the program runs. The attacks are then detected by VulLocator utilizing taint analysis or searching for program crashes. Exploited instructions referred to the last instruction in the trace. After an attack has been detected, the slicing approach is used to locate the vulnerability's relevant instructions in the execution trace. The vulnerability dependency tree is then generated by VulLocator. We add edge lengths to the tree to demonstrate the relation between exploited instruction and instructions, which is different from typical slicing methods. VulLocator uses edge lengths to locate susceptible code based on VDT. As a result, not all of the instructions in trace need to be examined. In practice, the first and second phases are integrated. VulLocator stops constructing VDT and tries to build a sample patch when it suspects the code is vulnerable. The final step is to create a sample patch. The patch acts as a stopgap measure in the event of an assault. It might also be utilized as an idea for analysts to come up with a real vulnerability. **[7]**

In experimentation **[7]**, VulLocator finds vulnerable code in a variety of programs and these programs have vulnerabilities including memory corruption, integer/stack overflow/heap, division by zero, double free, and format string. To assess the efficiency of our technique, we use input of the exploitation from Exploit-DB. The table below shows the evaluation results on effectiveness of VulLocator**. [7]**

Program	Vul Type	T.Len(A)	T.Len(S)	VDT.S	Pv	C.E	C.N
Dizzy 1.12	Stack Overflow	25360285	11831550	13	√	N	N
ZipItFast 3	Heap Overflow	547164959	210754229	10	√	N	N
MS Paint 5.1	Integer Overflow	66682422	30828159	31	√	N	N
FXSCOVER 5.2	Double Free	162868592	80391484	22	√	N	N
VLC 1.1	Memory Corrupt	214299644	94774038	6	√	Y	N
RadASM 2.2.1.5	Format String	379353073	151896256	19	√	N	N
Zortam MP3 1.5	Division by Zero	1685231785	1205914686	8	√	N	N

T.Len(A): Trace Length (All); T.Len(S): Trace Length (Memory/branch operation); VDT.S: VDT Size; Pv: Prevented; C.E: Crash (abnormal input); C.N: Crash (normal input)

Although VulLocator brought many benefits for the user, but it still existed some restrictions that should be focused on in the future work. Since this system follow the 4 conditions to detect vulnerability, these 4 conditions can detect most of those, especially for the arbitrary code execution vulnerabilities. But there are still some vulnerability types that cannot be discovered. To answer for this problem, I think the system should use more complex techniques such as Instruction Set Randomization **[16]**, Taint Analysis **[17]**, and Address Space Layout Randomization (ASLR) **[18]** in future work. Moreover, false positive and false negative have a chance to happened in the detection phase that we need to excluded the handler monitoring conditions to detect attacks. In this case, the system could be developed with more sophisticated detectors to deal with this problem. Last but not least, the temporary patch made by VulLocator may not work properly compared to the real patches, by that reason, the analysis needs to be delivered to the security analysists to generate the real patch. In my opinion, the patch generation process of this system VulLocator should be paid more attention and do more experimentation in future work.

- **Vulnerability – Cloud Computing**

Nowadays, assessing the security vulnerability in cloud computing is become one of the most concerns because more and more people are interested in using it and applying it into their on-premises system. The benefit of reducing the budget to maintain company system by moving to Cloud is very great, and it's gradually becoming a global trend. Despite of those benefits that Cloud Computing provides, many attackers are now doing research and exploiting the Cloud Computing functions to gain the unauthorized access to the sensitive data that have been stored in the Cloud. By that reason, the next 2 papers **[8] [9]** proposed framework and model to detect and evaluate the vulnerability for Cloud Computing.

b.7. VULCAN – A Framework to assess and evaluate vulnerability for Cloud Computing

Utilizing VULCAN modules [8] to evaluate vulnerabilities for android smartphone by applying Mercury Framework is a common use case scenario, and it would go like the following stage. A User sends both dynamic inputs (for example, "Android") and a natural language query (for example, "Assess for flaws that could allow unwanted access to my device?") to VULCAN framework's System Classifiers module. VULCAN Semantic Natural Language Processor (VulcanSNLP) will process this question. After that, The System Classifiers module develops potential android-based solutions and passes them on to the Indexer module. The Indexer then builds suitable vulnerabilities indexes, which are then used by the Vulnerability Class Index module to generate vulnerabilities groups. "Root Access" is a sample created vulnerabilities group that contains indexed data of these CVE-IDs "CVE-2009-2692", "CVE-2011-3874", and "CVE-2011-1823". The next stage is the SNLP component which has the mission to perform reasoning activities on the user query and use the vulnerabilities group data that has been produced. It will deliver appropriate results to the user through a conversation agent interface, such as IT Products with vulnerabilities and other essential information that corresponds to the user's inquiry. They map the discovered IT Products with a Mercury framework module named "Test for vulnerabilities that allow a malicious application to get root access" to implement the cyberattacks on the products inside our targeted android user device by leveraging our Middle-ware application. Finally, VULCAN tracks the module payloads' distribution and reports on whether or not the assaults were effective on the device, as well as whether or not the tested vulnerabilities are still present or addressed for that IT Products**. [8]**

In my own opinion, this VULCAN system should extend its features in future work. The first target is to incorporate metrics into our system that will allow users to compare infrastructure, vendor as well as products based on the absence and presence of published vulnerabilities. The second purpose is to expand

the information in the OKB so that users can see correlations between vulnerabilities. Moreover, we should investigate the possibility of deploying our SNLP component for mobile devices as the application that allows users to analyze vulnerabilities on the fly.

b.8. The quantifiable security model to assess vulnerabilities in cloud computing platform (IEEE).

The G-Cloud Platform **[9]** served as the foundation for the development of our quantitative security evaluation system. There are two independent Clouds on our experimental platform. One cloud was built using G-CLOUD OS, and the other was built using OpenStack. With the same security API, the two clouds can be monitored and interacted with. We ran two groups of trials to see how well our evaluation system performed on a single cloud platform versus a dual cloud platform. The trials were carried out with a compute resource pool of 3000 cores, a storage resource pool of 200TB, and a network resource pool of 10GB/S. There are two independent Clouds on our experimental platform. One cloud is built on G-CLOUD OS, while the other is built on OpenStack. Cloud 1 has 16 physical machines (totaling 512 cores) and 200TB of storage resources. Cloud 2 has 10 physical computers with a total of 320 cores as well as 150TB of storage. The two clouds share a 40Gb/s network bandwidth. The same security API can be used to monitor both clouds. To see how well our evaluation method worked, we ran two groups of studies. At first, the experiment was run on cloud 1 to replicate a single private cloud. The current state of the system, the resource scale, the security score, and the security trend. The worldwide security scanning can be cyclically performed on a single cloud platform. Similarly, the security state of two clouds may be scanned and displayed within a single user interface, because the scanning module can access both clouds using the same owing privileges as well as API. Only the securing status of the user's resources that were granted access will be checked by scanning module. The security view can then be used to display its status**. [9]**

The security assessment can be carried either in serial or parallel mode as you can see from figure below.

ID	Scanning Mode	Serial Scanning Time(s)	Parallel Scanning Time(s)
1	Global Scanning for Single Cloud	3607	806
2	Global Scanning for 2 Cloud	5700	1211
3	User Scanning in Sing Cloud	121	30
4	User Scanning in 2 Cloud	260	50

TIME COST OF SECURITY SCANNING WITH SERIAL OR PARALLEL MODE

When using serial scanning mode **[9],** the security scanning module is installed on a single server, and each computing resource – both virtual and physical, storage resource, and network resource is scanned individually. The resource scope was sometimes very large, and the global scanning duration was quite long, up to several hours. Because of that reason, we can utilize parallel mode with the purpose to implement scanning module on lots of servers (one cluster together with 5 virtual machines), partition each cloud into separate resource groups, allocate scanning jobs to other servers to speed up scanning and reduce the time it takes to get user approval. Similarly, security scanning for one user's cloud resources in a cloud or multiple clouds can be done in parallel or serial. In most cases, a user's security perspective can be obtained in under 30 seconds, and time can be accepted by users of real cloud platform**.**

It still existed many restrictions for this system to be implemented in other platform to extend the scope and the capability of applying this system widely. With consistent API and access privileges of cloud OS, the existing quantified security evaluation system can only be deployed on a single cloud or multi-cloud

environment. In my own opinion, we should create similar security evaluation tools and integrate resource and security monitoring into a single security management mode and user interface, in order to solve the key limitations that I have mentioned above.

- **Vulnerability – Virtualization**

The demand for greater virtualization security is growing in conjunction with the rapid rise of virtualization-based computing technologies. Since many vulnerabilities have been exposed in the virtualization technology and the cyberattacks which leveraged those vulnerabilities happened, increased day-by-day due to increment of cloud computing features, many cases that sensitive data of the company hosted in Cloud was leaked and published in the internet. It is critical for cloud Infrastructure as Service (IaaS) providers to provide users with safe and vulnerability-free resources in order to improve service quality. The next 2 paper **[10] [11]** presented method and framework to detect and assess the security vulnerability existed in the Virtualization.

b.9. Discovering specific vulnerabilities of virtualization in Cloud Computing Environment (IEEE)

In paper **[10]**, the framework is made up of two primary piece: an altered version of QEMU that executes in LIBQOS and QTEST ACCL Mode, which handles the virtual hardware test cases. These two components communicate with each other via sockets and work in a client-server model. When executing in QTEST ACCL Mode, QEMU can only initialize the device under test as well as a few other critical components required for the device to function properly. LIBQOS feeds an API to manage the execution of QEMU in a test environment by providing fundamental functions like IRQ and clock cycle. The symbolic execution engine that executes inside the Operating System host, directs the test execution. Symbolic variables can be defined in a unit test using a symbolic execution module, then supplied to QEMU via LIBQOS, where they are translated and assigned to device states using the QTEST SE Driver. During the test, the symbolic execution engine will record the execution trace and provide test output that represents virtual hardware's execution pathways. **[10]**

Next is the modified QTEST Framework, by introducing a symbolic execution module to QTEST, we expanded its capabilities of testing the functionalities of each virtual hardware to include the discovery of virtualization-specific vulnerabilities in the implementation of specific virtual hardware. Because QTEST is part of the QEMU codebase, virtual device developers can use it to perform not just functional testing but also security testing, which helps to remove implementation errors that can direct to virtualization vulnerabilities. When the device implementation becomes complicated, dealing with this manually will become quite tough. We combined our earlier work extracting critical device states for live virtual machine migration in cloud computing as a prototype and to check for vulnerabilities of device state in the implementation of virtual hardware.

In my opinion, because this work [10] is still an ongoing research from the author, even it can detect the virtualization specific vulnerability but it hasn’t been done and worked properly in the experimentation yet. By that reason, the future work needs to test the implementation of virtual hardware, as well to check for the device state vulnerabilities. Moreover, the project should include the resource availability vulnerability detection as well as the conformity inspection of virtual hardware on the framework.

b.10. The effective and efficient framework for vulnerability assessment and patching (VAP) in cloud computing's virtual environment (Springer)

The presented framework **[11]** exposes the security vulnerability existed in the new joining of the virtual machine. On each cloud server, we ran a distinct set of tests. We used three distinct cloud servers to run three different test scenarios. The suggested framework takes into account the joining of virtual machines on cloud servers. The specifics of freshly joined VMs for each test scenario. A thread of the proposed framework is assigned to each newly joined VM, which conducts vulnerability scanning, evaluation and patching. External modules such as Nmap, Java multi-threading, LIBVIRT library, and Nexpose API are used to scan for vulnerabilities. LIBVIRT gives program access to hypervisor-based virtual machines. VMD stores the details of newly joined VMs. The Java multi-threading API is used to link numerous virtual machines. Each thread looks for potential vulnerabilities and open ports in the virtual machine. To scan the VMs, we use the Nexpose Ruby API. Java is used to implement port scanning with Nmap. Secure Shell (SSH) is used to implement vulnerability patching, and the presented framework employs the Java secure channel API to execute SSH commands. It uses SSH instructions to connect to the virtual machine, then transits the patch file from the host machine to the virtual machine, and lastly runs the patch file on the virtual machine. **[11]**

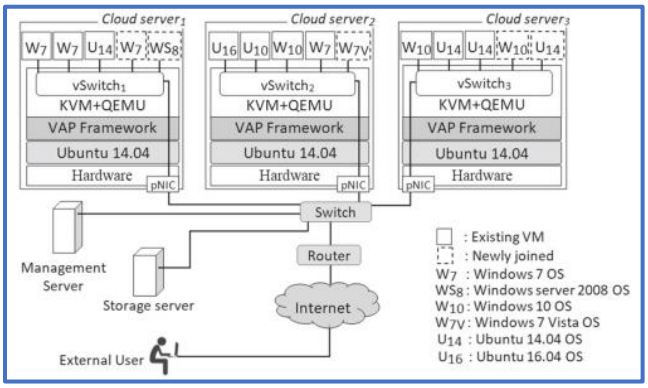

Fig. 13. Experimental Setup

Results and Analysis

No.	VM name	Number of vulnerabilities	Scan date and time	Scan duration (s)
Case-1	Windows-7	28	July 19, 2017, 16:46:06	123
	Windows-server-2008	27	July 19, 2017, 16:49:02	120
Case-2	Windows-7-Vista	29	July 19, 2017, 17:04:22	180
Case-3	Windows-10	11	July 19, 2017, 17:13:44	162
	Ubuntu 14.04	2	July 19, 2017, 17:16:09	112

Fig. 14. Number of vulnerabilities detected for newly joined VMs

No.	Virtual machine	Open ports	Protocol	Scan date and time	Scan duration (s)
Case-1	Windows-7 (192.168.9.56)	22, 135, 139, 445, 49152, 49153, 49154, 49155, 49156, 49157, 49158	tcp	July 19, 2017, 16:44:20	17
		137,5355	udp		
	Windows-server-2008 (192.168.9.57)	22, 135, 139, 445, 49152, 49153, 49154, 49155, 49156, 49157, 49158	tcp	July 19, 2017, 16:47:19	17
		137,5355	udp		
Case-2	Windows-7-Vista (192.168.9.70)	22, 135, 139, 445, 49152, 49153, 49154, 49155, 49156, 49157, 49158	tcp	July 19, 2017, 17:01:39	17
		137,5355	udp		
Case-3	Windows-10 (192.168.9.81)	22, 135, 139, 445, 49152, 49153, 49154, 49155, 49156, 49157, 49158	tcp	July 19, 2017, 17:11:19	17
		137,5355	udp		
	Ubuntu 14.04 (192.168.9.83)	47760	tcp	July 19, 2017, 17:14:20	3

Fig. 15. Port scan details for each VM

No.	Virtual machine	Vulnerability ID	*AV*	*AC*	*Au*	*CI*	*II*	*AI*	*ECM*	*RL*
Case-1	Windows-7 (192.168.10.56)	CVE-1999-0524	*Lc*	*L*	*N*	*N*	*N*	*N*	*X*	*X*
		CVE-2011-0657	*Nw*	*L*	*N*	*P*	*P*	*P*	*X*	*X*
		CVE-2016-0128	*Nw*	*M*	*N*	*P*	*P*	*N*	*X*	*X*
		CVE-2017-0267	*Nw*	*M*	*N*	*P*	*N*	*N*	*X*	*X*
	Windows-server-2008 (192.168.9.57)	CVE-1999-0524	*Lc*	*L*	*N*	*N*	*N*	*N*	*X*	*X*
		CVE-2011-0657	*Nw*	*L*	*N*	*P*	*P*	*P*	*X*	*X*
		CVE-2016-0128	*Nw*	*M*	*N*	*P*	*P*	*N*	*X*	*X*
		CVE-2017-0143	*Nw*	*M*	*N*	*C*	*C*	*C*	*X*	*X*
Case-2	Windows-7-Vista (192.168.9.70)	CVE-1999-0524	*Lc*	*L*	*N*	*N*	*N*	*N*	*X*	*X*
		CVE-2009-2532	*Nw*	*L*	*N*	*C*	*C*	*C*	*X*	*X*
		CVE-2009-3103	*Nw*	*L*	*N*	*C*	*C*	*C*	*X*	*X*
		CVE-2011-0657	*Nw*	*L*	*N*	*P*	*P*	*P*	*X*	*X*
		CVE-2016-0128	*Nw*	*M*	*N*	*P*	*P*	*N*	*X*	*X*
		CVE-2017-0143	*Nw*	*M*	*N*	*C*	*C*	*C*	*X*	*X*
Case-3	Windows-10 (192.168.9.81)	CVE-1999-0450	*Nw*	*L*	*N*	*P*	*P*	*P*	*X*	*X*
		CVE-2011-3389	*Nw*	*M*	*N*	*P*	*N*	*N*	*X*	*X*
		CVE-2016-2183	*Nw*	*L*	*N*	*P*	*N*	*N*	*X*	*X*
	Ubuntu 14.04 (192.168.9.83)	CVE-2009-4135	*Lc*	*M*	*N*	*P*	*P*	*P*	*X*	*X*

AV Attack Vector, *AC* Attack Complexity, *Au* Authentication, *CI* Confidentiality Impact, *II* Integrity Impact, *AI* Availability Impact, *ECM* Exploit Code Maturity, *RL* Remediation Level, *Nw* Network, *Lc* Local, *M* Medium, *L* Low, *N* None, *C* Complete, *X* Not Defined

Fig. 16. Vulnerability Assessment Results

No.	Virtual machine	CVE-ID	Patch file
Case-1	Windows-7 (192.168.9.56)	CVE-1999-0524	Patch is not available
		CVE-2011-0657	Windows6.1-KB2509553-x64.msu
		CVE-2016-0128	Windows6.1-KB3149090-x64.msu
		CVE-2017-0267	Windows6.1-kb4012212-x64_2decefaa02e2058dcd965702509a992d8c4e92b3.msu
	Windows-server-2008 (192.168.9.57)	CVE-1999-0524	Patch is not available
		CVE-2011-0657	Windows6.1-KB2509553-x64.msu
		CVE-2016-0128	Windows6.1-KB3149090-x64.msu
		CVE-2017-0143	Windows6.1-kb4012212-x64_2decefaa02e2058dcd965702509a992d8c4e92b3.msu
Case-2	Windows-7-Vista (192.168.9.70)	CVE-1999-0524	Patch is not available
		CVE-2009-2532	Windows6.0-KB975517-x64.msu
		CVE-2009-3103	Windows6.0-KB975517-x64.msu
		CVE-2011-0657	Windows6.1-KB2509553-x64.msu
		CVE-2016-0128	Windows6.1-KB3149090-x64.msu
		CVE-2017-0143	Windows6.1-kb4012212-x64_2decefaa02e2058dcd965702509a992d8c4e92b3.msu
Case-3	Windows-10 (192.168.9.81)	CVE-1999-0450	Patch is not available
		CVE-2011-3389	Windows6.1-KB2509553-x64.msu
		CVE-2016-2183	Windows6.1-KB3149090-x64.msu
	Ubuntu 14.04 (192.168.9.83)	CVE-2009-4135	Patch is not available

Fig. 17. Patch matching with CVE-ID

The strong point of this framework [11] is that it can perform VM-specific vulnerability scanning and evaluation, it can consider the virtual machine's flexibility, it can scan for potential and published vulnerabilities to patch them, it can identify highly severe vulnerabilities and apply patches, as well to patch them before the exploitation is conducted. Moreover, the framework can analyze the characteristics of vulnerability and find VMs which have capability to be used for exploitation that analyzed vulnerabilities, also, it can derive the level of risk for virtual machine via the analysis of risk and rank the VMs with the support of risk analysis to prioritize them. Last but not least, the good point of this frame can be seen from the way it is configured to be isolated from hypervisor and virtual machines. It is installed on each physical server that requires root capabilities to connect the framework. As a result, it is safe from both external and internal attackers. In my own opinion, this framework should be enhanced to restrict the possible cyberattacks that may be executed using unpatched or zero-day vulnerabilities.

III. Conclusions

In conclusion, by summarizing and diving into each method from each paper we can have a clear view about the ways that various models and framework have been designed and developed to detect and restrict the vulnerabilities in several types of technology. I have found some interesting things and challenges about the way to detect and prevent vulnerability while doing this assignment. First of all, we can leverage the existing tools and features of the operating systems to analyze and detect security vulnerability based on that. To deal with zero-day or unknown vulnerability, we can build a model to analyze the characteristics of the existing vulnerabilities in the published database to gain the insight view about the vulnerability, in order to detect the unknown vulnerability in our system and network. But the most interesting thing here is that the way we can apply Machine Learning Algorithm in the process of discovering and assessing security vulnerability in the software. For the emerging technology such as Cloud Computing or Virtualization, it still had many restrictions to create a proper framework and system to analyze and detect vulnerability in that environment, but we can still build system that leveraging published vulnerability databases to examine and detect cloud security vulnerability based on the records and cases included inside those databases. We can do the same way when consider the vulnerability prevention method for virtualization, by preventing security vulnerability at the initial stage. For example, we can create a secure tunnel and any virtual machines (VMs) that want to join the running virtual environment, need to go through that tunnel in order to be scanned and detect any vulnerability that still exists inside the newly added virtual machines. Although, we have researched and provide insight view about various models and systems to prevent security vulnerability, but the models and systems need to be updated day-by-day because of the rapid increase in both technology and vulnerability nowadays.

IV. References

[1] "Microsoft Vulnerabilities Report 2021." https://www.beyondtrust.com/resources/whitepapers/microsoft-vulnerability-report (accessed Jul. 04, 2021).

[2] M. Fujimoto, W. Matsuda, and T. Mitsunaga, "Detecting attacks leveraging vulnerabilities fixed in MS17-010 from Event Log," in *2019 IEEE Conference on Application, Information and Network Security (AINS)*, Nov. 2019, pp. 42–47. doi: 10.1109/AINS47559.2019.8968703.

[3] U. K. Singh, C. Joshi, and D. Kanellopoulos, "A framework for zero-day vulnerabilities detection and prioritization," *J. Inf. Secur. Appl.*, vol. 46, pp. 164–172, Jun. 2019, doi: 10.1016/j.jisa.2019.03.011.

[4] V. Kumar and D. Sinha, "A robust intelligent zero-day cyber-attack detection technique," *Complex Intell. Syst.*, May 2021, doi: 10.1007/s40747-021-00396-9.

[5] D. Bassolé, Y. Traoré, G. Koala, F. Tchakounté, and O. Sié, "Detection of Vulnerabilities Related to Permissions Requests for Android Apps Using Machine Learning Techniques," in *Proceedings of the 12th International Conference on Soft Computing and Pattern Recognition (SoCPaR 2020)*, Cham, 2021, pp. 810–822. doi: 10.1007/978-3-030-73689-7_77.

[6] Y. Malik, C. R. S. Campos, and F. Jaafar, "Detecting Android Security Vulnerabilities Using Machine Learning and System Calls Analysis," in *2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, Jul. 2019, pp. 109–113. doi: 10.1109/QRS-C.2019.00033.

[7] Y. Zhang, K. Chen, and Y. Lian, "VulLocator: Automatically Locating Vulnerable Code in Binary Programs," in *Information Security Practice and Experience*, Berlin, Heidelberg, 2013, pp. 295–308. doi: 10.1007/978-3-642-38033-4_21.

[8] P. Kamongi, S. Kotikela, K. Kavi, M. Gomathisankaran, and A. Singhal, "VULCAN: Vulnerability Assessment Framework for Cloud Computing," in *2013 IEEE 7th International Conference on Software Security and Reliability*, Jun. 2013, pp. 218–226. doi: 10.1109/SERE.2013.31.

[9] A. Sun, G. Gao, T. Ji, and X. Tu, "One Quantifiable Security Evaluation Model for Cloud Computing Platform," in *2018 Sixth International Conference on Advanced Cloud and Big Data (CBD)*, Aug. 2018, pp. 197–201. doi: 10.1109/CBD.2018.00043.

[10] G. Zhu, Y. Yin, R. Cai, and K. Li, "Detecting Virtualization Specific Vulnerabilities in Cloud Computing Environment," in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, Jun. 2017, pp. 743–748. doi: 10.1109/CLOUD.2017.105.

[11] R. Patil and C. Modi, "Designing an efficient framework for vulnerability assessment and patching (VAP) in virtual environment of cloud computing," *J. Supercomput.*, vol. 75, no. 5, pp. 2862–2889, May 2019, doi: 10.1007/s11227-018-2698-6.

[12] M. Satheesh Kumar, J. Ben-Othman, and K. G. Srinivasagan, "An Investigation on Wannacry Ransomware and its Detection," in *2018 IEEE Symposium on Computers and Communications (ISCC)*, Jun. 2018, pp. 1–6. doi: 10.1109/ISCC.2018.8538354.

[13] D.-Y. Kao and S.-C. Hsiao, "The dynamic analysis of WannaCry ransomware," in *2018 20th International Conference on Advanced Communication Technology (ICACT)*, Feb. 2018, pp. 159–166. doi: 10.23919/ICACT.2018.8323682.

[14] Y. Afek, A. Bremler-Barr, and S. L. Feibish, "Zero-Day Signature Extraction for High-Volume Attacks," *IEEEACM Trans. Netw.*, vol. 27, no. 2, pp. 691–706, Apr. 2019, doi: 10.1109/TNET.2019.2899124.

[15] J. David and C. Thomas, "Efficient DDoS flood attack detection using dynamic thresholding on flow-based network traffic," *Comput. Secur.*, vol. 82, pp. 284–295, May 2019, doi: 10.1016/j.cose.2019.01.002.

[16] G. S. Kc, A. D. Keromytis, and V. Prevelakis, "Countering code-injection attacks with instruction-set randomization," in *Proceedings of the 10th ACM conference on Computer and communications security*, New York, NY, USA, Oct. 2003, pp. 272–280. doi: 10.1145/948109.948146.

[17] M. Costa *et al.*, "Vigilante: end-to-end containment of internet worms," in *Proceedings of the twentieth ACM symposium on Operating systems principles*, New York, NY, USA, Oct. 2005, pp. 133–147. doi: 10.1145/1095810.1095824.

[18] A. Baratloo, N. Singh, and T. Tsai, "Transparent run-time defense against stack smashing attacks," in *Proceedings of the annual conference on USENIX Annual Technical Conference*, USA, Jun. 2000, p. 21.