東京工科大学

博士学位論文

Remote Detection Method for Operating Environment in Cyber Security Attack and its Countermeasures

サイバーセキュリティ攻撃における動作環境の遠隔検出法とその対抗策

平成 28 年 9 月

Noor Afiza Binti Mat Razali

Abstract

Virtual machines (VMs) have evolved from just being able extend desktop and server management to being used as security systems to help identify malware and other security platform detection systems. With the rapidly expanding usage of mobile devices, VMs are now commonly being used as emulators for scanning and detecting malware intrusion that are embedded in mobile applications. This is largely due to the constrained resources in mobile devices such as processing power, memory etc., that prohibit high end malware scanners to be executed in mobile devices. However, scanning and detecting malware processes using VMs may cause serious security threats to the end users when malware can detect its current running environment, it may change its behavior in such a way that it doesn't perform malicious operations if it detected that it is running on VM environment as an emulator.

In this dissertation, a proposal for network timestamping could be used as a potential tool to remotely detect operating environment changes that manipulate system processes and files to minimize the impact and reduce the security threats. This research shows that operating environment could be remotely detected by using IP timestamping information in network packets. The proposed method and process is to utilize successive timestamps to track how many times identical timestamps are stamped between the packets. There are differences between the numbers of successive identical timestamps replied from the VM environment and that of the real machine environment.

This research has differentiated and detected the target operating environment using the proposed method. A countermeasure technique to disguise IP timestamps characteristic from real machine such that it shows similar IP timestamp patterns as the VMs is proposed. By using this technique, malware may not be able to differentiate between a real machine and VMs. In the case of mobile devices, meanwhile the identical timestamps are frequently stamped for Android installed on VM environment; identical timestamps are never stamped for Android on real mobile device environment.

One potential usage of the method is by implementing it into video streaming application that hide the actual purpose of the application in detecting

target machine operating environment using the method. For future work, more characteristic patterns need to be obtained and be used in proposing more comprehensive mitigation. The remote detection method and relationship with malware behavior could be implied in bigger framework during security policy implementation to avoid cyber security attack by taking consideration of malware behavior could be change based on the detection of the operating environment.

Contents

List of Figures

List of Tables

Page

CHAPTER 1

Introduction

This chapter briefly introduces the background, the objectives, significance of this research and the summaries of each chapter of the dissertation.

1.1. Background

Computing technology is continually enhancing nearly all aspects of how businesses are done and the personal life of humans. It becomes an extension of the individual, making environment smarter, contextually aware, and better connected. Devices are also becoming smarter and more connected, and businesses are building deeper real-time connections with their suppliers, partners, governments and customers. This enables the collection and selectively sharing vast amounts of data. Device will continue to grow in volume and variety. Forecast for connected devices by 2020 are 200 billion and climbing [1]. Data sharing also made easier with cloud computing.

Cloud computing has become the new paradigm in networked computing and it has been identified as the utility after electricity, water, gas and telephone [2]. Cloud computing is a computing technique which provides the options for sharing and renting of storage infrastructure, computing services, software, applications customization and many more from remote platforms [3]. It offers organizations, governments and individuals with a cost-effective utility by delivering software and services over the Internet [4]. The basic principal behind resource sharing and delivering scalable services in cloud computing is virtualization [5]. Thus, making virtualization as one of the important pillars for cloud computing. Virtualization technologies allow multiple operating systems and applications run on the same machine that resulted on effective time and low cost for deployment.

Meanwhile, mobile devices are rapidly emerging as popular appliances that are being used by consumers due to its mobility and easy access to the Internet,

especially through the usage of Wi-Fi facilities. Android is widely used as one of the operating systems in mobile devices. According to Gartner Report [6], the Android OS's market share was 79% in August 2013 and it will keep increasing. Due to the advancement in mobile device technologies, the latest mobile devices are now able to perform many of the operations that had been exclusively done on PCs.

With the high dependencies of computing technologies in the current world scenario and the outburst usage of personal mobile devices that are utilizing the easy connectivity to cyberspace, the protection of sensitive and valuable information assets is becoming more challenging [7]. While most organizations, governments and personal users deployed security measures, the number of cyber security attacks incidents continue to increase and becoming an ever-increasing threat for organizations, governments and personal users around the world [8]. Moreover, mobile devices use the same architecture as traditional computers; thus they have the same vulnerabilities and security issues faced by personal computers [9]. Especially, since Android is an open, programmable software framework that essentially provided the attackers with the inside knowledge of the platform. Furthermore, mobile devices such as smartphone are constrained by their limited resources, i.e., processing power, battery power, and lack of storage, which prevents the integration of advanced security monitoring solutions that work with traditional personal computers. In recent studies, it was found that in 2011, 96 percent of smartphones and tablets do not have the necessary security software [10]. This increases the vulnerability of typical mobile device to critical attacks that can make the mobile devices unusable.

A breach in information protection could impact severely to the organizations, such as loss of profit, trusts, reputation and loss of lives if such breach occurred at any government critical infrastructures [11, 12]. Majority of the attackers are also driven by monetary motivation by selling or leveraging stolen sensitive or confidential financial information from personal individuals [13], such as credit cards and online bank accounts information. The stolen information cost billions of US Dollars of lost every year [14].

Hence, research in cyber security is a very important field that will provide the information, knowledge and protection for the organizations, governments and even individuals against those attacks from external or internal malicious attackers.

This dissertation studied one component of the security vulnerabilities, which is the remote detection of operation systems that could be exploited by the attackers. Remote detection of running environment could be exploited by malware by changing its behavior in different environments, especially upon detecting virtual environment that could be used as deception environment such as honeypot. This could lead to more vulnerability in the cyber security because amount of malware that harm or compromise user privacy has increased dramatically [15].

Related work that had been done by other researchers, [16] and [17] on virtual environment detection are focusing on differentiating the network behavior between virtual machine (VM) and real machine using timestamp. In the studies, the real machine environment was only limited to desktop machine without taking into consideration mobile devices. This created gap in the research area since in the current world scenario, mobile devices are becoming the main devices that are being used by organizations in daily operations.

In this dissertation, research was done by taking into consideration the current actual IT implementation in organizations. The main focus is on remote detection method using network timestamps to differentiate the operating environment on high performance machine such as servers that mainly host VMs and on mobile devices that are normally equipped with limited resources such as processing power. This research aims to validate the applicability of the remote detection method as a potential vulnerability in cyber security attack. Differentiation of operating environment, either they are VM, real machine or mobile device become very important because of the significant growth in the popularity of smart phones with seamless interconnectivity and increasing number of available mobile apps downloaded in smart phones.

This research contributes to the new knowledge on the remote detection of operating systems using the network timestamps analysis characteristic patterns in

recent technology scenario, which includes mobile devices. Base on the characteristic pattern obtained from this research, countermeasure was proposed to hide the differences observed and this research serves as initiative in improving the cyber security that focusing on remote detection of operating systems using network timestamps.

For this research, experiments were conducted in Kinoshita Lab at Tokyo University of Technology to obtain results from the utilizations of IP and ICMP timestamps characteristic as a method for remote detection of operating environment. The results from the studies had shown that the IP and ICMP timestamps characteristic could be used to remotely distinguish the running operating system on VMs, real physical machines and mobile devices. This research also proposes a countermeasure to disable the remote detection method between VM and real machine by hiding the timestamps differences.

Previous work by Kohno [16] highlighted the potential of remote detection method by using network timestamps. Then, work by Shimamura [17] explored on characteristic pattern, based on differences between IP and ICMP timestamps in one packet. In this dissertation, in order to determine the characteristic pattern of network timestamps which are focused at IP and ICMP timestamps in various environments, characteristic patterns as following are being used as the method of analysis:

1) Differences between 2 successive timestamps in the replied packets
2) How many times identical timestamps was stamped between the packets

This research explores remote detection method by using characteristic pattern differences between IP and ICMP timestamps in two measurements mentioned above and also validate remote detection method by using characteristic pattern differences between IP and ICMP timestamps that proposed in [17] in current technology scenario. This dissertation provides the initial finding for characteristic patterns on how network timestamps differences in of 2 successive timestamps and how many same timestamps stamped in the packets could be used as a remote detection method. This dissertation also serve as the research that highlight the characteristic patterns for remote detection that encapsulate the need

to study of the recent IT scenario that have machines that are improving rapidly in term of performance but at the same time, mobile devices with limited performance such as mobile devices are also being widely used.

Experimental procedure for this dissertation started with experiments to determine characteristic pattern in fundamental VM technologies using full virtualization technology using popular virtualization products for open platforms i.e., VMWare, Oracle VirtualBox and Xen. Next, based on the characteristic pattern, countermeasure was proposed and the validations experiments of the countermeasure were done. Finally, to determine the characteristic pattern for mobile devices, experiments using Android as mobile operating system were executed. These experiments aimed to validate current network timestamp behavior of Android. It was also to determine the possibility of wide application of remote detection method in Android as running environment. This could contribute to new knowledge in analyzing the remote detection method of mobile device environment. Discussion on the proposed countermeasure for android is also included in the dissertation on how it could be implemented to provide a basic solution for the remote detection method by disabling the different in network patterns of VM and real machine including the mobile devices in the same environment

This research contributes to the new knowledge of the analysis and countermeasure of remote detection of operating systems using the IP and ICMP timestamps characteristic for distinguishing operating systems running on virtual machines, real physical machines and mobile devices. It is hoped that the results from this research could be utilized for improving the protection of sensitives and valuable information assets.

1.2. Research Objectives

The objective of this research is to study, validate and analyze the potential of using network timestamp, which are IP and ICMP timestamp characteristic, as a method to remotely distinguish operating environment running on VM, real physical machine and mobile device. This research argues that such remote

detection method could become critical vulnerability for cyber security in an event of attacks by malicious attackers. Experiments were conducted to determine and analyzed the characteristic patterns of IP and ICMP timestamps on various machines. This research hypothesized that based on the analyzed characteristic patterns of IP and ICMP timestamps, it could remotely distinguish the operating environment on VM and non-VM environment including mobile device.

A proposed countermeasure that could be implemented in order to disable the remote detection between VM and real machine environment will also be the objective of the research. The countermeasure shall become one of the basic methods that could be extended in the future research in this area.

1.3. Problem Statement and Significance of Research

Building transparent VM are still an ongoing progress and more researches are required to make VM indistinguishable from non VM environment and mobile device. Currently, researches showed that malware are able to exploit the vulnerability of detecting the target operating environment, specifically detecting VM environment [18]. By detecting VM environment, these malware would be able to behave accordingly to their target environment in order to avoid from been detected by security applications that are normally implemented in VM environments. Through this detection also, malware will be able to hide itself from security applications thus escaping from their behavior being revealed and studied. In this research, a remote detection method that was first proposed by [16] was implemented to further studies the implication of such remote detection in the current computing environment that as mentioned previously, implementing core virtue of virtualization in cloud computing and widespread use mobile devices in daily lives of the society.

This research focus in performing behavioral analysis from experiments result to see if there are differences in timestamp between high performance VM and non-VM environment including mobile device. This research is highly significant as the research extended the work done by [16] to further analyzed the method in the current computing scenario. This research also contributes to new

knowledge in analyzing the method towards mobile device environment and a proposed countermeasure in order to provide a basic solution for the remote detection method by disabling the detection of VM and real machine including the mobile devices in the same environment.

The obtained data shall be use to prove the IP and ICMP timestamp method could be used to determine whether the target environment is running on a VM or stand-alone environment including mobile device in the present computing environment. The test environment in this research simulated the high performance VM environment with latest hypervisors from renowned makers and the latest mobile devices with Android operation system on the date of the experiment. Based on data that were gathered in the experiments, using variation of technology and machines, this research also proposed the countermeasure by changing the timestamp reply process in the real machine to address the issue.

This research could serve as the early research work in gathering the characteristic pattern for timestamp reply behavior in hypervisors technology that are normally used in cloud computing environment and mobile device running on Android operating system. Additionally, with the latest trend of Bring Your Own Device (BYOD) to workplace where employees bring their own personal device with various specification to their workplace could expose the organization's internal network with many security issues and vulnerability [19], [20]. Security issue within the corporation may occur when mobile device that affected with the malware or spoofing tools during Internet connection in non-secure Wi-Fi connection outside of the organization were used to access the corporation internal secure environment. With the remote operating environment vulnerability, malware will be able to detect of host's operating environment within the organizations and may exploit the vulnerability in only to attack and infected real and mobile devices while avoiding VM environments that may be used as security protection system such as honey pots. Through this, the malware could steal the information within the corporation and spread to affect other devices within the organization's internal network [21]

With BYOD, organizations are also trending in using smart and mobile device that runs on Android, Apple iOS, Apple Mac OS X, Blackberry and etc. This trend is the result of the emerging use of cloud computing environment. Organizations equipped to their employees with mobile devices for work purposes such as smartphone in order to give better mobility in completing their daily task. Due to limited resources in these mobile devices, they will require thin client software applications such as the ones that are related to sales, finance and customer managements to enable the utilization of shared information in the cloud computing environment.

The thin client software applications are normally made available to be downloaded and installed on the devices. However, before the applications could be released to the employees, there are high possibilities that the development, testing and security checking process for the applications will be done using emulator in the VM machines on the cloud computing environment. Test results might not give the true results, especially in term of security testing against various malicious software or malware because the malware may not show their behavior if they had detected that the target running environment are VM.

As a result, when the application released, the mobile device and other stand-alone environments might be compromised in such a way that the malware will start to execute malicious behavior once it had detected that it is not on a virtual machine environment. Thus, data that are stored or communications through the mobile devices might be revealed to malicious third party. It may also cause expensive billing due to unapproved SMS/MMS subscription services via smartphones [22, 23]. This would create serious consequences for mobile device users that are using Android as an operating system, as any applications that have passed a malware detection system on the VM are considered safe and may gain the user's trust. Furthermore, since mobile devices use the same architecture as PC, it leads to the rapid evolution of mobile device malware where it need only two years for mobile device virus to evolve to a level that computer virus reached in 20 years [23].

As for the personal user's perspective, in parallel with the growth of mobile devices usage, there has been a significant increase in malware aiming at gathering personal information from mobile devices [24]. This information could later be used by the malware owner or third party for their personal profit such as for marketing and selling services on the web or profiteering from online banking information [25]. The growing value of personal data will play a big part and it is already more valuable than payment card information. Increased use of cryptocurrencies such as Bitcoin by personal user will make virtual currency an attractive target for theft, not just the preferred payment method of criminals itself [26]. This growing threat points out the need for users to protect their mobile devices by using anti-virus or anti-malware applications [27]. But implementing such anti-virus or anti-malware applications on mobile devices may not be suited for majority of mobile devices due to the limited resources of CPU, memory and battery power [28], [29].

In order to conserve mobile resources and as a security solution due to the lack of resources available in mobile devices, the integration of mobile devices and cloud computing technology is being proposed by delegating security monitoring and malware detection to VMs in cloud computing facilities [6], [30]. An off-device in-cloud network service could be implemented [31]. Through this solution, it could improve the protection of mobile devices from malware threats [32]. With this approach also, since security services are delegated to VMs in the cloud system for scanning and protecting mobile device applications, it could free up on-device CPU and memory resources of the mobile devices while conferring a high level of malware protection, providing that the mobile devices are connected to the internet.

However, despite the attractiveness of this idea, this research argues that malware detection security system using VM may have critical vulnerability. That is, the malware may try to first detect the environment in which it will be running. Through such detection, malware creators may write programs that will not perform harmful operations such as botnet attacks upon detecting VM environment as the running environment, thus reducing the risk for their behavior from being studied and revealed. If the malware could detect their running environment and choose not to show their behavior in VMs, the mechanism of off-device in-cloud network service will not function well. Therefore, security tests aimed at

applications for mobile devices may not be effective since malicious programs are hiding their true nature once detecting that the running environment is on VM. Thus, security system such as signature-based detection in the VM might not capture the correct signature data [33]. As a result, when the applications are released, mobile devices that install the applications might be compromised even though the devices are installed with anti-virus and anti-malware applications.

With the above arguments, it had shown the significant of the vulnerability through the remote operating environment detection by malware. Malware programmers could design malware that first try to detect whether the system is running on a VM or not before executing any malicious or security breaching operations. Moreover, once that point is reached, the attacks can escalate from just VM detection to the exploitation of the VM itself [32], [34]. This creates a critical vulnerability since malware that has avoided detection in the VM may be downloaded to end user mobile devices as trusted applications. In addition, VM implementations range from those on known to those on unknown hardware configurations on various platforms, and hypervisors and VM detection spans a spectrum of scenarios that need to be investigated. This research believes those intensive studies should also look into VM detection methods and the capability of malware to differentiate VM or mobile device environment.

Therefore, this research examined and analyzed the characteristic patterns of IP and ICMP timestamps from VM and non-VM environments including mobile device in order to determine either by using characteristic patterns of IP and ICMP timestamps, the target operating environments could be distinguished. It is hoped that through the results reported in this thesis, could be useful in strengthening the security issue in cyber security that may be caused by exploiting the detection of VM and non-VM environment. Any loophole that may give risk to security due to malware attack will need to be addressed and handled. Thus by investigating possible methods for detecting VM and proposing ways to countermeasure the detection methods will provide important contribution towards closing down the possibility for malware in hiding their behavior from VM and choose to execute in mobile environment.

1.4. Summary of Chapter

This dissertation comprises of seven chapters. A brief summary of each chapter is organized as follows:

Chapter 1 discusses the background, the overall objectives and the significance of the research to the practical applications.
Chapter 2 gives a brief literature review of related studies and also related technology related to this dissertation.
Chapter 3 discusses this dissertation proposal of remote detection method using IP timestamps.
Chapter 4 discusses remote detection method in full-virtualization VM technology on high performance machine.
Chapter 5 gives a brief overview of proposed countermeasure to resolve the operating environment detection method using IP timestamps pattern characteristic.
Chapter 6 gives a brief overview of characteristic patterns of timestamps from Android operating system on mobile device and comparison with VM.
Chapter 7 discusses and concludes the issues addressed in the dissertation and it will also discuss the impact, limitations of the research and the future work.

CHAPTER 2

Literature Reviews

This chapter discusses the literature reviews on related technologies for this research and previous works in VM operating environment detection methods. It gives the overview information for the technical background of this research.

2.1. Overview

Malware that behave accordingly to running environment could become the threat to cyber security [11]. Researchers showed that malware that are able to exploit the vulnerability by detecting their running environment and escaped from being detected by security applications that normally implemented on VM environments and avoid from being revealed and studied.

Virtualization in computing technology or also known as a hypervisor is a technology that runs on a physical computer and hosts one or more virtual machines on top of the physical machine. Virtualization helped to free organizations and users from physical interface and resource constraints [35], [36]. VMs enhance software interoperability, system impregnability, and platform versatility. Virtualization could be done in many forms such as process, storage and network virtualizations [37]. The virtual machines could simulate physical computers which then could run software and programs such as operating systems, end-user applications and more.

Virtualization is also the core technology that empowers cloud computing [38], [39]. Cloud computing is the current trending service model for organizations in enabling convenient, on-demand network access to a shared pool of configurable computing resources, (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [40]. At a hardware level of cloud computing, a number of physical devices, including processors, hard drives and network devices, are

located in data centers, independent from geographical location, which is responsible for storage and processing needs [41]. It leverages virtualization technology in lowering the costs for the service users and improving utilization and management capabilities for the cloud service providers [42].

With the vast usage of cloud computing in the current world scenario, cyber security issues in hypervisors and VM is becoming more important than ever as any issues that arise in the hypervisors and VM shall be inherited in the cloud computing services [43], [44], [45]. Moreover, with the exploded usage of mobile devices that uses cloud computing services or are integrated with cloud services [46], further increases the significant of the research in VM security as the security breach using the vulnerability in VM could affect mass users around the world that are using mobile devices [47].

The detection of VM operating environment could become one of the potential vulnerabilities of VM that could be exploited by malicious attackers [18], [48], [49] [50]. As discussed in the previous chapter, through the detection of VM operating environment, attackers could design malware that selectively infect and operates malicious actions on real machines or mobile devices while avoiding and hiding its malicious actions on VM operating environment. Furthermore, malware targeting mobile devices is growing[51]. Malware could exploit the detection of the targets running operating environment to differentiate between mobile device and virtualization environment because malware analysis are normally being held in VM environment [52], [53].

This chapter provides information on the literature reviews of the basic technologies and the related works in VM operating environment detection method that were used at the basic references in this research. It covers the literature on virtualization, related works on VM operating environment detection methods, network timestamps and malware.

2.2. Cyber Security and Cyber Threats

With the new era of the Internet and introduction of cloud computing that implements distributed computing resources, information assets safeguard in cyber security is ever becoming a global interest and importance towards organizations, government and personal users [54], [55], [56], [57]. In any case of cyber-attacks or breaches of sensitive's information in organizations and governments information networks, it may causes in the loss of reputations, trust, economically in terms of financial and even threat to national defense if such attacks are originated from terrorist groups [58], [59], [60], [61]. Threats from cyber space include Distributed Denial of Service (DDoS) attacks, Advanced Persistent Threats (APTs) on organizations and governments network facilities and threats to mobile operating systems on mobile networked computing [57].

Furthermore, due to the popularity of smart mobile devices and as a result form the emergence of the Internet of Things (IoT) technologies that employed collection of smart mobile devices that interact on a collaborative basis and resources, cyber security attacks and cybercriminal actives are increasing exponentially [62], [63], [64]. Even with the attentions by researcher on mobile devices cyber security threats [65], new and more features and functionality are always being introduced into the mobile devices technologies and this at the same time also introduce new risks. [66] discussed the threat classifications in mobile devices which are as the following;

- Root Enablers - Apps that gain root access to the Android OS, escalating a user's administrative privileges. Rooting can make devices more vulnerable to malicious attack.
- Surveillanceware - Apps that remain invisible on the device while surreptitiously engaging in comprehensive device monitoring and data exfiltration. They are typically directly installed by someone with physical access to the device.
- Trojans - Apps that advertise legitimate functionality but surreptitiously perform malicious actions in the background, such as data exfiltration or billing fraud.

With a lot of mobile device connected to open Wi-Fi or unsecured network connection, attacker may use the vast resources of the network to turn mobile

device into a botnet and launch a cyber-attack once connected to organization's internal networks or even to national critical infrastructures [67], [68]. There are some Android applications that when downloaded are capable of accessing the root functionality of devices ("rooted") and turning them into botnet soldiers without the user's explicit consent [69].

Google implement security scanning before apps developer could make the apps downloadable in the Google Play Store. The scanning tools called "Bouncer". However, the malicious apps developer could still find a way to pass through it [70], [71]. Additionally Bouncer only scan for malicious applications but it does not covers on the vulnerability within the applications uploaded to the Google Play Store and attackers may exploits the vulnerability in the applications to attack end users [72]. Attackers could easily and unwittingly download malware to their smart devices or fall prey to "man-in-the-middle" attacks where cyber-criminals pose as a legitimate body, intercept and harvest sensitive information for malicious use.

In 2011, there was a mix of Android applications removed from the Android Market because they contained malware. There were over 50 infected applications - these applications were copies of "legitimate" applications from legitimate publishers that were modified to include two root exploits and a rogue application downloader [69]. Among these malware, Botnets are considered as the biggest challenge. Botnets are used to send email spam, carry out distributed denial of services (DDoS) attacks, and for hosting phishing and malware sites. Botnets are slowly moving towards smart devices since those devices are now basically everywhere, powerful enough to run a bot and offer additional gains for a bot-master such as financial gains as discussed earlier [73], [74]. The bots are then programmed and instructed by the bot-master to perform a variety of cyber-attacks, including attacks involving the further distribution and installation of malware on other information systems.

2.3. Malware

Malware are any kind of hostile, intrusive, or annoying software or program code (e.g. Trojan, rootkit, backdoor) designed to use a device without the owner's consent[24], [75]. Therefore users are required to protect their devices using anti-virus or anti-malware applications. However, overall solutions to address all vulnerabilities are yet to be established due to fast track in malware development and creation. Furthermore, with the existence of metamorphic and polymorphic types of malware, that have the capability to change its operation codes to avoid detection would increase the hardship of protection against malware[75].

Malware could be categorized to three type of malware which are basic, polymorphic, metamorphic malware, In basic malware, the program entry point is transferred to malicious payload so that the malware could obtain the control. Meanwhile, Polymorphic viruses mutate while keeping the original code intact. A polymorphic malware consists of encrypted malicious code along with the decryption module [76]. Polymorphic engine injected in the virus body in order to enable the polymorphic virus. The polymorphic engine generates new mutants each time it is executed. Meanwhile, metamorphic malware can reprogram itself using certain obfuscation techniques so that the reprogram version of malware never look like the original malware [77]. Such malware evade the detections from the malware detector since each new variant generated will have different signature, hence it is impossible to store the signatures of multiple variants of the same malware sample.

There are three main methods for detecting malware. The first method is called Pattern Matching Method. This method compares target malware with its signature. This signature is the malware characteristic pattern in binary. However, with the ever-increasing number of target malware that needs to be compared, malware detection using this method proves to be very challenging. The second method is the Generic Method which is applied for processes in the computer. Operation rules are written in definition file and anti-virus software compares the next operation of a target with the rules. When the next operation determined as out of rules, it will be canceled. The third method is called Heuristic method which

is applied before applications are operated. Programs are analyzed by anti-virus software to determine if any suspicious operation is performed base on its behavior when it was executed somewhere else. For virtual machine, dynamic heuristic method is applied for scanning programs before it is operated. Programs are analyzed by antivirus software and if the program determines as malicious software, it will be isolated.

Behaviors of malware could be studied and analysis in security purposes VM. In order to avoid from being tracked by VM-based security systems, malware may now try to detect the system in which it is currently operating to determine either it is VM environment or not. By such detection, the malicious system could withdraw any harmful operations such as botnet attack and therefore hiding itself from the VM security systems. This argument is supported by various researches, which had proved that malware are able to detect if they are running in Virtual Machine (VM) or not.

2.4. Virtualization

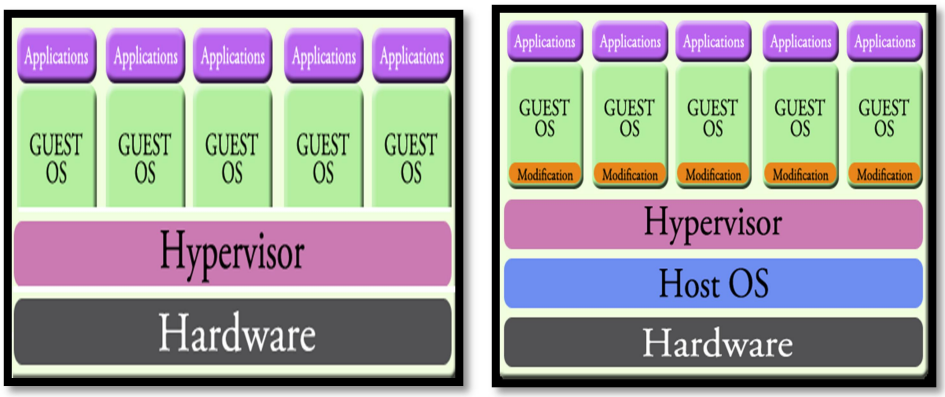
Virtualization is the simulation of the software and/or hardware upon which other software runs. This simulated environment is called a VM. In full virtualization, one or more operating systems (OSs) and the applications are run on top of virtual hardware. Each instance of an OS and its applications runs in a separate VM called a guest operating system. The guest OSs on a host are managed by the hypervisor, which controls the flow of instructions between the guest OSs and the physical hardware, such as CPU, disk storage, memory, and network interface cards.

Hypervisor provides complete independence and autonomy of each virtual server to other virtual servers running on the same physical machine. The hypervisor also monitors and controls the physical server resources, allocating what is needed to each operating system and making sure that the guest operating systems or the VM cannot disrupt each other.

Hypervisors are large pieces of software with 100,000 codes or more that in charge of providing each VM with the illusion of being run on its own hardware by exposing a set of virtual hardware devices such as CPU, memory, NIC and storage [78]. The hypervisor can partition the system's resources and isolate the guest OSs so that each has access to only its own resources, as well as possible access to shared resources such as files on the host OS. Also, each guest OS can be completely encapsulated, making it portable. Some hypervisors run on top of another OS, which is known as the host operating system. VM is one of the underlying technologies in the information technology industry. The VMs are implemented on hypervisor hosts.

Hypervisor supports the creation of a virtual network that connects the virtual network interface card (NIC) to a network that is composed of virtual switches. This virtual network connects to the physical NICs on the host machines and allows applications on VMs to connect to services outside of the hosts. As with other resources in the VM, the hypervisor is the manager of network traffic in and out of each VM and the host. Applications send network requests to the guest operating system which passes the request through the virtual switch. The hypervisor then takes the request from the network emulator and sends it through the physical NIC card out into the network. When the response arrives, it follows the reverse path back to the application. As a result, virtualization adds a number of wrinkles to the networking environment.

There are 2 main types of virtualization as shown in Figure 1. Full virtualization, or bare-metal implementations, run directly on the server hardware without any host operating systems beneath them, whereas para-virtualization run on top a traditional operating system. Para-virtualization is easy to install and deploy because much of the



Type 1 Hypervisor (full virtualization) Type 2 Hypervisor (para-virtualization)

Figure 1: Types of hypervisor

hardware configuration work such as networking and storage is handled by the underlying operating system [79].

In full virtualization, the hypervisor provides most of the same hardware interfaces as those provided by the hardware's physical platform. This means that the OSs and applications do not need to be modified for virtualization to work if the OSs and applications are compatible with the underlying hardware. Full virtualization also uses the hypervisor to coordinate the CPU of the server and the host machine's system resources in order to manage guest operating systems without any modification. In this scenario, the hypervisor provides CPU emulation to handle and modify privileged and protected CPU operations made by unmodified guest operating system kernels.

Meanwhile, in para-virtualization, hypervisor offer interfaces to the guest OS that the guest OS can use instead of the normal hardware interfaces. If a guest OS use para-virtualization interfaces, they offer significantly faster access to resources such as hard drives and networks. Difference types of para-virtualization are offered by difference hypervisor systems.

2.5. Network Timestamps

Various network protocol exists in network timestamp that enables more than one type of network timestamp to be obtained in one packet. IP option type 44 can be attached to request for IP timestamp reply in the IP header. Meanwhile ICMP timestamps request reply can be used to obtain ICMP packets timestamps information.
Timestamp that is used in this research are as following:

IP Timestamp: RFC 781 - IP Timestamp is an optional extension to the IPv4 header that allows the sender to request timestamp values from any machine which handles the packet by specifying its IP address. According to RFC781, the IP timestamp option is a right-justified, 32-bit timestamp in milliseconds since midnight UT [80]. It is primarily used for diagnostics purposes specifically to measure network delay time between gateways.
ICMP Timestamp: RFC 792 - The data received (a timestamp) in the message is returned in the reply together with an additional timestamp. The timestamp is 32 bits of milliseconds since midnight UT. The Originate Timestamp is the time the sender last touched the message before sending it, Received Timestamp is the time the echoer first touched it on receipt, and the Transmit Timestamp is the time the echoer last touched the message on sending it.

IP Header structure with timestamps request option is as Figure 2 and ICMP with timestamps reply request is as Figure 3.

The combination of this IP Timestamps Request Option and ICMP Timestamps reply message are manipulated in this research to create packets that sent to target machine to obtain timestamps data.

2.6. Overview of Android Operating System

Android [81] is an open-source software with special architecture. Android architecture is as per Figure 4. Android architecture composed of five layers:

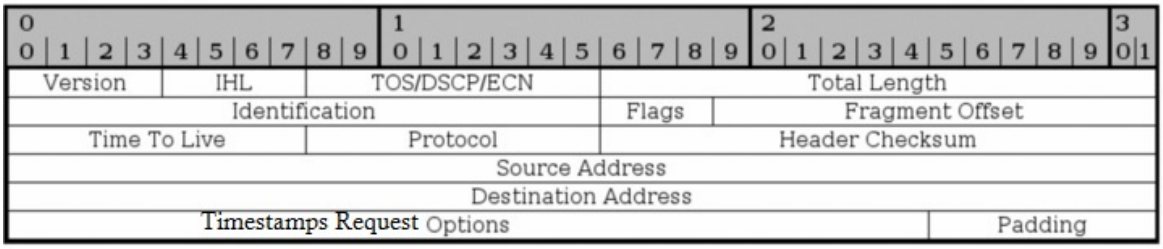


Figure 2: IP header

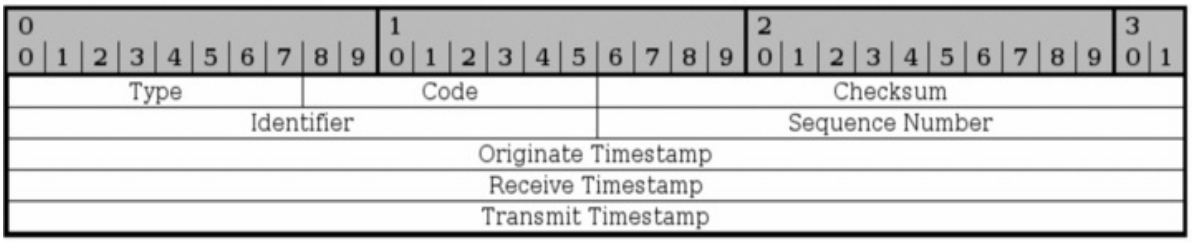


Figure 3: ICMP timestamp/timestamps reply message

Applications, Application Framework, Libraries, Android Runtime and the Linux kernel. The uppermost layer, the Applications layer, provides the core set of applications that are commonly offered out of the box with any mobile device. The Application Framework layer provides the framework Application Programming Interfaces (APIs) used by the applications running on the uppermost layer. Besides the APIs, there is a set of services that enable the access to the Android's core features such as graphical components, information exchange managers, event managers and activity managers.

Below the Application Framework layer, another layer containing two important parts: Libraries and the Android Runtime. The libraries provide core features to the applications. The Android Runtime consists of the Dalvik virtual machine and the Java core libraries. The Dalvik virtual machine is an interpreter for byte code that has been transformed from Java byte code to Dalvik byte code.Dalvik is compiled to native code whereas the core libraries are written in Java and interpreted by Dalvik. Until recently, there are various versions of Android which includes Marshmallow, Lollipop, KitKat and Jelly Bean [81].

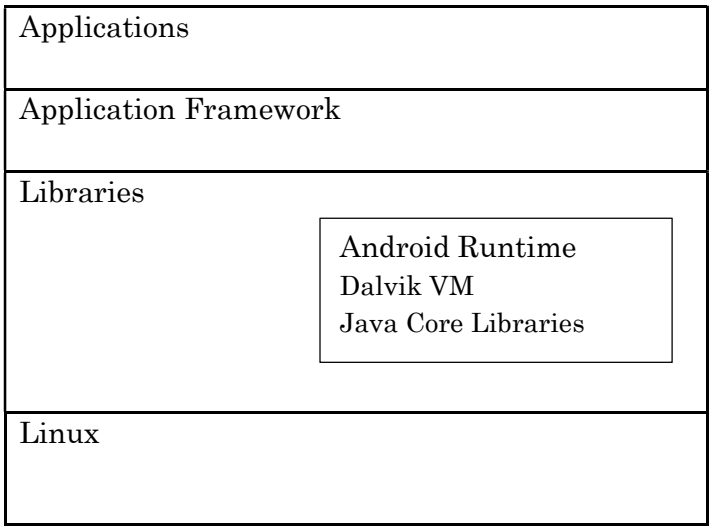


Figure 4: Android system architecture

2.7. VM Operation Environment Detection Methods

VM was not completely transparent to the applications and malware might exploit the detection of VM in not show their behavior if they are running in VM environment. As mentioned previously in the previous chapter, various software which are targeted to be operated in mobile devices are created and tested first in the VM in order to detect any potential harm and abnormality with assumption that the result will be the same when it is executed in stand-alone environment of the mobile devices. However, if malware are able to detect that they are running on virtual environment and not showing their real behavior during the testing phase in VM environment, once the software is installed for users usage on their smart devices, malware might infect and compromised their devices. Thus, research to address this issue need to be done to avoid the risk of users in disposing their information when using mobile devices that are infected by malware that was not detected during the testing in the VM environment.

On the other hand, with the recent emerging of cloud computing, and with more systems and applications are implemented on clouds infrastructure, if malware and attacker could determine they are running on virtual environment, more comprehensive attack to take over the VM itself might be deployed base on the

information that they could obtain. More research needs to be done to understand how VM could be detected and how to improve on that especially in the emerging of cloud computing that utilized the VM technologies.

Various researches [82], [17], [83], [84] discussed about hypervisor detection method. Previous methods for detecting execution within a hypervisor have typically focused on specific artifacts of the implementation, such as hardware naming, guest-to-host communication systems, or memory addresses etc. This research is focusing on detection method that focuses on network implementation and VM behavior.

A technique for remotely detecting hypervisor existence without compromising the target using network timestamp was discussed in [17]. The technique examines discrepancies between two timestamps, IP and ICMP Timestamps in one packet to determine VM presence. When the target hosts use VM, some of the timestamp will show discrepancies because hypervisor sometimes interrupts timestamp operations to complete other operations. Also, hypervisor supports the creation of a virtual network that connects the virtual network interface card (NIC) to a network that is composed of virtual switches.

This virtual network connects to the physical NICs on the host machines and allows applications on VMs to connect to services outside of the hosts. As with other resources in the VM, the hypervisor is the manager of network traffic in and out of each VM and the host. Applications send network requests to the guest operating system which passes the request through the virtual switch. The hypervisor then takes the request from the network emulator and sends it through the physical NIC card out into the network. When the response arrives, it follows the reverse path back to the application. As a result, virtualization adds a number of wrinkles to the networking environment. This mechanism is shown in Figure 5.

Thus, real machine and VM should show the difference in behavior such as following:

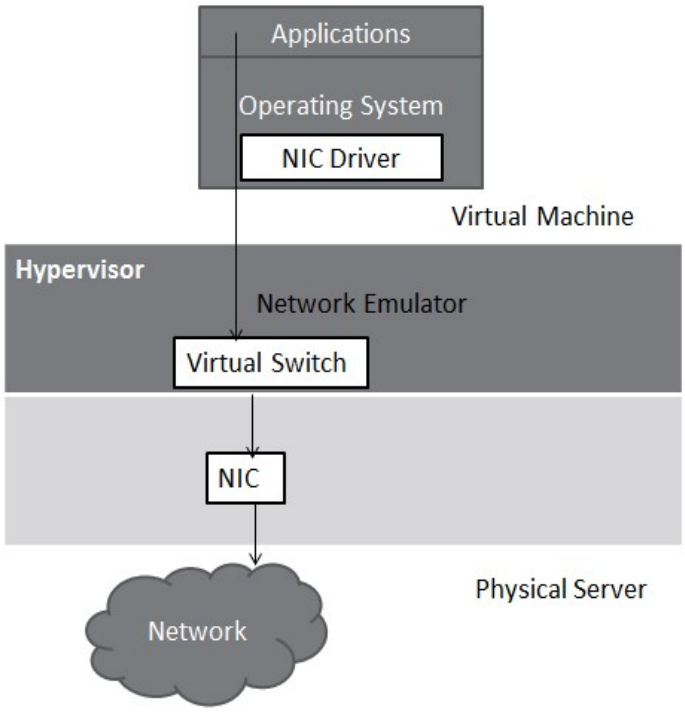


Figure 5: Virtual network path

i. Deviation of the clock.

In VMWare, behaviors of network communications are difference because timer device emulation was held. As a result, the gap can be observed by network timestamp and connection time out.

ii. Packet delivery.

In VM, host network interface that shared packet delivery overhead will exist. For example, in Xen, when packet was delivered, it will deliver to the VM queue and will be processed according to VM schedule and this behavior is difference in real machine because when packet delivered, it will be processed immediately.

iii. Packet operation.

Due to the ability of VM to handle packets, the way packets handled will be different.

IP timestamp option of an Internet Protocol version 4 (IPv4) packets also could be exploited to exchange a secret message as a covert channel. This covert channel will

use the Hypertext Transfer Protocol (HTTP) which usually has a lot of traffic associated with it[85].

2.8. Previous Works

Previous work by Kohno [16] highlighted the potential of remote detection method by using network timestamps. In the study, Kohno introduced remote fingerprinting of physical device. The remote fingerprinting was done without the fingerprinted physical devices known cooperation. The research exploited clock skew from small, microscopic deviations in device hardware and did not require any modification to the fingerprinted devices. The method that was introduced by Kohno showed consistent measurements even when the measurer was thousands of miles and multiple hops away from the fingerprinted device, and even when the fingerprinted device is connected to the Internet from different locations and via different access technologies.

While the work by Shimamura [17] explored the characteristic pattern based on the differences between IP and ICMP timestamps in 1 packet. The study [17] proved that timestamps discrepancies between IP and ICMP timestamps were caused by time correction in VM environment. In [17], experiments were done to detect discrepancies of timestamps in two cases, when ICMP timestamp + 1 < IP timestamp and when ICMP timestamp < IP timestamp. The experimental results showed that discrepancies occurred only 0.12% from total 1,000,000 packets in the case when ICMP timestamp + 1 < IP timestamp for real environment. However, for VMs, discrepancies occurred in the range between 0.1% until 0.5% from total 1,000,000 packets. On the other hand, in the case when ICMP timestamp < IP timestamp, the study found out that 0% of discrepancies occurred from total 1,000,000 packets from real environment and the range of 0% until 2.5% of discrepancies occurrence for VMs.

In this dissertation, both IP and ICMP timestamps replies was obtained and compiled from the experiments. However, the ICMP information was obtained and compiled only for validation purposes of remote detection using method that was discussed in [17].

2.9. Concluding Remarks

Various research [16], [17], [34] already discussed on VM detection method since VMs are introduced. Previous methods for VM detection have typically focused on specific artifacts of the implementation, such as hardware naming, guest-to-host communications systems, or memory addresses. Functional and transparency detection method was discussed in [34] by highlighting detection strategies that look upon the characteristic of logical discrepancies, resource discrepancies and timing discrepancies between VM and non-VM environment. Detection method focuses on the implementation of the VM that was discussed, includes method in targeting hardware sources that contain specific word or command related to VM implementation.

Detection could be also done by using tools that are available on websites. Detection method that emphases on difference in performance for VM and physical hardware also were discussed in [82] [34]. But, as machine that is used to install the VM is continuously improved, the difference according to performance might have changed and tests need be done constantly to verify current situations. A light weight detection method of VM using CPU instruction execution performance stability had been studied in [83]. However, this method requires adjustment to be made in the OS and could lead to instability in the OS itself.

On the other hand, detection methods that focus on the network implementation and behavior of VM could be considered ways of remotely detecting VMs without compromising the target. A VM detection method that uses network timestamps was first suggested by Kohno [16] wherein the TCP timestamp was used as a covert channel to reveal the target host's physical clock skew. Meanwhile in [17], discrepancies between two different kinds of timestamp, IP and ICMP in one packet were used to determine the presence of a VM.

From the literature review, in this research, scope of studies expanded to explore the distinguishable differences of timestamps pattern. Validation using method that implemented in [17] on high performance real machine, virtual machine on high performance machine and also on mobile device that uses Android

as OS, and Android that emulated in VM on high performance machine were done and the results also mentioned in this dissertation. Since blocking ICMP timestamps is not available by default on Android platforms, the detection method using timestamps pattern could prove to be a vulnerability for the mobile devices [86] in this study.

Remote detection using differences between 2 successive timestamps in the replied packets and how many times identical timestamps was stamped between the packets were researched as new analysis method for remote detection using network timestamps in this dissertation.

CHAPTER 3

Research Proposal

This chapter discusses the proposed remote detection method of operating environments using network timestamps characteristic that is used in this research. This chapter also discusses on the hypothesis of expected results of the research based on literature reviews and related works that had been conducted by other researchers [16], [17], [82], [83], [84].

3.1. Overview

Previous work by Kohno [16] highlighted the potential of remote detection method of operation systems by using network timestamps. Meanwhile, the work by Shimamura [17] explored on the characteristic patterns, based on the differences between IP and ICMP timestamps in 1 packet to remotely fingerprinting operation systems. In [17], experiments were done to detect discrepancies of timestamps in two cases, which are when ICMP timestamp + 1 < IP timestamp and when ICMP timestamp < IP timestamp.

As discussed in chapter 1 and 2, with the advancement in computing technologies that contribute to the improvement in machine performances and the increased usage of personal smart mobile devices, it is necessary to perform validations of the remote detection methods discussed above.

In this dissertation, by extending the works done by Kohno and Shimamura, the following new approach of analysis methods were used in order to determine the characteristic patterns of network timestamps (IP and ICMP) in various environments;

1) Differences between 2 successive timestamps in the replied packets
2) How many times identical timestamps was stamped between the packets

Both IP and ICMP timestamps replies was obtained and compiled from the experiments. However, the ICMP information was obtained and compiled only for the validation purposes of remote detection using the method that was discussed in [17].

3.2. Method in Determining Network Timestamps Different

This research explores remote detection method by comparing differences in characteristic patterns between IP timestamps using two new approaches that were based on the works done by Kohno and Shimamura in measuring the discrepancies. This research also performs validations of the remote detection method by using characteristic pattern differences between IP and ICMP timestamps that was proposed in [17] in current technology scenario which includes smart mobile devices.

In [17], experiments were done using machine with 1.86 GHz clock speed. Experiments result in [17] showed that in the case when ICMP timestamp + 1 < IP timestamp, the discrepancies occurred only 0.12% from total 1,000,000 packets for real machine environment. However, for VMs, discrepancies occurred in the range between 0.1% until 0.5% from total 1,000,000 packets. On the other hand, in the case when ICMP timestamp < IP timestamp, the results in [17] found out that 0% of discrepancies occurred from total 1,000,000 packets from real environment and the range of 0% until 2.5% of discrepancies occurrence for VMs.

As part of this study, experiments were done to verify if the same results could still be obtained using the method that was used in [17] on machine with better performance compared to the machine that was used in [17]. However, the experiments that were conducted in this dissertation showed that the same result was not reproducible when machine with higher performance, which is 2.40 GHz clock speed was used.

Since nowadays, machines with higher CPU performance is widely being used, new approach was proposed in this dissertation to remotely detect the operating environment. The new approach methods that are proposed in this dissertation are as the following:

1) Differences between 2 successive timestamps in the replied packets
2) How many times identical timestamps was stamped between the packets

This dissertation also proposed a countermeasure technique to disguise IP timestamps characteristic from real machine such that it shows similar IP timestamp patterns as the VMs.

This dissertation provides the initial findings of network timestamps characteristic patterns using the two new analysis approach mentioned above. The new approach gives new potential as a method for remotely detecting operating environment using network timestamps. This research also could serve as the validation works of the remote detection methods that were previously proposed by Kohno and Shimamura in current world scenario that has improved machines performance compared to the machines that were used in [16] and [17], and also smart mobile devices with low performance which are widely used in daily lives.

In the experiments in this study, packets were sent to the target machines to request for the IP and ICMP timestamps information. The IP and ICMP timestamps information in the replied packets from the target machines were obtained and compiled. The compiled timestamps information was used to determine the characteristic pattern of IP and ICMP timestamps using the analysis methods used in [17] and the new approach discussed above. The ICMP information was obtained and compiled only for validation purposes of remote detection using method that was discussed in [17].

In order to determine the IP and ICMP timestamps, packets were created and sent to the target machines. IP option type 0x44 was inserted in the created packet's IP Header in order to request for IP timestamps reply. In addition, ICMP timestamps request type 13 was also embedded in the created packet to request for ICMP timestamp.

Figure 6 shows the packets, which includes IP and ICMP timestamp request that were used in the experiments. In the created packets, IHL was set to 14, which IP timestamp option including header length 1 is for 32 bit. The total length of the packet is 76 bits, which is the accumulated sum of header and data segment. The

header segment is 56 octets, while 20 octets are for the data segment. The length in the header is 36 bits and the maximum length value is 40. Flag is set to 1 for each timestamp that stamped before the internet address embedded.

As real life application scenario, the packets that request for IP and ICMP timestamps information to the target machines could be embedded in video datagram to avoid from being blocked as potential attacks. Studies in [87] showed that timestamp-based content aware mechanism for video streaming exits, thus provide the prove that timestamps reply request could be embedded in video streaming.

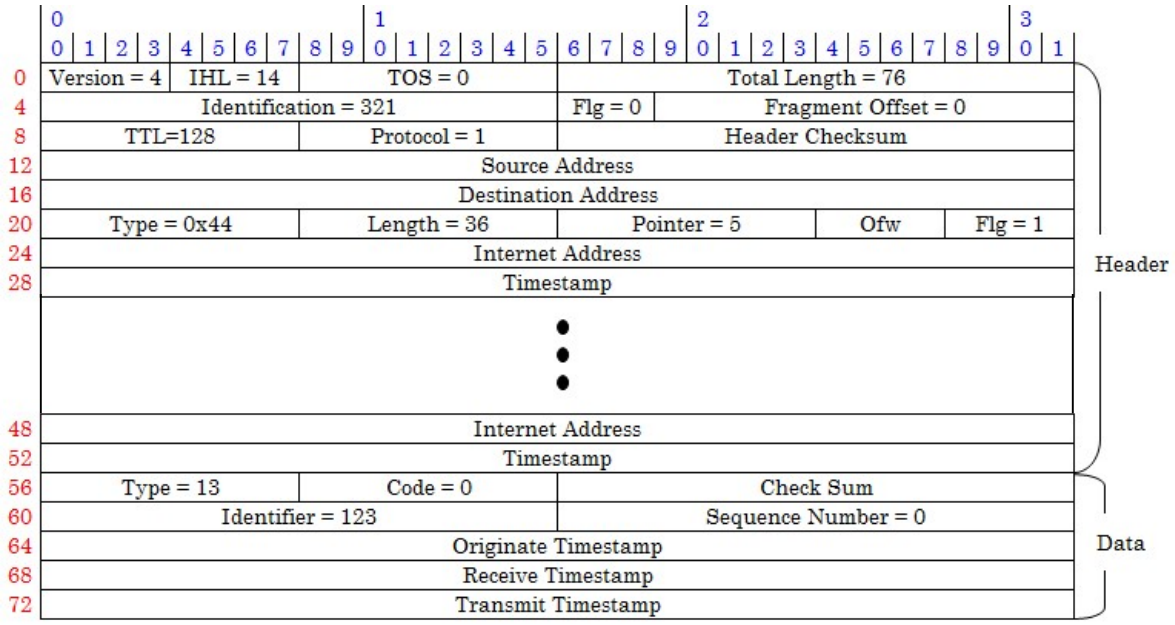


Figure 6: Packet structure

3.3. Experiments Environment

The general network settings for experiments that were conducted in this research are shown in Figure 7. In all of the experiments conducted, 1,000,000 packets were sent from measurer machine to target machine in order to obtain sufficient data to determine the characteristic pattern of IP and ICMP timestamps. These amounts of packets are considered sufficient for the evaluation. The target machines were real machine or VM that was set up in high performance machine. To simulate real life scenario that normally utilize the usage of the high performance machine that hosts the VM, beside of the target VM, another 30 VMs with utilization of more than 80 percent were set up in the high performance machine. Details specification of experiments environment is discussed in the next chapters. For experiments conducted in chapter 4, 2 VM hypervisors that were used are VMWare ESX and VirtualBox. While for experiments conducted in chapter 5, 3 hypervisors which are, VMWare ESX, Xen and VirtualBox were used.

As for the experiments conducted in chapter 6, Sony Xperia mobile device was used as the target real host to obtain data for mobile device. 4 versions of Android operating system which are Ice Cream Sandwich 4.0.4, Jelly Bean 4.2.2, KitKat 4.4.4 and Lollipop 5.0.2 was used as the test mobile operating systems. Android operating system was chosen as the target operating system for the mobile device in this research because Android is commonly being used by majority of mobile devices and it is based on Linux platform that is open to public.

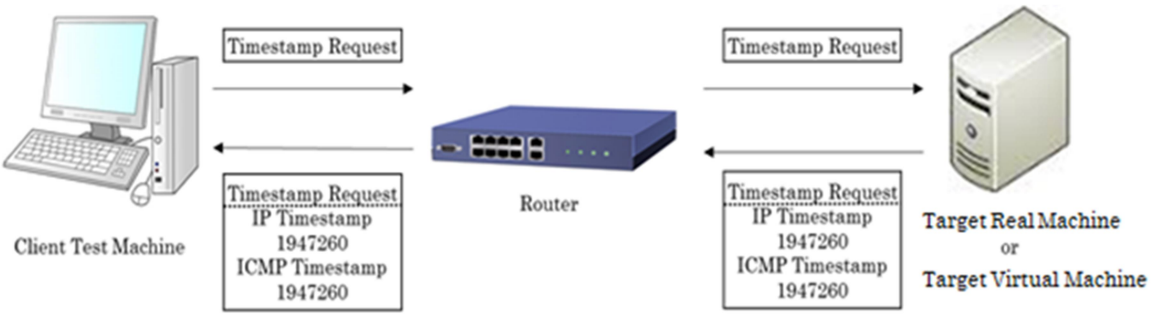


Figure 7: General network settings for experiments

3.4. Analysis Method

In the data analysis, since VM will have queues in handling job for multiple virtual machines, it is expected that IP and ICMP timestamp will show discrepancies between successive packets and also timestamps discrepancies in the same packet. Figure 8 below shows the mechanism on how the network timestamps discrepancies in one packet could occur as proposed by [17]. Timestamp stamping for IP and ICMP might differ due to the reason that VM might switch its operation to other VM. However, as machines performance improves, the data that were published in previous works do not represent the actual the current scenario. Thus, in this dissertation, validations of the method were done to understand current characteristic pattern in high performance machine with 2.40 GHz clock speed and also on smart mobile device that was not tested in previous works.

Due to differences in job handling mechanism between VM and non-VM, IP and ICMP timestamp discrepancies between real environment and virtual environment are expected to occur. Timestamps from real machine on high performance machines are expected to be faster than that of VM due to the behavior of VM itself that share their underlying hardware with the host operating system. Other applications and other virtual machines might also be running on the same host. When more than two guests operated on VM Host, one VM will be suspended to give ways for the other VM to operate. When the VM suspended, the timestamp will be saved and the timestamp value will be re-written when the operation resumed. Because the guest operating system keeps time by counting interrupts, time, as measured by the guest operating system falls behind real time whenever there is a timer interrupt backlog. VM deals with this problem by keeping track of

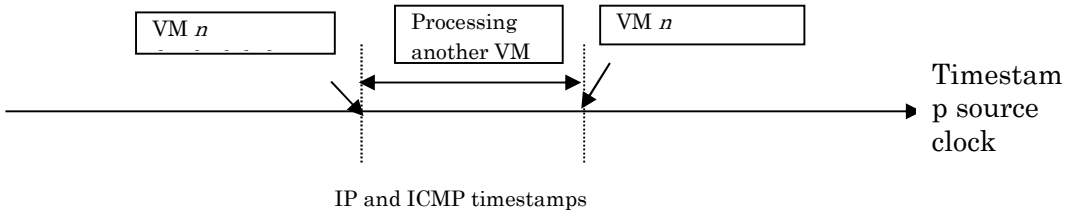


Figure 8: Points that expected to show time discrepancies between IP and ICMP timestamps in VM

the current timer interrupt backlog and delivering timer interrupts in order to catch up. However, time adjustments to catching up could create IP and ICMP discrepancies between the two successive timestamps.

For high performance real machine, this research predicts that quite a number of same timestamps will be stamped in the successive packets because of the speed that the timestamps were stamped is faster than the increase of timestamp source. The differences of timestamps between successive packets were analyzed to characterize the patterns that may occur from VM, real machine and mobile device.

In this dissertation, the analysis method was based on how many same timestamps are stamped in 1 millisecond such is shown in Figure 9 and the difference between two successive timestamps. The number of times when packets were sent are represented as *n* and (*n*+1) until n =1,000,000. Calculation of differences between 2 successive timestamps *n* and *n*+1, … in the replied packets and how many times identical timestamps was stamped between the packets were done. In the experiments, timestamps reply for count *n* is in millisecond unit and different characteristic pattern for VM and non-VM including mobile devices is expected to be disguisable due to machine performance factor and time adjustment mechanism that implemented in VM. Thus, this research anticipates that the results for IP and ICMP timestamps from in the experiments should be as shown in Table 1 below.

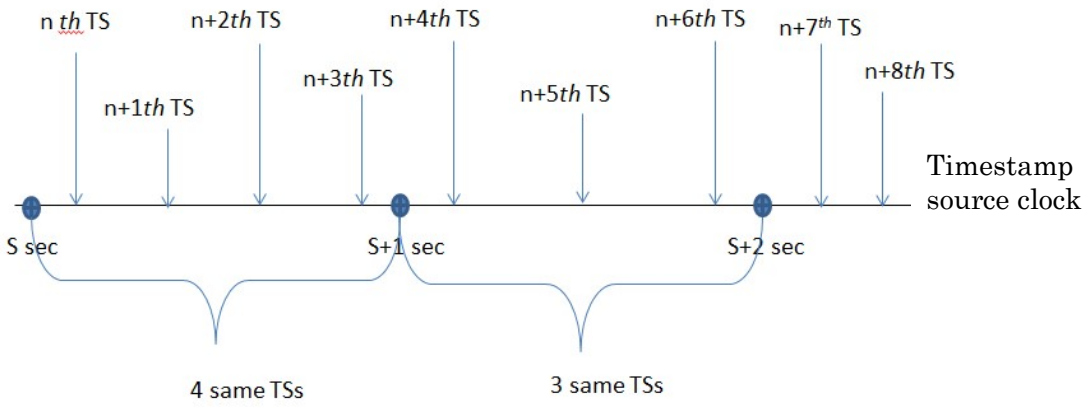


Figure 9: Points that expected to show timestamps discrepancies for successive packets in VM

Table 1: Anticipated experiment results for IP and ICMP timestamps reply

Count (n), …, (n+1) until n =1,000,000	IP Timestamp x10	ICMP Timestamp x10	Differences between two successive IP timestamps	How many same IP timestamps was stamped
1	xxxxxxxx	xxxxxxxx	0	
2	xxxxxxxx	xxxxxxxx	0	
3	xxxxxxxx	xxxxxxxx	0	
4	xxxxxxx0	xxxxxxx0	0	4
5	xxxxxxx0	xxxxxxx1	1	
6	xxxxxxx1	xxxxxxx1	0	
7	xxxxxxx1	xxxxxxx1	0	
8	xxxxxxx1	xxxxxxx1	0	
9	xxxxxxx1	xxxxxxx1	0	5
10	xxxxxxx2	xxxxxxx2	1	1

3.5. Concluding Remarks

This dissertation proposed the remote detection using IP timestamps by observing the characteristic patterns of the differences between 2 successive timestamps in the replied packets and how many times identical timestamps were stamped between the packets. The proposed analysis method and the experiment background were discussed in this chapter.

The anticipation of the experiment results for IP and ICMP timestamps was made based on finding that differences between IP and ICMP timestamps reply were not reproducible in machine with 2.40 GHz clock speed that was used for experiments in this dissertation, contrary from finding in [17] that used machine with 1.8 GHz clock speed.

Also as presented in the next chapters, based on the data from the experiments, the new approaches that are proposed in this dissertation, by analyzing the differences of successive timestamps and how many times same timestamps were stamped as IP timestamps characteristic patterns between VM and real machine, shows the potential to be used as a remote detection method to detect VM operating environment.

CHAPTER 4

Vulnerability Analysis using Network Timestamps in Full Virtualization

This chapter presents a remote detection technique using IP timestamp option that could be used to detect full virtualization VM environment and contributing to the VM vulnerability. IP timestamp option allows a requester to request timestamps value from any machine that handles packets by specifying the machine's IP address. This chapter discusses the characteristic patterns for IP timestamps received from the full virtualization VM environment and the comparison with IP timestamps received from real machine.

4.1. Overview

This test case served as the basic study to determine if network timestamp could potentially still be used by attackers to remotely distinguish operating environments such as VM and real machine. Related work by [17] had proved that IP and ICMP timestamp combinations could be used to remotely detect VM environment by using the characteristic pattern of timing discrepancies between IP and ICMP timestamps. Furthermore, 1/3 from hosts over the internet is set up to be able to process IP and ICMP request [88]. Thus, assumption that there are numbers of host that able to be remote scanned using IP and ICMP timestamp data were made.

In [17], experiments were done to detect discrepancies of IP and ICMP timestamps in two cases, which are when ICMP timestamp + 1 < IP timestamp and when ICMP timestamp < IP timestamp. The experimental results in [17] showed that discrepancies occurred only 0.12% from total 1,000,000 packets in the case when ICMP timestamp + 1 < IP timestamp for real environment. However, for VMs, discrepancies occurred in the range between 0.1% until 0.5% from total 1,000,000

packets. On the other hand, in the case when ICMP timestamp < IP timestamp, the study found out that 0% of discrepancies occurred from total 1,000,000 packets from real environment and the range of 0% until 2.5% of discrepancies occurrence for VMs.

Experiments in this research were done to validate the remote detection method by using characteristic pattern differences between IP and ICMP timestamps that were proposed in [17] in full virtualization VM environment. However, the experiments data in this study showed that the result in [17] was not reproducible when machine with higher performance was used. This research used machine with 2.40 GHz clock speed while in [17], machine with 1.8 GHz clock speed was used.

Since nowadays, machines with better CPU performance are increasingly being used, new approaches are proposed in this dissertation to remotely detect the operating environment. In experiments conducted for this chapter, both IP and ICMP timestamps replies were obtained and compiled. However, the ICMP information was obtained and compiled only for validation purposes of remote detection using the method that was discussed in [17]. As mentioned above, the results from the experiments done in this work showed that the IP and ICMP timestamps in the same replied packets have the same value, thus only IP timestamps data were used for further discussion in this dissertation. The experiments for this chapter aim to determine either operating environment could be detected using the following methods which are being proposed in this dissertation:

1) Differences between 2 successive timestamps in the replied packets
2) How many times identical timestamps was stamped between the packets

When the target hosts use VM, some of the timestamps will show discrepancies because VM sometimes interrupted timestamp operations to complete other operations. The VM clock is managed by timer device emulation, which called as VM switch and the differences in timestamps are bigger because VM operation will be switched with other VM in the queue. On the other hand, in real machine

environment, the timestamp difference could be very small since there is no interruption in the process.

This research is focusing on finding possible way in using timestamp data to detect the presence of VM. As the extension from the previous chapter, remote detection technique for full virtualization VM that uses IP timestamp option is discusses in this chapter. Experiments were conducted to obtain IP and ICMP timestamps reply data using method that was explained in Chapter 3 to prove that VM is detectable using timestamps discrepancies in full virtualization. However, from the experiments, the results showed that the IP and ICMP timestamps in the same replied packets have the same value, thus only IP timestamps data were used for further discussion in this dissertation.

The experiments environment was a private cloud computing environment that consists of full-virtualization VM and was set up in campus lab. Experiments were done in the lab network environment to minimize interruption to the IP timestamps data due to router clock skew. This research set up test environment with high performance machine as a host to run 30 VMs in order to provide maximum utilization for the machine in full virtualization technique. In the experiments, packets were sent from stand-alone machine as measurer client to target machine that host VMs and a real machine.

4.2. Experiments Environment

A custom script was executed in the measurer machine to send packets to target machine requesting for IP timestamps replies. These requests were repeated until 1,000,000 times. The IP packets data structure with IP timestamps request options and ICMP request is shown in Figure 6 in Chapter 3. The physical experimental environment that was set up for this experiment is shown in Figure 10. VM software from VMWare and VM vSphere 5 were emulated in the VM environment as full virtualization VM system and Linux Ubuntu 12.04 was emulated as the host OS for the VMs.

The specifications for the experiment environment are shown in Table 2, Table 3 and Table 4.

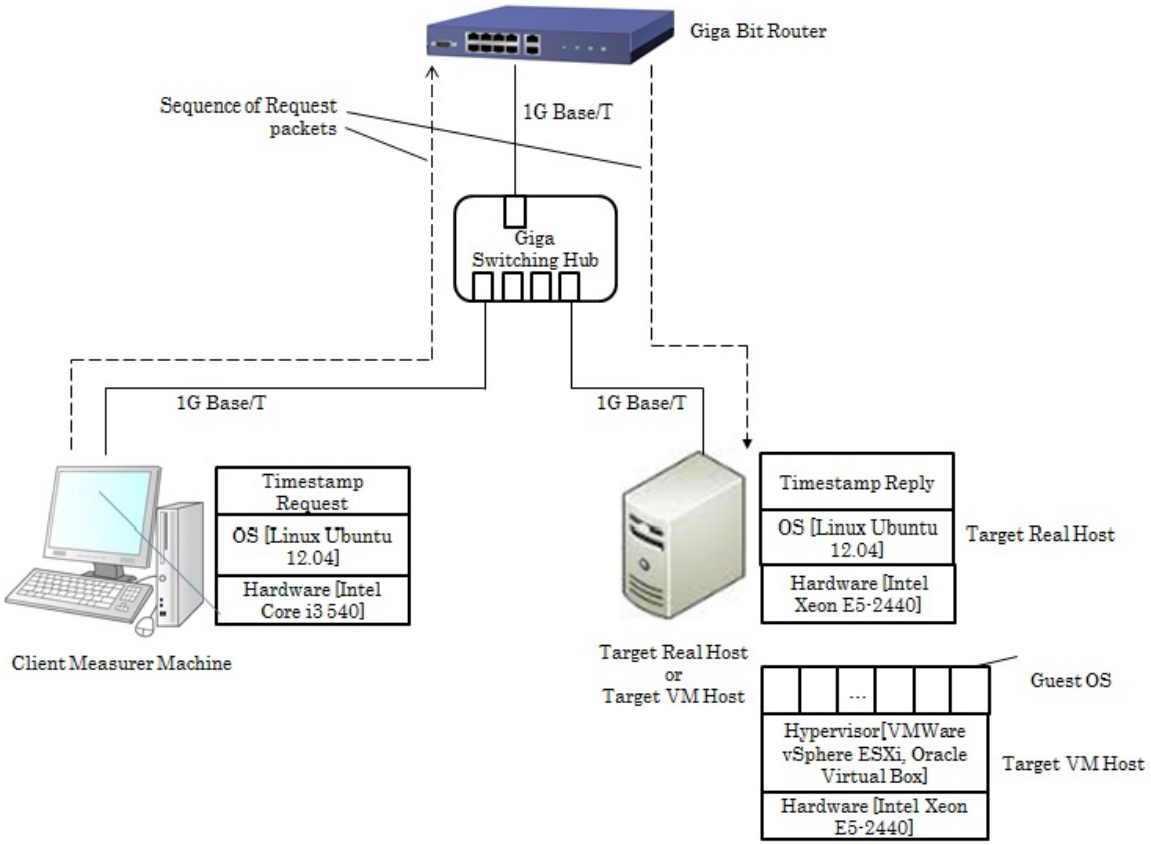


Figure 10: Experimental environment set up for real machine and 2 VMs

Table 2: Experiment environment for real OS

OS	Linux Ubuntu 12.04
CPU	Intel Xeon E5-2440
Memory	1GB
Hardware	Dell Power Edge

Table 3: Experiment environment for VMWare

OS	Linux Ubuntu 12.04
Hypervisor	VMWare vSphere Hypervisor (ESXi) 5.1.0
CPU	Intel Xeon E5-2440
Hardware	Dell Power Edge
Memory	(Virtual Allocation) 1GB
Storage	(Virtual IDE) HDD 16GB

Table 4: Experiment environment for VirtualBox

OS	Linux Ubuntu 12.04
Hypervisor	Oracle Virtual Box 4.3.12
CPU	Intel Xeon E5-2440
Hardware	Dell Power Edge
Memory	(Virtual Allocation) 1GB
Storage	(Virtual IDE) HDD 16GB

1,000,000 times of request packets were constantly sent to the target machines in order to measure the differences in timestamps reply characteristic received from real machine and VM. The differences were measured to the nearest microsecond. For the target machine in the experiments, high performance Intel(R) Xeon(R) Processor E5-2440 with 6 cores and 2.40GHz clock speed were used.

Timestamps replies that were received at the measurer machine were compiled to the nearest microsecond as data in a CSV file. In the case of VM, since there will be VM interface between the CPU and the network interface, it is expected that there will be a small delay in issuing the timestamps between the requests. On the other hand, it will not be an issue in real machine since it will only

have CPU and network interface interactions. Thus, as the results from analyzing and comparing the timestamp replies data between VM and real machine, it is expected that the timestamps value from VM will change more frequently compared to the timestamp replies from real machine.

4.3. Experimental Results

In this study, testing was performed using high performance machine with clock speed of 2.40 GHz and implementing full virtualization technique using VMWare ESX and VirtualBox. The timestamps reply consists of 32 bits but only could be display in 8 digits of timestamps in hexadecimal and 10 digits of timestamp in decimal number. The collected IP timestamps data were analyzed to determine its characteristic behavior. In the data analysis, this research defines *n* as count for how many times same timestamps were replied from the targeted machines. For real machine, 62% of the timestamps replies were 5 times of the same timestamps value and 25% of the timestamps replies were 4 times of the same timestamps value. However, for VM environments the behavior of timestamp stampings is different from the real machine where lesser same timestamps were sent as replies when the timestamps reply packets were sent to the requestor. The behavior pattern differences in these experiments are as shown in Table 5. Percentage of same timestamps that were stamped are shown in Table 6. Figure 11 shows the comparison of real environment and VM environments on how many time the same timestamps replies were observed. In real machine, more than 60% of timestamps are stamped for 5 times. In contrast, this research could see that there were no 5 times same timestamps replies in VirtualBox and VMWare ESX.

The reason is that VM sometimes interrupted timestamps operations to complete other operations and make the time taken to complete job longer than real machine. Even though in full virtualization that simplifies migration and portability, the remote detection by using IP timestamps packet reply still could be observed and this could reveal the environment that one system is running on. Malware will be able to detect the environment that they are running on and could choose not to

run at all or manipulate the running environments for the attack. Full virtualization offers the best isolation and security for virtual machines,

Table 5: Portion of collected IP timestamps information

Count	IP timestamp (millisecond)		
	Real Machine	VirtualBox	VMWare
n	xxx72359	xxx20763	xxx68337
n+1	xxx72359	xxx20766	xxx68338
n+2	xxx72359	xxx20767	xxx68339
n+3	xxx72360	xxx20768	xxx68339
n+4	xxx72360	xxx20769	xxx68339
n+5	xxx72360	xxx20769	xxx68340
n+6	xxx72360	xxx20770	xxx68340
n+7	xxx72361	xxx20771	xxx68341
n+8	xxx72361	xxx20772	xxx68342
n+9	xxx72362	xxx20773	xxx68343
n+10	xxx72362	xxx20775	xxx68344
n+11	xxx72362	xxx20775	xxx68344
n+12	xxx72363	xxx20776	xxx68344
n+13	xxx72363	xxx20776	xxx68345
n+14	xxx72363	xxx20777	xxx68346
n+15	xxx72363	xxx20778	xxx68346
n+16	xxx72363	xxx20779	xxx68347
n+17	xxx72364	xxx20780	xxx68347
n+18	xxx72364	xxx20781	xxx68348
n+19	xxx72364	xxx20782	xxx68349

Table 6: Percentage of same timestamps that were stamped

Count N	Real Machine (%)	VirtualBox (%)	VMWare (%)
7	2	0	0
6	3	0	0
5	62	0	0
4	25	1	36
3	5	19	55
2	2	47	7
1	0	33	2

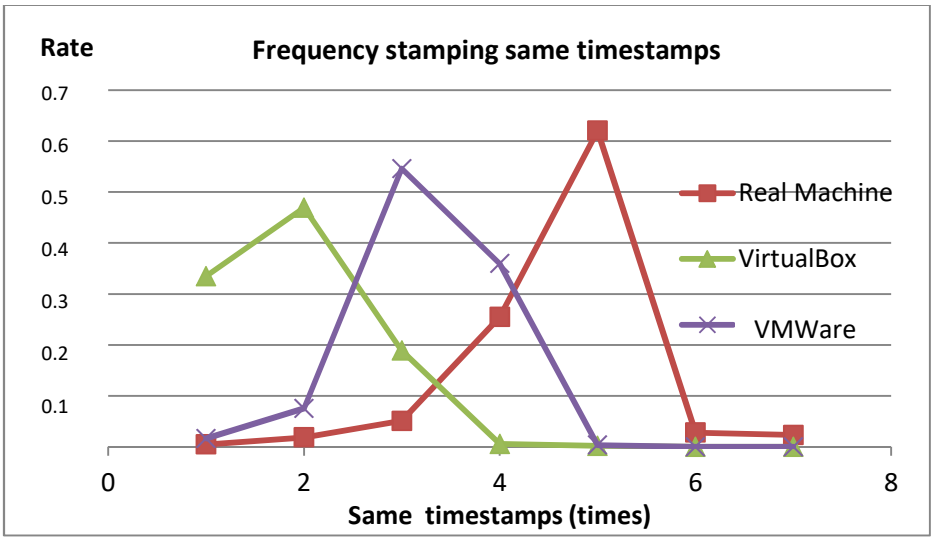


Figure 11: Count for same timestamps reply for Real Machine, VirtualBox and VMWare

however, remote detection by using network timestamps need to be addressed to resolve the VM vulnerability. Since that machine that was used to install the VM is continuously improved, the difference according to performance might be different.

More tests need to be done constantly and periodically to verify the current situation and detect vulnerability.

4.4. Concluding Remarks

Building a transparent VM is still a difficult task, as shown from the results of this research where remote detection method was discovered even for full virtualization VM technique. How VM could be detected has significant value that requires extensive studies and research in order to prevent any security loop holes that could exploit the vulnerability of VM including security holes that caused by any possible detection method. By analyzing all possible detection methods, which will be the ideal expectation, countermeasures could be proposed and implemented for creating better secured VM in cloud computing environment.

This study explores detection methods by looking at it from the perspective of detecting full virtualization VM existence by performing timestamps analysis using IP timestamp option, in order for that vulnerability in detecting VM existence from timestamps data could be addressed. This research shows that behavior between IP timestamps packets reply from the most popular technique of choice, VMWare ESX and VirtualBox could be differentiated remotely and this could be potentially used as a VM detection method. As future work, researches could be done to perform more tests using various machines and also in grid and cloud test bed to get more data characteristics for comprehensive analysis.

Based on the discussion in this chapter, it shows that the characteristic of timestamps patterns is different for VM and real machine. In the next chapter, based on the finding in this chapter, a mechanism will be proposed to change timestamps replies in real machine in order to make it look similar with VM's timestamps characteristic pattern. This mechanism should be able to hide the differences between timestamps replies from VM and non-VM and prevent it from being used as method to detect either that the machine is running on VM environment or not. Based on this research, future research about malware behavior and its relationship with VM remote detection for virtualizing environments could be studied. Thus, it is the goal of this dissertation to open up

various scenarios in virtualization implementation and the consideration to obtain various characteristic of data for future researches in this field.

The results will contribute on providing more secure cloud computing services. The approach that was proposed will contribute to the prevention of VM environment detection based on remote timestamps replies. This approach will decrease the possible detection method that could lead to manipulation of cloud computing environment and the possible attack vector. This finding also could be applied to cloud computing security policy framework for cloud computing.

CHAPTER 5

Proposed Countermeasure for Virtual Machines Detection Methods Using IP Timestamps Pattern Characteristic

This chapter presents a method for detecting VM environments by remotely detect a VM without installing any program on the target machine. It works by analyzing the pattern of IP timestamps in replies sent from the target machine to determine whether the target machine is a real machine or a VM. This chapter gives a brief overview of proposed counter measure to resolve the operating environment detection method using IP timestamps pattern characteristic.

5.1. Overview

Remote VM detection method works by exploiting the IP timestamp information in the reply packets from the VM. By analyzing the pattern of the IP timestamps, it can reveal the presence of a VM. The IP timestamp is an optional extension to the IP header that allows the sender to request timestamp values from any machine that handles the packet by specifying its IP address [89]. The IP timestamp option has three modes available:

- Collect timestamps from each device; space in the header is available for up to nine devices.
- Collect the IP address and up to four timestamps from each device
- Specify in advance up to four IP addresses from which a timestamp is requested.

Here, the third mode was chosen to be implemented, because the target host IP address in the controlled environment of the experiments could be set up in advanced. Another reason for selecting this option is that it is conveniently available in the Linux ping command that used in the program that used in this experiment.

In [17], it was observed that the IP and ICMP timestamp patterns could be differentiated by analyzing the IP and ICMP timestamps in 1,000,000 packets sent back as replies from the target VM. Thus, it is concluded that 1,000,000 IP timestamps from the target machines would be sufficient for observing the differences in IP timestamps patterns between the target machines. In this chapter, experiments were conducted to obtained IP and ICMP timestamps data from 3 major hypervisors which are VMWare, VirtualBox and Xen to be used for implementation of proposed modification. In the experiments conducted, both IP and ICMP timestamps replies were obtained and compiled. However, the ICMP information was obtained and compiled only for validation purposes of remote detection using the method that was discussed in [17]. From the experiments, the results showed that the IP and ICMP timestamps in the same replied packets have the same value, thus only IP timestamps data were used for further discussion in this dissertation.

5.2. Proposed Remote Detection method and Countermeasure

In client test, 1,000,000 consecutive non-suspicious packets were sent that had the IP timestamp option enabled to the target machines. To avoid the request queue from being flooded, request packets was sent after the previous packet reply was received. The reply packets from the target machines included the IP timestamp information, i.e., the time in the target machine when the reply packet was generated. The data structure of the sent IP packet is shown in Figure 6 in Chapter 3.

The reply packets from the target clients and their IP timestamps were analyzed to determine if there was any distinguishable pattern characteristic between the target machines. In the same way as shown in [17], [90], IP timestamp patterns of the VMs from the real machine could be distinguished.

Distribution graphs were plotted on the basis of the timestamp pattern differences identified in the experiment. Then, a solution devised using the delay modification technique, wherein real machines are made to show the same IP timestamp reply pattern as the VM, thereby eliminating any chance differentiating between VMs and real machines within the network by using IP timestamp

patterns. This countermeasure should be implemented in the real machine rather than the VM mainly because VMs work by time-sharing host physical hardware and it is impossible for a VM to duplicate the timing activity of a physical machine even if it uses several techniques to minimize and conceal differences in timing performance [91].

The modification was implemented in the time-stamping process in a real machine to create delays in the timestamps in the reply packets. When the packets with the IP timestamp option arrive at the real machine, they are delayed using the countermeasure before being forwarded to OS for processing. The delay is implemented by adjusting the mean number of repetitions of the same IP timestamp in the real machine to match those in the VMs. The following notation will be used to describe our method to calculate the mean number of repetitions of the same IP timestamp of the real machine and the VMs.

r_i: number of identical timestamps in the i-th run of a real machine
(= i-th run length of identical timestamps)

r_j: number of identical timestamps in the j-th run of a virtual machine
(= j-th run length of identical timestamps)

n_r, n_v : number of runs of repetitions of the same timestamp for a real machine and virtual machine, respectively

R_r, R_v : mean run length for a real machine and virtual machine, respectively

t_r, t_v : mean interval time between successive packets from a real machine and virtual machine, respectively.

The mean run lengths, R_r and R_v, are calculated formulas follows.

$$R_r = \frac{\sum_{i=1}^{n_r} r_i}{n_r}, \quad R_v = \frac{\sum_{j=1}^{n_v} r_j}{n_v}.$$

Then, the mean interval times between packets with consecutive identical timestamps are $t_r = \frac{1}{R_r}$ (millisecond) and $t_v = \frac{1}{R_v}$ (millisecond), and the delay time is $\Delta t = t_v - t_r$. By delaying all the packet replies for Δt, the peak position in the graph of the modified run length of identical timestamp packets for a real machine coincides with the peak position for each virtual machine.

However, the rate of the modified mean run lengths will concentrate around the peak position and the rate in the neighborhood of the peak position will be too low because this research set the delay to be constant. To avoid such a concentration, the delay times around 15 ~ 25% of randomly selected identical timestamps in reply packets are modified to $0 \sim 0.5 \cdot \Delta t$ and $2 \cdot \Delta t \sim 4 \cdot \Delta t$. Prediction that the graph of the modified mean run a length of identical timestamp packets from the real machine would almost coincide with those of the VMs was made. The proposed modification is shown in Figure 12.

The experiments showed that by implementing this technique, IP timestamps differences for a real machine and VM could be eliminated. That means the VM and real machines could no longer be distinguished by remotely detecting their IP timestamps.

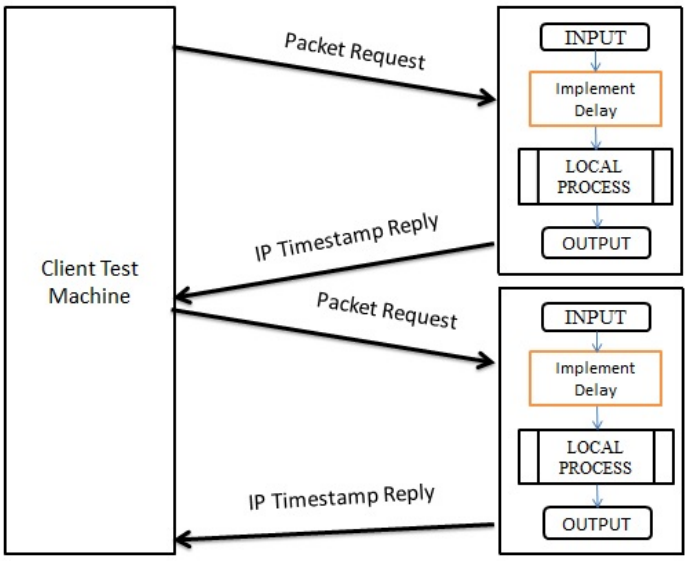


Figure 12: Proposed modification technique

5.3. Experimental Design and Measurement Methodology

The experiment was conducted on a high-performance Dell Power Edge server equipped with an Intel Xeon CPU E5-2440. XenServer 6.5 SP1, Oracle VirtualBox 4.3.12, and VMWare vSphere Hypervisor (ESXi) 5.1.0, all major commercial hypervisor products, were implemented to host the VM test machines. Open source Linux Ubuntu 12.04 with 1GB of virtual memory and IDE HDD with 16GB of virtual storage was used as the guest OS in the hypervisors and as the VM measurement targets. A machine running with Ubuntu Linux as the OS and Intel Core i3 540 as the CPU with 2.8GB of RAM was used as the real machine measurement target. A measurer machine was set up to send request packets with the IP timestamp option to the measurement target machines. The measurement target machines and the measurer machine were connected to each other via Ethernet. The experimental environment is shown in Figure 7 in Chapter 3. The physical experimental environment to obtain IP timestamps characteristic pattern are shown in Figure 13. Data that was obtained in this experiments were used to reflect the proposed modification technique to validate if the proposed modification technique is applicable.

Each VM consisted of the minimum software base of GNU Linux running on 12.04. The utilization of each VM test environment was set up to 80% and more. The experiments were conducted by sending request packets with the IP timestamp option enabled in the header after the previous timestamp reply was received by the measurer machine. Although IP timestamps can be used with any type of IP packet, this research only explored the attachment of the IP option and sent it with ICMP packets due to their convenient availability in the Linux ping command. We sent non-suspicious packets to the target host machine in order to make sure they would not be dropped or denied by the network or devices.

As many as 1,000,000 packets were continuously sent from the test client machine to each target machine. The packets were sent from the test client machine by executing a customized script developed for this experiment. The next request packet of the client test machine was only sent to the target host once the measurer machine had received the reply for the previous packet request. The timestamp

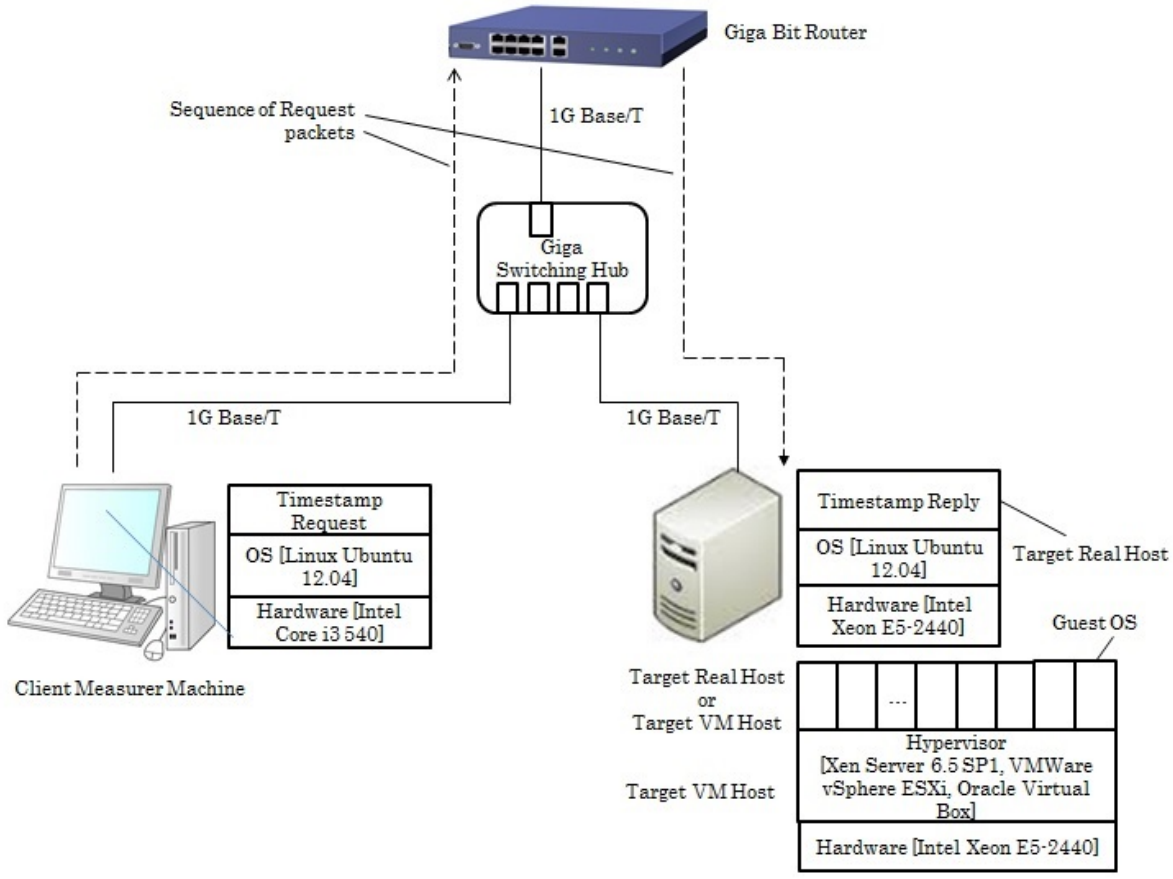


Figure 13: Experimental environment set up for real machine and 3 VMs

information in the reply packets from the target machine was recorded and compiled. The IP timestamps were analyzed in decimal units to the nearest millisecond. Milliseconds was chosen as the unit for analysis as it is the standard unit for the timestamp in the IP packet [80]. Each target operating system added its timestamp, and the timestamps in the packets were not affected by the network until they reached the client machine. Thus, accurate timestamps were obtained from the target guests.

Finally, the IP timestamp information from the measurement target machines was analyzed and distribution graphs were plotted in order to observe the differences between the timestamps of the target guests. After that, this research implemented timestamp modification technique that proposed in this dissertation based on the distribution graphs and tested it to confirm its usability. A study by

Kohno has proven that the clock skew is independent of the access topology, regardless of whether the hosts use random or constant IP addresses [16].

Therefore, for these experiments, this research used a controlled environment with an Ethernet connection in our laboratory to eliminate the network latency issue. Note that the characteristics of the data might vary from machine to machine, from one hypervisor technology to another, and with changes in the implementation environment. Hence, this research plan to conduct more tests in different experiment environments to get more data with which can improve the implementation of the modification.

5.4. Data Analysis

The research presented in this paper are an extension to work in [90], [92] for testing and analyzing the network timestamps discrepancies of major VM products and thus strengthens contention that timestamps could be used for remotely detecting VM environments. In this research, non-suspicious packets were sent to the target host machines in order to make sure the packets would not be dropped or denied by the network or devices. This research then analyzed the reply packets to determine whether there were any behavior differences between VMs and real machines. The results showing that there are indeed such differences supported our hypothesis that VM environments could be remotely detected by analyzing the timestamp reply patterns.

This research analyzed the collected data to understand the differences between the time-stamping behaviors of the target hosts. The timestamps were extracted from the reply packets from each target host in the test client machines. Table 7 shows a portion of the IP timestamp data that was collected.

The count information means the sequence of packets sent to the target machines. The shown sequence runs from the first packet sent, n, until the 20th packet, n+19. In particular, for the real machine, the timestamps in the first three reply packets were the same, and those in the next five packets timestamp were also the same. Figure 14 shows how many times the same timestamps were stamped in the sequence of reply packets from the target hosts.

The percentage of identical timestamps from the real machine is obviously higher than those from the VM. This behavior occurred throughout the collected IP timestamp data. Table 8 shows the results of the analysis of all 1,000,000 reply packets from the real machine and VMs.

Table 7: Portion of collected IP timestamps information

Count	IP Timestamp (millisecond)			
	Real Machine	Xen	VirtualBox	VMWare
n	xxx81920	xxx38539	xxx70653	xxx77867
n+1	xxx81920	xxx38541	xxx70656	xxx77868
n+2	xxx81920	xxx38543	xxx70657	xxx77869
n+3	xxx81921	xxx38546	xxx70658	xxx77869
n+4	xxx81921	xxx38548	xxx70659	xxx77869
n+5	xxx81921	xxx38552	xxx70659	xxx77870
n+6	xxx81921	xxx38553	xxx70660	xxx77870
n+7	xxx81921	xxx38645	xxx70660	xxx77870
n+8	xxx81922	xxx38647	xxx70660	xxx77871
n+9	xxx81922	xxx38650	xxx70661	xxx77871
n+10	xxx81922	xxx38652	xxx70662	xxx77871
n+11	xxx81922	xxx38654	xxx70662	xxx77872
n+12	xxx81923	xxx38656	xxx70663	xxx77872
n+13	xxx81923	xxx38659	xxx70665	xxx77872
n+14	xxx81923	xxx38660	xxx70666	xxx77873
n+15	xxx81923	xxx38663	xxx70666	xxx77873
n+16	xxx81923	xxx38665	xxx70668	xxx77873
n+17	xxx81924	xxx38667	xxx70668	xxx77873
n+18	xxx81924	xxx38669	xxx70669	xxx77874
n+19	xxx81924	xxx38672	xxx70669	xxx77874

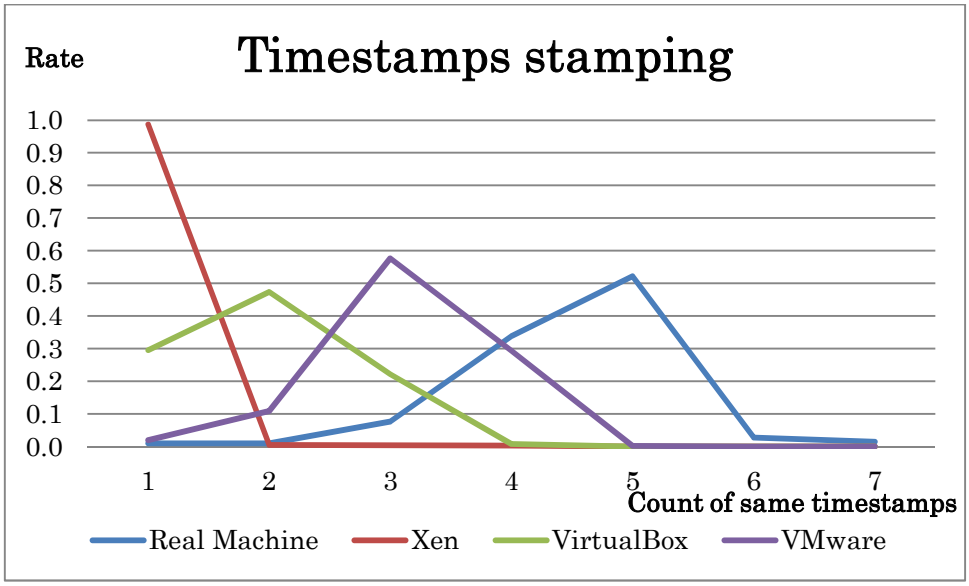


Figure 14: Analysis of IP timestamp behavior pattern of target machines in real machine, Xen, VirtualBox and VMWare

Table 8: Percentages of identical IP timestamps for real machine and VM

How many times the same IP time-stamp was stamped	Real Machine (%)	Xen (%)	VirtualBox (%)	VMWare (%)
1	0.99	98.71	29.52	1.95
2	0.99	0.48	47.37	10.87
3	7.64	0.42	22.21	57.64
4	33.90	0.34	0.80	29.35
5	52.20	0.05	0.04	0.19
6	2.79	0	0.06	0
7	1.49	0	0	0

Table 9: Mean number of packets with the same timestamps

Host	Real Machine	Xen	VirtualBox	VMWare
Mean number of repetitions of the same timestamp in the reply packets	4.50	1.03	1.95	3.15

Mean number of repetitions of the same IP timestamps in the reply packets from the real machine and the VMs was calculated. As shown in Table 9, the mean values reveal significant differences in the IP timestamp patterns.

The real machine had a mean of almost 4.50 repetitions of the same timestamp, while the VM target hosts had smaller mean repetition values.

Accordingly, it could be determined that VMs and real machine had significant differences in their IP timestamp behaviors. These results support this dissertation hypothesis that since VM packets are handled via hypervisor, timestamp delays will occur in the reply packets from the VM environment. Furthermore, as expected, different hypervisor technologies had different timestamp behavior patterns.

Thus, VMs could be remotely detected by using the IP timestamp pattern behavior, and each VM technology had its own mean values and pattern of timestamps. Findings in this research are limited to the testing environments that were established in a controlled environment. More characteristic of pattern will be obtained in the future by performing more testing on various machines and implementation technologies that could be done as future work. The findings will be recorded and shared for references. Comprehensive framework will be proposed when enough data could be gathered for more comprehensive analysis.

5.5. Modification of Delay in Real Machine Environment

Numerical experiments were performed to determine the effectiveness of proposed countermeasure of imposing delays in the real machine in order to camouflage it within the IP timestamp behavior patterns of the VMs. The experimental results, shown in Figure 15 (a), (b) and (c), indicated that the real

machine could emulate the IP timestamp behaviors of the VMs. Therefore, such a such a modification could possibly be used as a countermeasure to prevent attackers or malware from remotely detecting VMs by exploiting the difference in IP timestamp behavior.

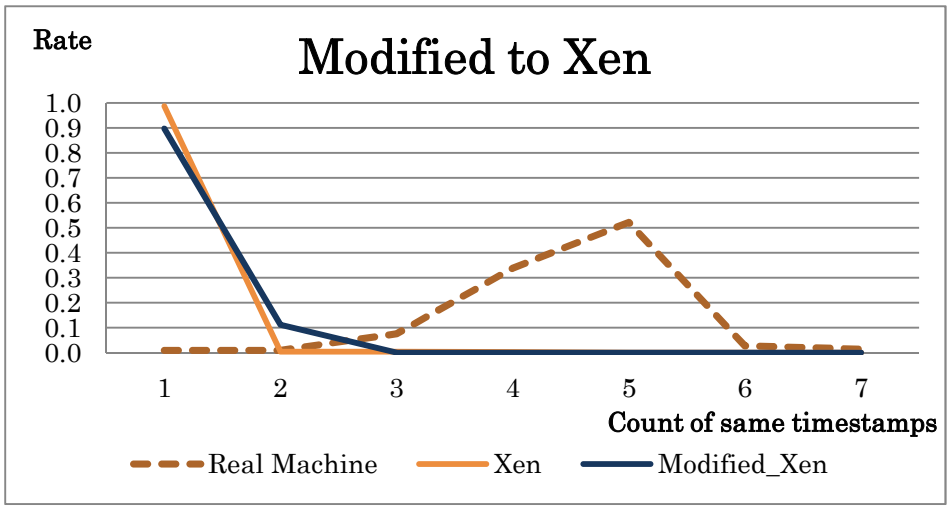


Figure 15 (a). Technique to match IP timestamp behavior of Xen

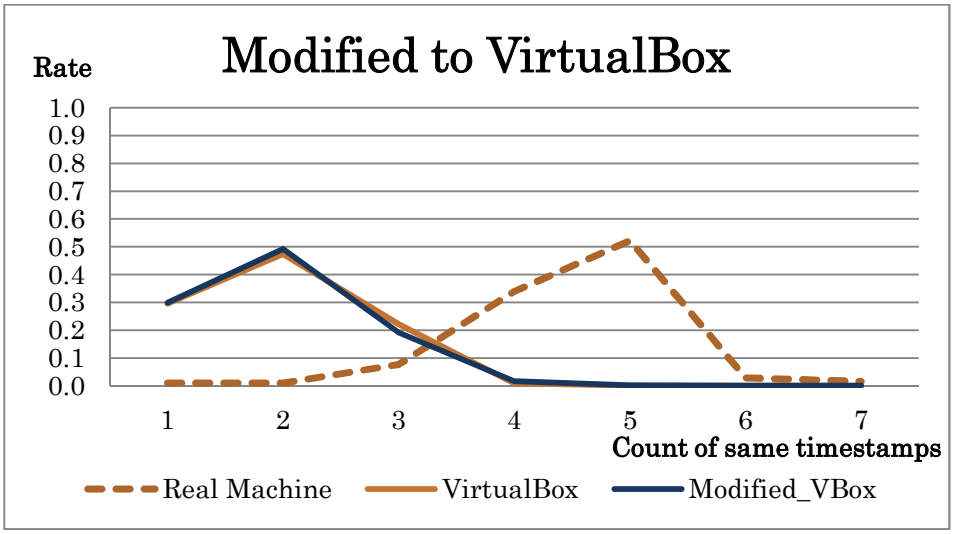


Figure 15 (b). Technique to match IP timestamp behavior of VirtualBox

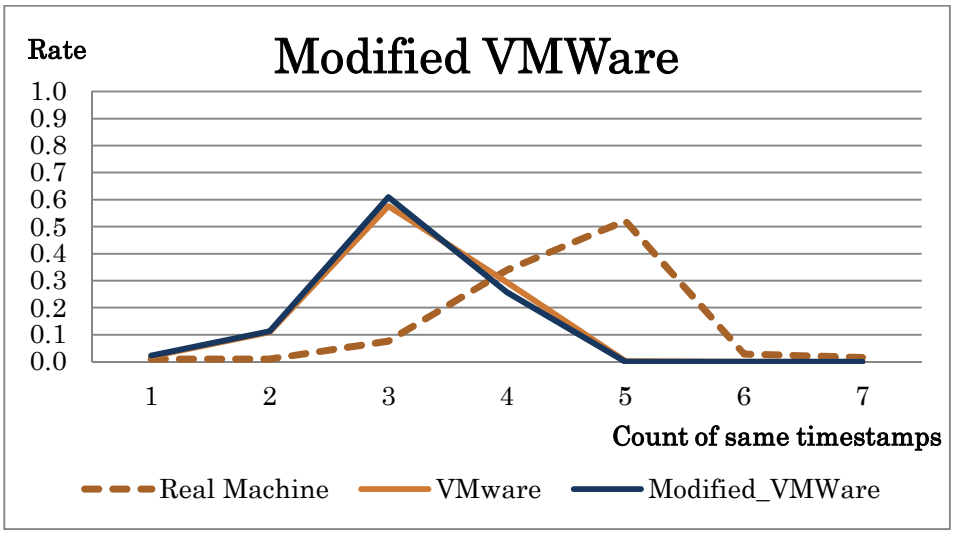


Figure 15 (c). Technique to match IP timestamp behavior of VMWare

5.6. Considerations

In this research, way to determine whether the running environment is a VM or real one was devised by exploiting the differences between the patterns of network timestamps and used it to detect the presence of VMs in a cloud computing environment. In this proposed method, packets that included IP timestamp option requests were sent from the client test machine to target VM hosts. Packets reply were collected packets from the target hosts and analyzed as to their behavior. Pattern of reply of timestamp information from a VM was predicted would be different from that of a real machine. The reason is that VMs sometimes interrupt timestamp operations to complete other operations. The time differences in the timestamps of the reply packets would thus be larger than in a real machine because a VM sometimes stops operation so that other VMs in the queue can operate and the VM clocks are managed with a timer device emulation called a VM switch. In the experiment, the collected IP timestamp data were in milliseconds as used in the IP timestamp specification.

As per this research hypothesis, the real machine added timestamps more frequently than the VMs did. The real machine can be camouflaged by making its IP timestamp reply pattern indistinguishable from those of the VMs. The research

presented here is an extension of research in [92],[90] to testing and analyzing major VM products and thus strengthens our conclusions in those papers. Countermeasure also was proposed wherein the timestamps in the real machine are modified to match those of the VMs. This countermeasure ensures that the VMs and real machines in the same cloud can no longer be distinguished by exploiting the IP timestamp remote detection.

5.7. Concluding Remarks

Building a transparent VM is still a difficult task; the question of how VMs can be detected is a critical one that requires extensive s research in order to prevent any security loop holes that could exploit the vulnerabilities of VMs, including security holes caused by any possible detection method. VM detection is a prelude to an attack if no countermeasure can be found. Ideally, by analyzing all possible detection methods, countermeasures could be devised for making VMs more secure.

This study explored the possibility of such a detection method from the perspective of detecting the VM's existence by performing timestamp analysis in full and para-virtualization using the IP timestamp option. This research shows that the behavior of IP timestamps in the reply packets from the most popular technique of choice, i.e., VMWare ESX, VirtualBox, and Xen, could be differentiated remotely and could be potentially used as a VM detection method. Our findings that timestamps in reply packs from VMs are different from those of real machines then were made the basis for a delay mechanism to make timestamps in the real machine look similar those of the VMs.

This mechanism should make it impossible to detect whether or not the machine is a VM. In this research, VMs could be detected remotely even when they were running on a high-performance server with an Intel processor in a high-performance cloud computing environment was proved. Countermeasure in which a delay is imposed in the real machine in order to camouflage the IP timestamp behavior patterns also was proposed.

Mobile devices such as smartphones will have other characteristic pattern and delay also should be implemented in their timestamp stamping operations to camouflage the existence of a mobile devices environment as well. Proposal of countermeasure that are made in this dissertation will decrease the chances of detection that could lead to malicious manipulation of VMs in cloud computing environment and vice versa and also eliminates a possible attack vector.

CHAPTER 6

Characteristic Patterns of Timestamps from from Android Operating System on Mobile Device and Virtual Machine

This chapter gives a brief overview of characteristic patterns of timestamps from Android operating system on mobile device and comparison with VM.

6.1. Overview

In this chapter, the analysis of characteristic patterns of IP and ICMP timestamps from Android OS running on mobile device and VM environment are presented. In the experiments conducted, both IP and ICMP timestamps replies were obtained and compiled. However, the ICMP information was obtained and compiled only for validation purposes of remote detection using method that was discussed in [17]. From the experiments, the results showed that the IP and ICMP timestamps in the same replied packets have the same value, thus only IP timestamps data were used for further discussion in this dissertation. From the findings, mobile device and VM environment proven could be distinguished by examining characteristic patterns of IP timestamps characteristic patterns using methods that were proposed in this research. Such characteristic could be exploited by malware in hiding its malicious programs upon detecting the VM environment.

In [92], timestamps behavior for VM hypervisors was checked. In the study, IP timestamp information received from VMs proven to exhibit different behaviors compared the IP timestamps from a real machine (PC). In [90], IP timestamp patterns for the full virtualization hypervisor was proved to show distinguishable differences between real machines (PC) and VM. On the other hand, in mobile devices case, timestamp discrepancy could occur due to limited resources such as processing power in the devices. As the result, bigger different between timestamp

t_b and timestamp t_a could be observed. The comparison of characteristic patterns of IP and ICMP timestamps for Android OS running on mobile device and VMs are not addressed comprehensively yet.

Thus in this research, tests were conducted to verify the characteristic patterns of timestamps from Android OS on mobile device and VMs. Since VMs are normally operated in high performance machines and mobile devices such as smartphone are constrained by their limited resources, prediction had been made that differences of characteristic patterns of timestamps could be observed clearly. Therefore, by using the characteristic patterns, Android OS on mobile device VM could be differentiated. Mobile platforms grow increasingly in popularity that contributes to bigger incentives for attackers. As a result, mobile attack become rapidly increasing in number and more sophisticated. In this experiment, information about characteristic pattern of remote detection using IP and ICMP timestamps as per method that was explained in chapter 3, 4 and 5 were gathered.

In the years where mobile devices are widely being used, malware creator would want to detect if they are running in mobile environment that still have minimum security measure implemented to make sure their existence will not be revealed and they can maximize their benefit from their malicious activity in mobile device. With the expand usage of mobile phone, the remote detection method of operating environment could be one great potential of vulnerability in cyber security attack.

Thus, this study could be considered as study that addresses the security issue for mobile device that using Android as OS that are most being used. This study could provide a basic study that could contribute in building the integrated security solution for smart device which also consider passive remote detection method of android operating environment that could enable the malware to not reveal their malicious behavior whenever they are not in android operating environment to skip the security test such as malware scanning procedure that majority being implemented in VM environment.

6.2. Experimental Environment

In the experiments, packets that request both IP and ICMP timestamps from measurer machine was sent to the target machines which includes Android OS operated as emulator in mobile device and VM running Android OS. The experiments environment to determine characteristic patterns of timestamps from Android operating system on mobile device and VMs is shown in Figure 16.

A measurer machine running with open source Linux Ubuntu 12.04 as the OS and Intel Core i3 540 as the CPU with 2.8GB of RAM was set up to send packets with the timestamp option to the measurement target machines. High-performance Dell Power Edge server with Intel Xeon CPU E5-2440 and 2.40 GHz clock speed was used to host the VM target machine. Major hypervisor products, [93], i.e., VMWare[94], Oracle VirtualBox[95] and Xen[96] were implemented in the

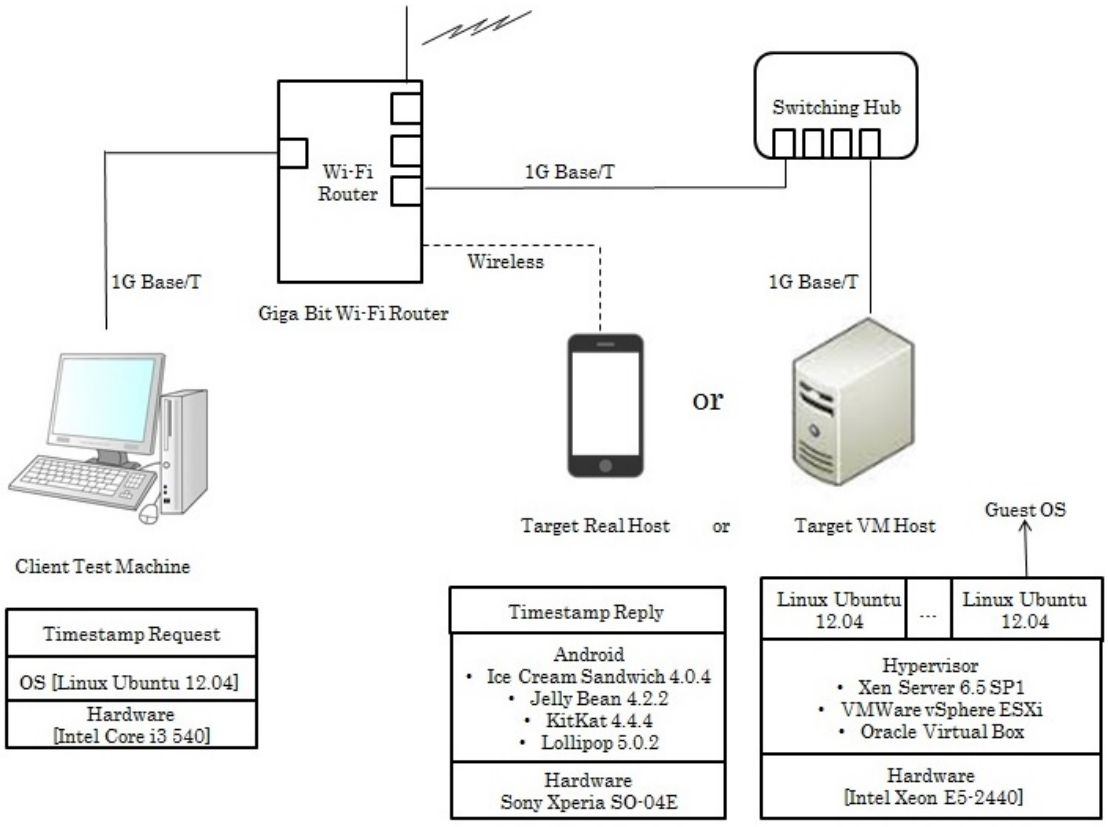
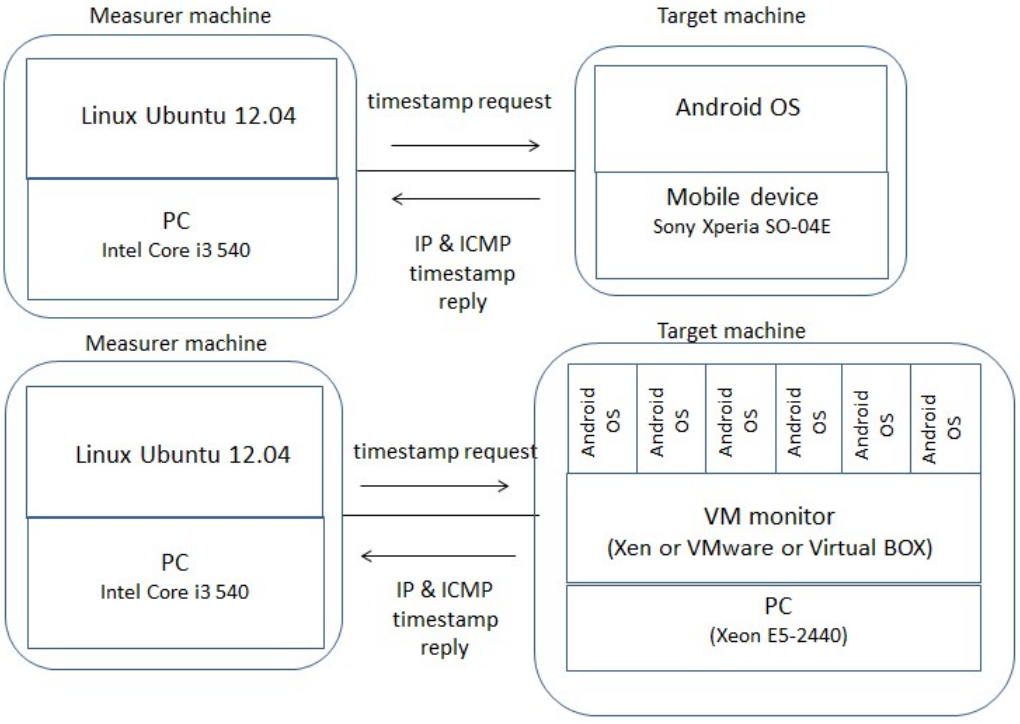


Figure 16: Experimental environment for Android OS on mobile device and Android as emulator in VMs

experiments as emulators environment for Android OS. Open source Android Lollipop 5.0.2 with 1GB of virtual memory and IDE HDD with 16GB of virtual storage was set up as the Android OS on the VMs and tests were done accordingly. As for the measurement target mobile device, Android was installed on Sony Xperia SO-04E and tests were done separately on 4 different Android versions which are Android Ice Cream Sandwich 4.0.4, Android Jelly Bean 4.2.2, Android KitKat 4.4.4, and Android Lollipop 5.0.2.

Timestamps request packets were sent from the measurer machine by executing customized script developed for this experiment. Figure 17 shown the mechanism of timestamps distribution collection. The measurer machine collects the timestamps information in the replied packets received from the measurement target machines. The target machines and the measurer machine were connected using Wi-Fi that set up in laboratory. C language scripts were written to send



Request timestamps Send timestamps response

Collect timestamps distribution

Figure 17: Timestamps distribution collection mechanism

packets to request for IP and ICMP packets reply with timestamp option from the client machine to the target machines. In this research, non-suspicious packets sent to the target machines in order to make sure the packets would not be dropped or denied by the network or devices. CPU busy ratio of each target machine was set up and maintained at 80% in order to emulate the normal usage of the machines.

As many as 1,000,000 packets were continuously sent from the measurer machine to each target machine by executing the developed C language scripts. The next packet from the measurer machine was only sent to the target machines once the measurer machine had received the reply for the previous packet. In the experiment environment, the timestamps in the packets from the target machines were not affected by the network until they reached the measurer machine. Thus, accurate timestamps were obtained from the target machines. The timestamp information in the reply packets from the target machines were recorded and compiled. The IP and ICMP timestamps from the compiled data were analyzed in decimal units to the nearest millisecond. Milliseconds was chosen as the unit for analysis as it is the standard unit for the timestamp in the IP packet [80]. Also, RFC 792 imposes a 1 milliseconds resolution to the ICMP timestamps and, since we use active requests for them, sufficient timestamps can be collected in a short amount of time, which makes the method feasible for fast identification.

The data were analyzed by examining the difference of timestamp between successive packets that were received from the target machines. This research also examines deviation of IP and ICMP timestamps in 1 packet. From the analyzed data, graphs of the timestamps difference in value, rate of the occurrence and IP and ICMP deviation were plotted to investigate the characteristic pattern differences of IP and ICMP timestamps from each target machine respectively.

6.3. Limitations

A study by Kohno had proven that the clock skew is independent of the access topology, regardless of whether the hosts use random or constant IP addresses [16]. Therefore, for our experiments, this research used a controlled

environment that was set up in our laboratory to eliminate the network latency issue. Note that the characteristics of the data might vary from device to device, from one VM technology to another, and with changes in the implementation environment. Latency issue was not addressed in this research. This research hypothesized that a VM environment could be detected by comparing the behavior patterns of IP and ICMP timestamps sent from VM target hosts and with the IP timestamps of mobile devices that are using Android as OS within the same environment.

6.4. Results Analysis

The collected data was analyzed to understand the time-stamping pattern behaviors of the target machines. Table 10 shows a sample of a portion of IP and ICMP timestamp data for the 15 count sequence, from n th to (n+14) th packet. 1,000,000 IP and ICMP timestamp data were collected from all the target machines. Based on the collected data, the differences of ICMP timestamps value between (n+1) th – n th were calculated for all the count sequence data. The differences of timestamps in the sequence were compiled to find the distribution of difference successive timestamps in order to find the characteristic of the timestamps reply from the target machines.

Distribution graphs were plotted in order to observe the differences between the timestamps of the target machines. Figure 18 (a), (b), (c), (d) are the distribution patterns of the difference value between the timestamps from the mobile device target machine on which 4 versions of Android OS are operated and the reoccurrence rate in the 1,000,000 ICMP timestamp data.

Based on the distribution graph, the peak of reoccurrence rate for timestamp difference for Android Ice Cream Sandwich 4.0.4 and Android Jelly Bean 4.2.2 is 2 and 3 milliseconds. While for Android KitKat 4.4.4, and Android Lollipop 5.0.2 the peak is 2, 3 and 4 milliseconds. From this results, This research could observe that pattern characteristic for 4 versions of Android in Wi-Fi environments are quite similar, where the peak of reoccurrence rate for the difference of timestamps value in the sequence are 2, 3 and 4 milliseconds.

Table 10: Portion of collected IP and ICMP timestamp information

IP and ICMP timestamp (millisecond)				
Count	IP Timestamps	ICMP Timestamps	Difference of successive ICMP timestamps	Different between IP and ICMP timestamps
n	25567551	25567551	Nil	Nil
n+1	25567556	25567556	5	0
n+2	25567560	25567560	4	0
n+3	25567566	25567566	6	0
n+4	25567571	25567571	5	0
n+5	25567575	25567575	4	0
n+6	25567579	25567579	4	0
n+7	25567584	25567584	5	0
n+8	25567592	25567592	8	0
n+9	25567595	25567595	3	0
n+10	25567599	25567599	4	0
n+11	25567602	25567602	3	0
n+12	25567605	25567606	3	1
n+13	25567618	25567618	13	0
n+14	25567626	25567626	8	0

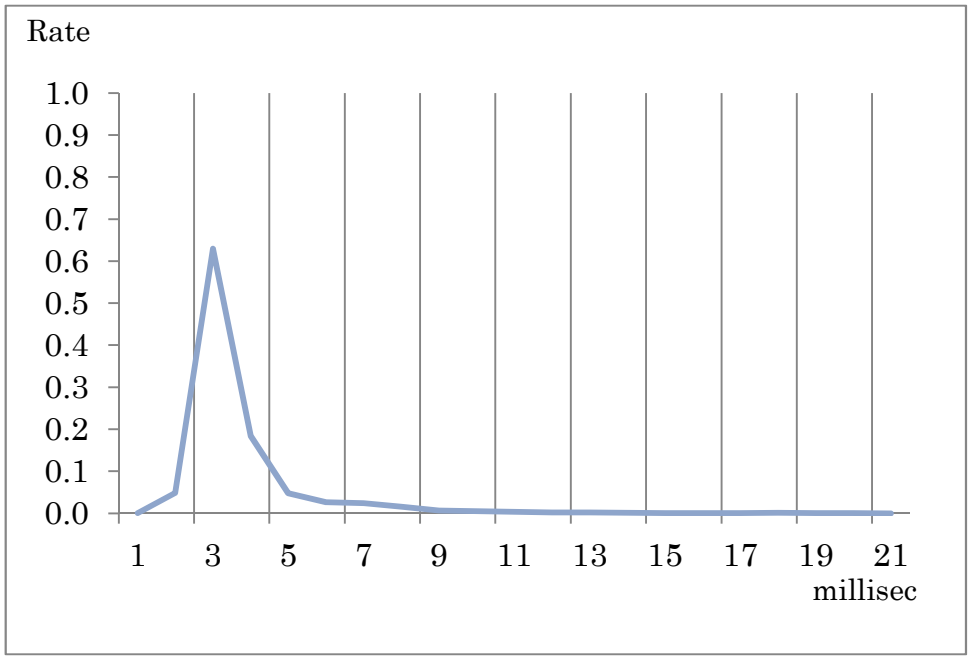


Figure 18 (a): Timestamp difference distribution of Android Ice Cream sandwich 4.0.4

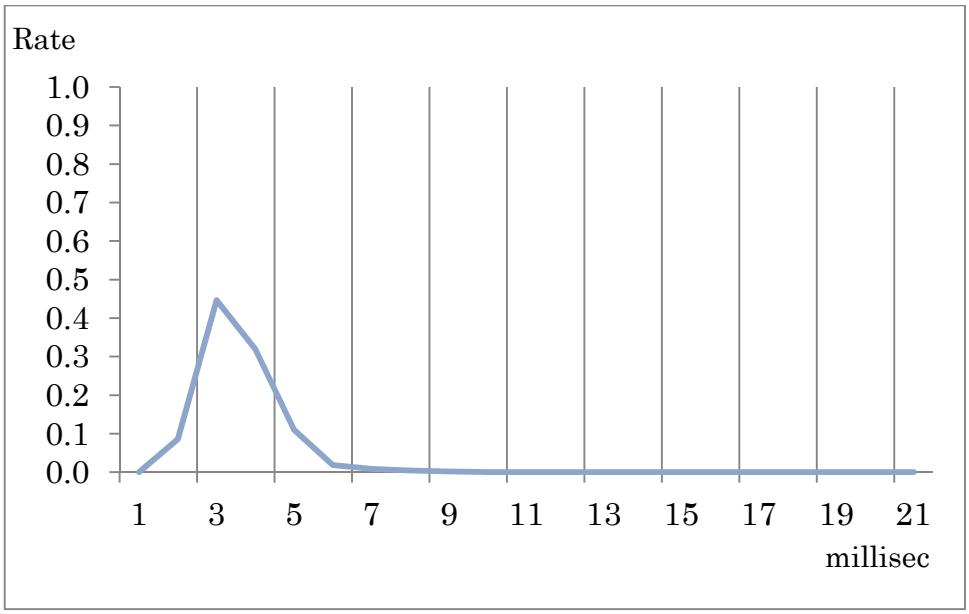


Figure 18 (b): Timestamp difference distribution of Android Jelly Bean 4.2.2

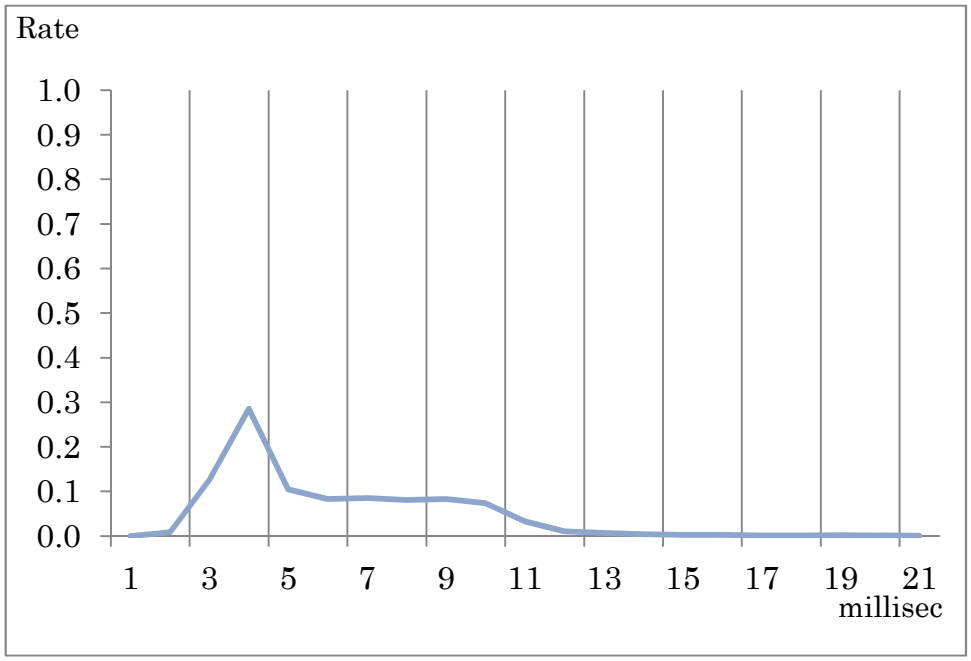


Figure 18 (c): Timestamp difference distribution of Android KitKat 4.4.4

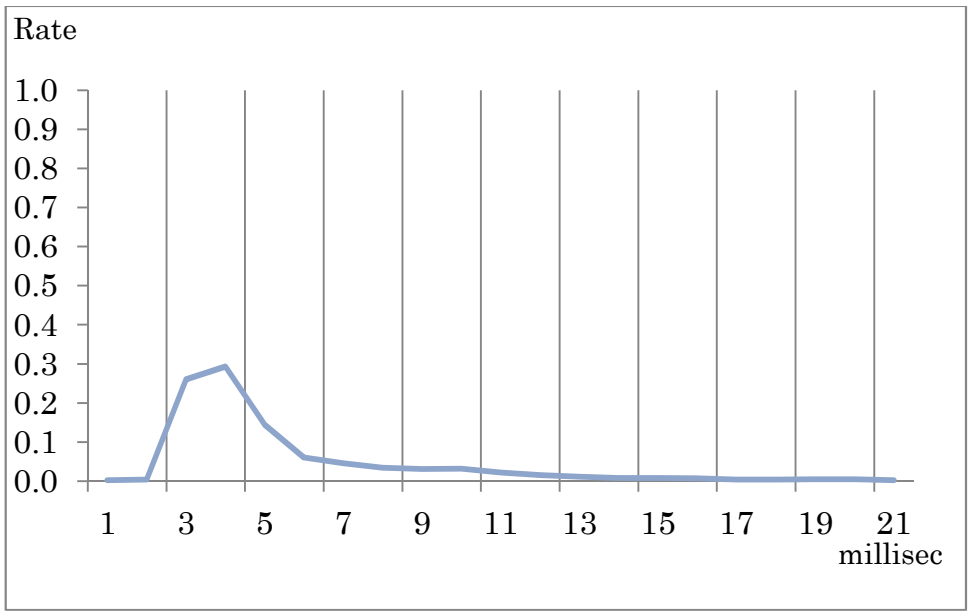


Figure 18 (d): Timestamp difference distribution of Android Lollipop 5.0.2

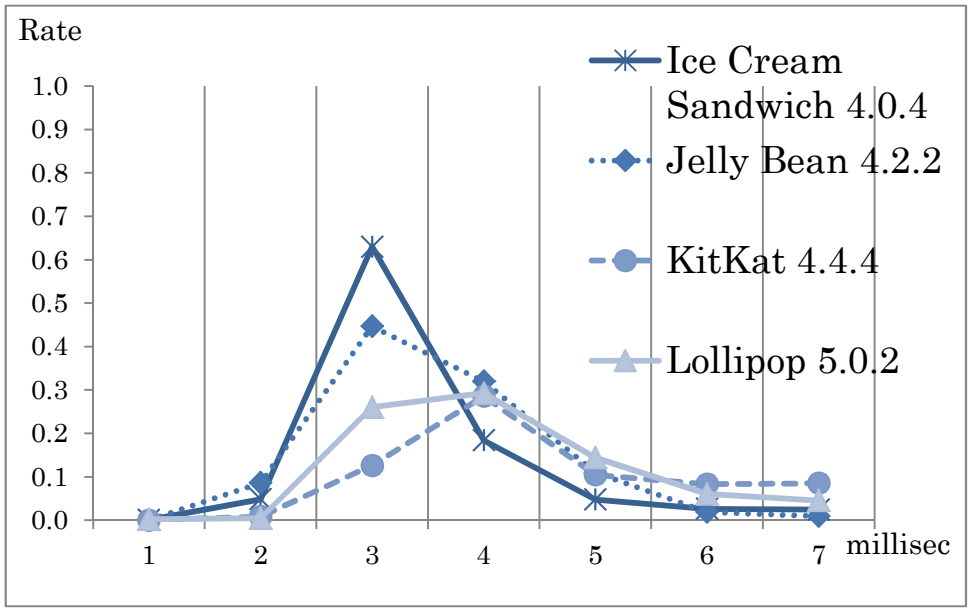


Figure 19: Timestamps difference distributions for 4 versions of Android

Figure 19 shows the compilation of distribution patterns for all 4 tested Android versions. The distribution patterns show that all Android versions that were used in the experiments were not stamping any same timestamp in 1 millisecond time frame. This is due to the mobile device constrain, which are limited processing power.

Figure 20 shows the compilation of distribution patterns for Android Lollipop 5.0.2 that was installed in the target VMs. It shows the distribution patterns of the difference between successive timestamps from the VMs target machine and the reoccurrence rate in 1,000,000 ICMP timestamp data. In Figure 20, observation that can be made is, the peaks for the reoccurrence rate are at 0 for all the VMs. 70% of the timestamps from Xen and VMWare have the same value as the timestamps from the previous sequence packets, where $(n+1)$ th $-$ n th $= 0$, while 50% of the timestamps from VirtualBox have the same value as the timestamps from the previous packets.

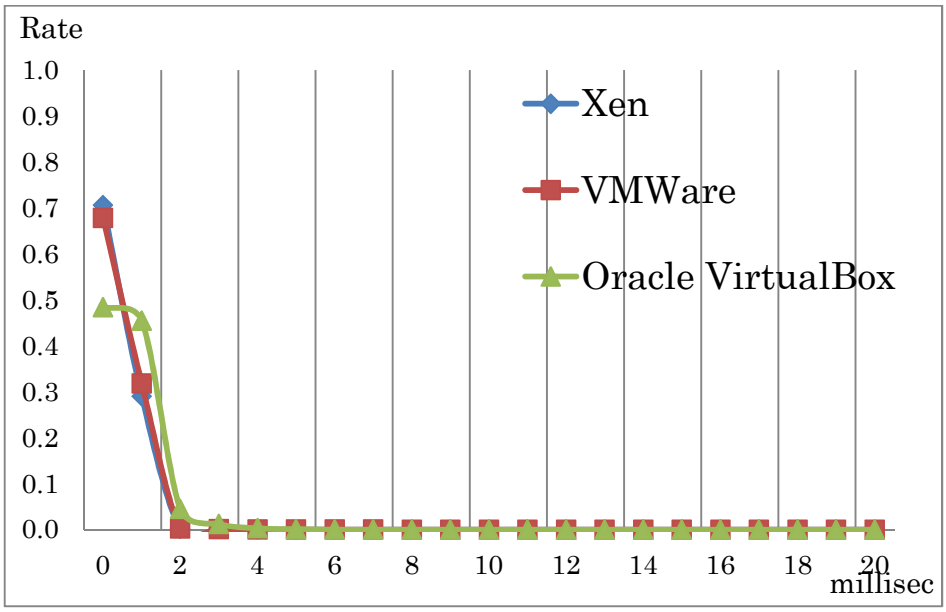


Figure 20: Timestamps differences when Android installed as emulator on different types of VMs

From the distributions graphs in Figure 19 and 20, this research could notice a compelling different of the characteristic patterns of IP timestamps from Android OS running on mobile device environment and VMs.

Further data analysis also shows that the data for IP and ICMP timestamps from the mobile device replicated the phenomenon as per study completed in [17], where different IP and ICMP timestamps in the same packet could be observed. As displayed in Figure 21, 3.31% of the packets from the Android KitKat 4.4.4 give difference value of 1 millisecond between the value of IP and ICMP timestamps in the same packet. Same characteristic was detected in 2.69 % of the received packets from Android Lollipop 5.0.2, 2.57 % from Android Ice Cream Sandwich 4.0.4 and 2.21% from Android Jelly Bean 4.2.2.

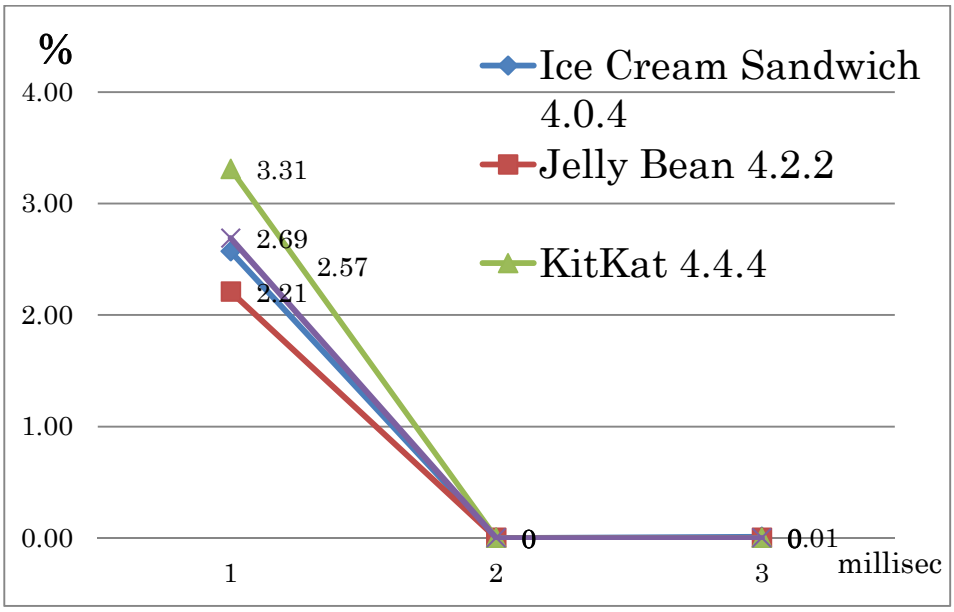


Figure 21: IP and ICMP timestamps differences for 4 versions of Android

6.5. Concluding Remarks

This research could clearly observe that characteristic patterns of timestamps from Android OS on mobile device and VMs are distinguishable. Experiments were conducted to gather and analyze data to determine the difference of successive ICMP timestamps and reoccurrence rate for timestamp difference for Android. This research observed that the timestamps differences between timestamp and the successive timestamps in mobile device are 2, 3 and 4 in 4 versions of Android OSs. Meanwhile, for the latest version of Android OS in major hypervisor products, experiments conducted in this dissertation found out that the difference is almost 0 for timestamps differences between timestamp and the successive timestamp.

The results also showed that IP and ICMP timestamps were deviates for the timestamps replied from mobile device installed with Android OS. This characteristic could not be observed in the packets replied from the VMs target machine because of the high performance machine that used to host VM. Due to this different in characteristic patterns of IP and ICMP timestamps, this research had

proved hypothesis that IP and ICMP timestamps pattern characteristic could be used in detecting either the target machine is running Android on VM environment or on mobile devices, therefore enabling the detection of VM environment.

This research also showed that machine performance could be exploited in detecting the environment in which Android OS is running. Thus, mobile devices that have limitation in performance need to address this issue which could become vulnerability for the mobile devices with Android OS.

Wireless local area networks (WLANs) or hotspots [97] or commonly known as "Wi-Fi" [98] provides a convenient, cost-effective means for network connectivity in designated areas. With the changing mobile computing landscape that empowers mobile device users to access on-line on the go through this Wi-Fi., it is vital for security related studies to be performed in such environment. Concerns regarding security and privacy with the expanding usage of Wi-Fi discussed in various studies [99-101].

Furthermore, with the current trend of Bring Your Own Device (BYOD) to workplace, employees are bringing their personal mobile devices to access applications and corporate data in the corporate internal network. This could cause security issue within the corporation. Mobile device could be affected with the malware or spoofing tools in non-secure Wi-Fi connection and when it access the corporation environment, malware could start stealing the information within the corporation [102, 103].

This research had shown that IP and ICMP timestamps could be used in differencing between Android in mobile device and VM environment. VMs are normally installed on high performance machine in cloud computing environment whereas mobile devices have limited resources such as the processing power. Due to this, it creates different characteristic patterns of IP and ICMP timestamps in the replied packets from Android OS on mobile device and VMs. This scenario could be used by malware to differentiate the Android OS running environment.

In such scenario where the infected mobile device is connected to corporate internal network, the detection method using IP and ICMP timestamps could be

used in sniffing the internal network to avoid from infecting Android OS implemented in VMs while targeting only Android in mobile devices. This could create security issue within the corporation. Similar method could also be used by malware in hiding its malicious behavior from being detected by security services running on VM, for example by connecting to a command and control server and gathering the IP and ICMP of the running environment.

In conclusion, from results in this research, this study showed that Android OS running on mobile device could create security loophole that can be exploited by attackers. Thus as future works in this study, researchers will need to focus not only developing in VM environment that emulates the Android operating system but also emulating the special characteristic of mobile devices such as the IP and ICMP timestamp characteristic pattern that was shown in this research. The gap between applications running on real machine mobile devices should be addressed by making the distribution of timestamps characteristic pattern resolved.

Modification in timestamps reply could be implemented for other environment which is, in this case is VM to meet the characteristic pattern of timestamps distribution of mobile devices as well. In this mobile device case, when the distribution patterns for mobile devices proven to be slower compared to the VMs, the modification that could be implemented in the time-stamping process in a VMs to create delays in the timestamps in the reply packets are proposed. When the packets with the IP timestamp option arrive at the VMs, they should delay using the countermeasure before being forwarded to OS for processing. The delay could be implemented by adjusting the mean number of repetitions of the same IP timestamp in the VMs to match those in the mobile devices. This research is proposing the same method that already being discussed in Chapter 5 could be implemented to implement the delay to hide this difference.

CHAPTER 7

Discussion and Conclusion

This chapter discusses the research flow, experiments towards the hypothesis and the analysis of the results.

7.1. Discussion and Conclusion

In this research, an analysis was performed to verify the remote detection method using network timestamps to differentiate the operating environment in current phenomena of high dependency of computing technologies. The technologies are based on high performance machine and also mobile devices that have limited processing power that provides cyber security scenario. This research aims to validate the applicability of the remote detection method as a potential vulnerability in cyber security attack. Differentiation of operating environment, either they are VM, real machine or mobile device become very important because of the growth of usage of mobile device with seamless interconnectivity. This research contributes to the new knowledge of characteristic pattern of remote detection of operating systems using the network timestamps analysis in recent technology scenario. Base on the characteristic pattern obtain from this research, countermeasure was proposed to hide the differences observed and this research serves as an initiative in improving the cyber security that focusing on remote detection of operating systems using the network timestamps.

Previous work by Kohno [16] highlighted the potential of remote detection method by using network timestamps. Then, work by Shimamura [17] explored on characteristic pattern, based on differences between IP and ICMP timestamps in 1 packet. In [17], experiments were done to detect discrepancies of timestamps in two cases, when ICMP timestamp + 1 < IP timestamp and when ICMP timestamp < IP timestamp. The experimental results showed that discrepancies occurred only 0.12% from total 1,000,000 packets in the case when ICMP timestamp + 1 < IP timestamp for real environment. However, for VMs, discrepancies occurred in the

range between 0.1% until 0.5% from total 1,000,000 packets. On the other hand, in the case when ICMP timestamp < IP timestamp, the study found out that 0% of discrepancies occurred from total 1,000,000 packets from real environment and the range of 0% until 2.5% of discrepancies occurrence for VMs.

This dissertation validated the remote detection method by using characteristic pattern differences between IP and ICMP timestamps that proposed in [17] in current technology scenario which include mobile devices. However, the experiments result showed that the same result was not reproducible when machine with higher performance, which is 2.40 GHz clock speed was used as hardware to host experiments conducted in this research. In [17], machine with 1.86 GHz was used in the experiments. Since that machines with better CPU performance is widely being used, new approach is proposed in this dissertation to remotely detect the operating environment.

In this dissertation, in order to determine the characteristic pattern of network timestamps which are IP and ICMP timestamps in various environments, proposal of determining characteristic patterns had been made and method of analysis as following are being used:

1) Differences between 2 successive timestamps in the replied packets

2) How many times identical timestamps was stamped between the packets

This research explores remote detection method by using characteristic pattern differences between IP and ICMP timestamps in two measurements mentioned above and also validate remote detection method by using characteristic pattern differences between IP and ICMP timestamps that proposed by other researchers in current technology scenario. This dissertation provides the initial finding for characteristic patterns for how network timestamps differences in term of 2 successive timestamps and how many same timestamps stamped could be used. This dissertation also could serve as the research that ongoing as machines are improving rapidly in term of performance but little studies had been done to address this issue.

As the flow of this dissertation, firstly, literature review was done to gain deep understanding regarding the existing issues and in remote detection method of operating environment. From the literature review, it is understood that there are substantial motivations of cyber security attacker to detect VM operating environment in order to make sure that they are not running their malicious activities in VM environment, which have high potential to implement as security analysis systems. Thus reducing the possibility for their program to be trapped in VM-based honeypot and reveals their existence to security analysis system.

Applicability of the remote detection method in simulated environment of high performance machines compared to the previous machine types that were used in [17] was revalidated. In this study, high performance machine is defined as high specifications machines that could be used as a server in cloud computing environments. The goals and objectives are to investigate the differences in IP and ICMP timestamps characteristic patterns in different machine and VM technologies. Experiments were implemented using full virtualization type of VM technologies. In the case study, this method also was used as well to validate it in real environment and VM environment. The results and data were analyzed to see IP and ICMP characteristic pattern differences from the target operating environments.

In this research, all the experiments were conducted in a controlled environment that was set up in a university laboratory. Focus of the experiment is to determine the characteristics of the data that might vary from device to device, from one VM technology to another, and with changes in the implementation environment. Network latency issue was not addressed in this research. This research hypothesized that VM environment could be detected by comparing the behavior patterns of IP and ICMP timestamps in one packet and also in packets that are continuously sent from client measurement machine to target machines. The analysis for the characteristic pattern in mobile devices that are using 4 versions of Android were also tested and the findings serve as vital point in remote detection of mobile devices operating environment. It shows the potential vulnerability that could be used by attackers and therefore putting in risks the personal details, data and information that are stored in mobile devices by personal users and employees in organizations.

Experiments were done to validate the scenario for full virtualization technology using VMWare vSphere, Oracle VirtualBox and Xen Hypervisor in high performance machine. Timestamps replies that were received at the requesting machine were compiled to the nearest millisecond. Even in full virtualization VM case, since there will be VM interface between the CPU and the network interface, it is expected that there will be a small delay for measurer machine in receiving the timestamps between the requests. Through the experiments, it was proven that even though VM are set to imitate the real environment, the technology still unable to stamp timestamps as per real environment. The experiments result shows that network stamping deferral behavior still exist and clearly could be observed in full-virtualization VM target hosts because VM sometimes interrupted timestamp operations to complete other operations. In the experiments to obtain data for data analysis, t_n was define as the value of IP timestamp of the n times for the packet replies from the targeted server. Then, the differences of the IP timestamp values for the t_n+1th – t_n th was calculated. The differences between t_n+1th and t_n th timestamps data were small for the timestamps replied from real machine but showed negative and large numbers differences for t_n+1th and t_n th of the timestamps from the virtual machine, either from both VM Player or Virtual Box. In contrast, results from real machine environment, the differences between timestamps stamping were small, 1-digit time difference or less and in most cases, same timestamp was stamped for more than 2 times continuously. However, in VM, the differences were bigger because VM operations were switched with other operations in queue and this affected the VM clock that is managed by timer device emulation which called as VM switch. Thus, although VM timestamp adjustment was made to make it look like real environment, the behavior difference still could be seen between the timestamp due to the operation processes within the VM technology itself. Therefore, the experiments confirmed that the difference in the timestamp replies behavior could clearly be seen between the replies sent by real machine and VM full virtualized environment.

On the other hand, it will not be an issue in real machine since it will only have CPU and network interface interactions. Hence, from the results that were obtained from analyzing and comparing the timestamps reply data between full

virtualization VM and real machine, as expected the timestamps value from VM changed more frequently compared to the timestamp replies in real machine. Base on the observation in full virtualization VM environment that the behavior of timestamps stamping is different which are, lesser same timestamps were sent in the replied packets received by the requestor.

Behavior pattern differences had clearly been seen in the experiments. By comparing the real environment and full virtualization VM environment on the numbers of how many times the same timestamps were replied in the received packets, it was shown that, in real machine, more than 60% of the same timestamps are stamped for 5 times in the continuously replied packets. In contrast, there were no same timestamps stamped 5 times in the replied packets from VirtualBox and VMWare ESX. The reason is, VM sometimes interrupted timestamps operations to complete other operations and make the time taken to complete job longer than real machine. Even though in full virtualization that simplifies migration and portability, the remote detection by using IP timestamps packet reply still could be observed and this could reveal the environment that one system is running on. Based on this analysis of the results, this research had proven that VM are clearly detectable remotely by analyzing the replies from IP timestamp request packets in cloud computing environment. This research contributes to identifying and proved that distinguishable differences in the timestamp replies from VM and real machines even in a high performance cloud computing environment exist and need to be addressed.

This research also investigated and validated the remote detection using the same method for mobile device that runs on Android OS. In the experiments, major full virtualization hypervisor products, Oracle VirtualBox, VMWare and Xen were used as the Android OS emulators. Open source Android Lollipop 5.0.2 with 1GB of virtual memory and IDE HDD with 16GB of virtual storage was set up as the Android OS on the VMs and tests were done accordingly. As for the measurement target in mobile device, Android was installed on Sony Xperia SO-04E and tests were done separately on 4 different Android versions which are Android Ice Cream Sandwich 4.0.4, Android Jelly Bean 4.2.2, Android KitKat 4.4.4, and Android Lollipop 5.0.2.

This research posited that using the same remote detection method, by analyzing the characteristic pattern of IP and ICMP timestamps in one packet and also in packets that were received continuously from Android OS running on mobile device, the mobile device could be detected remotely. Mobile devices such as smartphone are constrained by their limited processing power etc. This research predicted that timestamps differences in continuously sent packets from mobile devices could be bigger rather than the timestamps differences of VM running on high performance machine that this research validated in previous experiments.

Experiments results proven that IP timestamps differences from mobile device were bigger than those from VM, and this is as per prediction done at the early stage of research. Characteristic patterns of timestamps from Android OS on mobile device and VMs are distinguishable. Experiments were done to gather and analyze data to determine the differences of successive ICMP timestamps and reoccurrence rate for timestamp difference for Android. Experiments in this research provide results that the timestamps differences between timestamp and the successive timestamps in mobile device are 2, 3 and 4 milliseconds in 4 versions of Android OSs. Meanwhile, for the latest version of Android OS in major hypervisor products, this research found out that the differences are either 0 or 1 millisecond for timestamps differences between timestamps and the successive timestamps. This means that for Android that installed on VMs, identical timestamps was frequently been stamped. One more thing that could be learned here was that, during the 3 years of experiments that were done for this research, the VMs hypervisor technology keep improving and the imitation of real environment in VMs (VMWare, VirtualBox and Xen) are getting more better compared to 3 years ago where timestamping pattern differences could be observed more clearly.

The result shows that the data for IP and ICMP timestamps from the mobile device replicated the phenomenon as per study completed in [17], where different IP and ICMP timestamps in same packet could be observed. The result shows that 3.31% of the packets from the Android KitKat 4.4.4 give difference value of 1 between the value of IP and ICMP timestamps in the same packet. Same characteristic were detected in 2.69 % of the received packets from Android Lollipop

5.0.2, 2.57 % from Android Ice Cream Sandwich 4.0.4 and 2.21% from Android Jelly Bean 4.2.2.

From the results analysis, it had shown that IP and ICMP timestamps were deviates for the timestamps replied from mobile device installed with Android OS. This characteristic could not be observed in the packets replied from the VMs target machine because of the high performance machine that used to host VM. Due to this different in characteristic patterns of IP and ICMP timestamps, this research had proved hypothesis that IP and ICMP timestamps pattern characteristic could be used in detecting either the target machine is running Android on VM environment or on mobile devices, therefore enabling the detection of VM environment.

These results conclude that timestamps reply are strongly related to performance of the machine that it is operating. The higher the spec of the machine, the faster timestamps reply will be sent back and more identical timestamps will be stamped. Mobile device that has limited processing power will be slower in replying timestamps and will contribute to characteristic pattern of bigger timestamps differences between the successive timestamps sent through reply packets. The results exposed the vulnerability of remote detection of android operating system on mobile devices. This research showed that the limitation of machine performance could be exploited in detecting the environment in which Android OS is running. Thus, mobile devices that have limitation in performance need to address this issue which could become vulnerability for the mobile devices with Android OS.

Likewise, with the seamless internet connection almost everywhere and wireless local area networks (WLANs) or hotspots or commonly known as “Wi-Fi” provides a convenient, cost-effective means for network connectivity in designated areas, means that attacker could manipulate the time frame when the mobile devices connected to particular network to remotely detect the running environment and perform malicious activities once they detected that they are not in the VM environment. This also avoids the risk of them being trapped in potential malware analysis environments.

As the countermeasure for the timestamps characteristics pattern differences of VM and real machine, countermeasure in hiding the differences by

making modification to the real machines so that similar characteristic patterns of timestamps as those from VM will be sent in the reply packets is proposed. Modification implemented in the time-stamping process in a real machine to create delays in the timestamps in the reply packets.

When the packets with the IP timestamp option arrive at the real machine, they are delayed using the countermeasure before being forwarded to OS for processing. The delay is implemented by adjusting the mean number of repetitions of the same IP timestamp in high performance machine to match those in the lower performance machine. This modification technique was tested and published in [104]. Since that, mobile devices are the lowest in machine performance. Mobile device characteristic behavior should be the bench mark. Thus, proposed delay countermeasure should be implemented in VM to change the characteristic pattern for IP and ICMP timestamps reply in VM to close the gap between VM and Android.

7.2. Challenges and Limitations

Several challenges and limitations exist and were discovered while performing the experiments in this research. This dissertation does not address the latency issues. This research also does not include the malware analysis using real malware samples and malware behavior hypothesis made by referencing to literature review. This dissertation is limited to operating environment remote detection method using network timestamps. Experiments and analysis were not done in the operating environment using real malware sample.

7.3. Future Works

As the future works, experiments could be run to identify and analyze malware that attached to applications installed on a mobile device, and also popular applications that injected with a malicious code. Then, behavior analysis of this real malware could be done to determine how the malware behavior changed according to the running environment. The remote detection method and relationship with malware

behavior could be implied in bigger framework during security policy implementation to avoid cyber security attack by taking consideration of malware behavior could be different because of the operating environment. The weakest environment such as, mobile device characteristic patterns should be the benchmark on how the timestamps characteristic pattern should be, such as mobile device which was shown in this research. More tests in a different environment using different machines and different implementation styles and on a grid and cloud test bed also should be done.

Acknowledgments

Alhamdulillah, praise to Allah, the Most Gracious, the Most Merciful.

First and foremost, I would like to address my highest gratitude to my supervisor for this dissertation, Professor Dr. Toshiyuki Kinoshita and Assistant Professor Dr. Ryuya Uda for their supervision, guidance, advice and help. They had supported me throughout this dissertation with their patience and knowledge, and without their encouragement and effort this dissertation would not have been completed or written. One simply could not wish for a better or friendlier supervisor.

I would like to thank my supportive husband, Khairul Khalil Ishak and my kids, Adam Ariff Khairul Khalil and Hani Erina Khairul Khalil for their endless support during the duration in preparing this dissertation and also during the course of my PhD degree program.

I also would like to thank my parents, Mat Razali Jaafar and Anisah Ibrahim, my father and mother in law, Ishak Surin and Latifah Rauf and may siblings for their loving support and encouragement.

I also would like to thanks my sponsors, the government of Malaysia, the Ministry of Higher Education Malaysia and National Defence University of Malaysia for granting me the scholarship to complete my PhD.

Finally I thank all my friends in Computer Science Program, Graduate School of Bionics, Computer and Media Sciences, Tokyo University of Technology especially Hiroshi Maeda, Madoka Shiratori and Itaru Koike that work closely with me for their support and kindness. I have been blessed with a friendly and cheerful group of fellow students.

Best Regards,

Noor Afiza Binti Mat Razali

References

[1] C. Beek, C. Cochin, A. Hinchliffe, J. Jarvis, H. Li, Q. Liu, D. Mandal, M. Rosenquist, R. Samani, R. Sherstobitoff, "McAfee Labs Threats Predictions 2016", McAfee Labs, 2016

[2] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility", Future Generation Computer Systems vol. 25, pp. 599-616, 2009

[3] M. Nazir, P. Tiwari, S.D. Tiwari, R.G. Mishra, "Cloud Computing: An Overview, Book Chapter of Cloud Computing: Reviews, Surveys, Tools, Techniques and Applications", An Open-Access eBook published by HCTL Open, January 2015

[4] Z. Tari, X.Yi, U.S. Premarathne, P. Bertok, I. Khalil, "Security and Privacy in Cloud Computing: Vision, Trends, and Challenges", Cloud Computing IEEE, vol. 2(2), pp. 30-38, 2015

[5] J. Shayan, A. Azarnik, S. Chuprat, S. Karamizadeh and M. Alizadeh, "Identifying Benefits and risks associated with utilizing cloud computing", International Journal of Soft Computing and Software Engineering [JSCSE], vol. 3, no. 3, pp. 416-421, 2013

[6] A. Gupta, C. Milanesi, R. Cozza, C.K. Lu, "Market Share Analysis: Mobile Phones, Worldwide, 2Q13", Gartner, 2013

[7] A. Bendovschi, "Cyber-Attacks–Trends, Patterns and Security Countermeasures", Procedia Economics and Finance vol. 28 pp. 24-31, 2015

[8] A. Ahmad, S.B. Maynard, S. Park, "Information security strategies: towards an organizational multi-strategy perspective", Journal of Intelligent Manufacturing, vol. 25(2), pp. 357-370, 2014

[9] N. Leavitt, "Malicious code moves to mobile devices", Computer, vol. 12, pp. 16-19, 2000

[10] A. Favell, “96 percent of smartphones and tablets lack necessary security software. Why it matters to your business – a lot”, 2011, [cited January 2016], Available from: https://mobiforge.com/news-comment/96-percent-smartphones-and-tablets-lack-necessary-security-software-why-it-matters-to-your-business

[11] R. de Oliveira Albuquerque, L.J.G. Villalba, A.L.S. Orozco, R.T. de Sousa Júnior and T.H. Kim, “Leveraging information security and computational trust for cybersecurity”, The Journal of Supercomputing 2015, pp. 1-35, 2015

[12] A. Munshi, P. Dell, H. Armstrong, “Insider threat behavior factors: A comparison of theory with reported incidents”, Proceedings of the IEEE International Conference on System Science (HICSS), Hawaii, 2012

[13] A. Mylonas, M. Theoharidou, D. Gritzalis, “Assessing privacy risks in android: A user-centric approach”, in Springer Risk Assessment and Risk-Driven Testing, pp. 21-37, 2013

[14] Symantec Corp., “Internet Security Threat Report 2014”, 2016

[15] A. Mylonas, A. Kastania, D. Gritzalis, “Delegate the smartphone user? Security awareness in smartphone platforms”, Computers & Security, pp. 47-66, 2013

[16] T. Kohno, “Remote physical device fingerprinting”, IEEE Transactions on Dependable and Secure Computing, vol. 2(2), pp. 93, 2005

[17] M. Shimamura, K. Kono, “Remote Virtual Machine Monitor Detection Using Network Timestamp,” Information Processing Society of Japan(IPSJ), vol. 50, no. 8 (Japanese), pp. 1870-1882, 2009

[18] P. Ferrie, “Attacks on more virtual machine emulators”, Symantec Technology Exchange, pp.55, 2007

[19] A. Ghosh, P.K. Gajar, S. Rai, “Bring your own device (BYOD): Security risks and mitigating strategies", Journal of Global Research in Computer Science, vol. 4(4), pp. 62-70, 2013

[20] J. Pinchot, K. Paullet, “Bring your own device to work: benefits, security risks and governance issues”, Issues in Information Systems, vol. 16(3), 2015

[21] S. Abraham, I. Chengalur-Smith, “An overview of social engineering malware: Trends, tactics, and implications", Technology in Society, vol. 32(3), pp.183-196, 2010

[22] D. Dagon, T. Martin, T. Starner, “Mobile phones as computing devices: The viruses are coming”, IEEE Transaction of Pervasive Computing, vol. 3(4), pp. 11-15, 2004

[23] A. Gostev, D. Maslenikov, “Mobile malware evolution: An overview”, Kaspersky Labs Report on Mobile Viruses, 2006

[24] M. La Polla, F. Martinelli, D. Sgandurra, “A survey on security for mobile devices”, IEEE Communications surveys & tutorials, vol.15(1), pp. 446-471, 2013

[25] J. Hong, “The state of phishing attacks", ACM Communications, vol. 55(1), pp. 74-81, 2012

[26] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system”, 2008

[27] J. Oberheide, K. Veeraraghavan, E. Cooke, J. Flinn, F. Jahanian, “Virtualized in-cloud security services for mobile devices”, Proceedings of the 1st Workshop on Virtualization in Mobile Computing, June 2008

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

[29] F. Liu, P. Shu, H.Jin, L. Ding, J.Yu, D. Niu, B. Li, "Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications", IEEE Wireless Communications, vol. 20(3), pp. 14-22, 2013

[30] S. Zonouz, A. Houmansadr, R. Berthier, N. Borisov, W. Sanders, "Secloud: A cloud-based comprehensive and lightweight security solution for smartphones", Computers & Security, vol. 37, pp. 215-227, 2013

[31] G. Suarez-Tangil, J.E Tapiador, P. Peris-Lopez , A. Ribagorda, "Evolution, detection and analysis of malware for smart devices", IEEE Communications Surveys & Tutorials, vol. 16(2), pp. 961-987, 2014

[32] T. Garfinkel, M. Rosenblum, "A Virtual Machine Introspection Based Architecture for Intrusion Detection", Network and Distributed System Security Symposium (NDSS), vol. 3, pp. 191-206, 2003

[33] H.J. Liao, C.H.R. Lin, Y.C. Lin, K.Y. Tung, "Intrusion detection system: A compre-hensive review", Journal of Network and Computer Applications, vol. 36(1), pp. 16-24, 2013

[34] P. Ferrie, "Attacks on more virtual machine emulators", Symantec Technology Exchange, 2007

[35] J.E. Smith, R. Nair, "The architecture of virtual machines", Computer, vol. 38(5), pp. 32-38, 2005

[36] V. Mauch, M. Kunze, M. Hillenbrand, "High performance cloud computing", Future Generation Computer Systems, vol. 29(6), pp. 1408-1416, 2013

[37] R. Perez, L. van Doorn, R. Sailer, "Virtualization and hardware-based security", IEEE Security & Privacy, vol. 2008 (5), pp. 24-31,2008

[38] M. Armbrust, A. Fox, R. Griffith, A.D Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica and M. Zaharia, "A view of cloud computing", ACM Communications, vol. 53(4), pp. 50-58, 2010

[39] S. Shang, S. Zhang, X. Chen, X. Huo, "Cloud computing research and development trend", IEEE Second International Conference on Future Networks (ICFN'10), 2010

[40] P. Mell, T. Grance, "The NIST definition of cloud computing", pp. 20-23, 2011

[41] R. Buyya, C.S. Yeo, S. Venugopal, "Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities", 10th IEEE International Conference on High Performance Computing and Communications (HPCC'08), 2008

[42] R. Schwarzkopf, M. Schmidt, C. Strack, S. Martin, B. Freisleben, “Increasing virtual machine security in cloud environments”, Journal of Cloud Computing, vol. 1(1), pp. 1-12, 2012

[43] K. Hashizume, D.G. Rosado, E. Fernández-Medina, E.B. Fernandez, “An analysis of security issues for cloud computing”, Journal of Internet Services and Applications, vol. 4(1), pp. 1-13, 2013

[44] M. Ali, S.U. Khan, A.V. Vasilakos, “Security in cloud computing: Opportunities and challenges”, Information Sciences, vol. 305, pp. 357-383, 2015

[45] F. Lombardi, R. Di Pietro, “Secure virtualization for cloud computing”, Journal of Network and Computer Applications, vol. 34(4), pp. 1113-1122, 2011

[46] H.T Dinh, C. Lee, D. Niyato, P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches”, Wireless communications and mobile computing, vol.13 (18), pp. 1587-1611, 2013

[47] S.A.E.I.D. Abolfazli, Z. Sanaei, M. Sanaei, M. Shojafar, A. Gani. “Mobile cloud computing: the-state-of-the-art, challenges, and future research” Encyclopedia of Cloud Computing, Wiley, 2015

[48] A.A. Omella, “Methods for virtual machine detection”, Grupo S21sec, 2006

[49] J. Franklin, M. Luk, J.M. McCune, A. Seshadri, A. Perrig, L. Van Doorn, “Remote detection of virtual machine monitors with fuzzy benchmarking”, ACM SIGOPS Operating Systems Review, vol. 42(3), pp. 83-92, 2008

[50] C. Thompson, M. Huntley, C. Link, “Virtualization detection: New strategies and their effectiveness”, Accessed January 2016:
http: //www-users. cs.umn.edu/cthomp/papers/vmm-detect-20. 1

[51] Juniper Networks, "Juniper Networks Third Annual Mobile Threats Report, March 2012 through March 2013", pp.4-6, 2013

[52] Q.U. Bo, X. Wang, K. Sanders, “Detecting malware”, U.S. Patent 9,104,870, issued August 11, 2015

[53] C.H. Smith, K. Maclean, J.J. Liu, S. Mann, M.A.N.N. Wendy, R.Chapin "System and method for advanced malware analysis" U.S. Patent 9,106,692, issued August 11, 2015

[54] R. Kozik, M. Choras, "Current cyber security threats and challenges in critical infrastructures protection", IEEE Second International Conference on Informatics and Applications (ICIA), 2013

[55] R. Von Solms, J. Van Niekerk, "From information security to cyber security", Computers & Security, vol. 38, pp. 97-102, 2013

[56] M. Hashim, "Malaysia's National Cyber Security Policy", 2011

[57] R. Pilling, "Global threats, cyber-security nightmares and how to protect against them", Computer Fraud & Security, vol. 2013(9), pp. 14-18, 2013

[58] B. Cashell, W.D. Jackson, M. Jickling, B. Webel, "The economic impact of cyber-attacks", Technical Report RL32331, U.S.A. Government and Finance Division, April 2004

[59] H. Saini, Y.S. Rao, T. Panda, "Cyber-crimes and their impacts: A review", International Journal of Engineering Research and Applications, vol. 2(2), pp. 202-209, 2012

[60] J. Hua, S. Bapna, "The economic impact of cyber terrorism", The Journal of Strategic Information Systems, vol. 22(2), pp. 175-186, 2013

[61] J.S. Hiller, R.S. Russell, "The challenge and imperative of private sector cyber-security: An international comparison", Computer Law & Security Review, vol. 29(3), pp. 236-245, 2013

[62] A. Razzaq, A. Hur, H. Farooq, Ahmad, M. Masood. "Cyber security: threats, reasons, challenges, methodologies and state of the art solutions for industrial applications", IEEE Eleventh International Symposium on Autonomous Decentralized Systems (ISADS), pp. 1-6, 2013

[63] S. Sicari, A. Rizzardi, L.A. Grieco, A. Coen-Porisini, “Security, privacy and trust in Internet of Things: The road ahead. Computer Networks”, vol. 76, pp. 146-164, 2015

[64] R. Alur, E. Berger, A.W. Drobnis, L. Fix, K. Fu, G.D. Hager, D. Lopresti, K. Nahrstedt, E.Mynatt, S.Patel, J. Rexford, “Systems Computing Challenges in the Internet of Things”, 2016

[65] M. Dawson, J. Wright, M. Omar, “Mobile Devices: The Case for Cyber Security”, New Threats and Countermeasures in Digital Crime and Cyber Terrorism, pp. 8, 2015

[66] Lookout Inc, “Enterprise Mobile Threat Report 2016”, 2016

[67] R.P Jover, P. Giura, “How vulnerabilities in wireless networks can enable advanced persistent threats”, International Journal on Information Technology (IREIT), vol. 1(2), pp. 145-151, 2013

[68] M.O. Nassar, “Wireless and Mobile Computing Security Challenges and Their Possible Solutions”, American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS), vol. 3(1), pp. 66-74, 2015

[69] A. Arabo, B. Pranggono, “Mobile malware and smart device security: Trends, challenges and solutions”, IEEE 19th International Conference on Control Systems and Computer Science (CSCS), 2013

[70] S. Mansfield-Devine, “Android malware and mitigations”, Network Security, vol. 2012(11), pp. 12-20, 2012

[71] M. Rahman, B. Carbunar, D.H. Chau, “FairPlay: Fraud and Malware Detection in Google Play”, SIAM International Conference on Data Mining (SDM), May 2016

[72] P. Faruki, A. Bharmal, V. Laxmi, V. Ganmoor, M.S. Gaur, M. Conti and M. Rajarajan, “Android security: a survey of issues, malware penetration, and defenses”, IEEE Communications Surveys & Tutorials, vol. 17(2), pp. 998-1022, 2015

[73] A. Karim, S.A.A. Shah, R.B. Salleh, M. Arif, R.M. Noor, S. Shamshirband, “Mobile Botnet Attacks-an Emerging Threat: Classification, Review and Open Issues”, TIIS 9, no. 4 , pp. 1471-1492, 2015

[74] P. Far na, E. Cambiaso, G. Papaleo, M. Aiello,“Are mobile botnets a possible threat? The case of SlowBot Net”, Computers & Security, vol. 58, pp. 268-283, 2016

[75] A. Shabtai, L. Tenenboim-Chekina, D. Mimran, L.Rokach, B.Shapira, Y. Elovici, “Mobile malware detection through analysis of deviations in application network behavior”, Computers & Security, vol. 43, pp. 1-18, 2014

[76] R. Uda, “Protocol and Method for Preventing Attacks from the Web”, World Academy of Science, Engineering and Technology, vol.76, pp. 456-460, 2011

[77] M.R. Chouchane, A. Lakhotia, “Using engine signature to detect metamorphic malware”, Proceedings of the 4th ACM workshop on Recurring Malcode, 2006

[78] D. Perez-Botero, J. Szefer and R.B. Lee, “Characterizing hypervisor vulnerabilities in cloud computing servers”, Proceedings of the ACM International Workshop on Security in Cloud Computing, 2013

[79] M. Portnoy, “Virtualization essentials”, John Wiley & Sons, Vol. 19, 2012

[80] Z. Su, “Specification of the Internet Protocol (IP) timestamp option”, 1981

[81] Google Inc, "Developer Android Dashboard",
URL: http://developer.android.com/about/dashboards/index.html,
[Accessed: April 2016]

[82] T. Garfinkel, K. Adams, A. Warfield, J. Franklin, “Compatibility is not transpar-ency: VMM detection myths and realities,” Proceedings of the 11th USENIX workshop on Hot topics in operating systems, USENIX Association, pp. 1-6, 2007

[83] K. Miyamoto, H. Tanaka, “Proposal of Effective Detection Method of VMM without Feature Database”, Information Processing Society of Japan, vol. 52 (Japanese), pp. 2602-2612, 2011

[84] T. Raffetseder, C. Kruegel, and E. Kirda, “Detecting system emulators,” in Springer Information Security, pp. 1-18, 2007

[85] H. Alsaffar, D. Johnson, “Covert Channel using the IP Timestamp Option of an IPv4 Packet”, in The International Conference on Electrical and Bio-medical Engineering, Clean Energy and Green Computing, The Society of Digital Information and Wireless Communication, 2015

[86] M. Cristea, B. Groza, “Fingerprinting Smartphones Remotely via ICMP Time-stamps”, IEEE Communications Letters, vol. 17(6), pp. 1081-1083, 2013

[87] M-H. Lu, P. Steenkiste, T. Chen, “Video streaming over 802.11 WLAN with content-aware adaptive retry”, IEEE International Conference on Multimedia and Expo, 2005

[88] R. Fonseca, G. Porter, R. Katz, S. Shenker, I. Stoica, “IP options are not an option", Univ. of California, Berkeley, 2005

[89] D. Mills, Network Time Protocol (Version 3) specification, Implementation and Analysis, 1992

[90] M. Noorafiza, H. Maeda., R. Uda, T. Kinoshita, M. Shiratori, “Vulnerability Analy-sis using Network Timestamps in Full Virtualization Virtual Machine,” in 1st International Conference on Information Systems Security and Privacy (ICISSP 2015), SCITEPRESS Digital Library, 2015

[91] VMware, Inc., “Timekeeping In Virtual Machines”, URL:https://www.vmware.com/files/pdf/Time keeping-In-VirtualMachines.pdf, 2011 [Accessed: January 2014]

[92] M. Noorafiza, H. Maeda, R. Uda, T. Kinoshita, “Virtual Machine Remote Detection Method using Network Timestamp in Cloud Computing,” International Conference on Information Science and Technology (ICITST), pp. 380-385, December 2013

[93] A. J. Younge, R. Henschel, J.T. Brown, G. Laszewski, J. Qiu, G.C. Fox. "Analysis of Virtualization Technologies for High Performance Computing Environments", IEEE International Conference on Cloud Computing (CLOUD), 2011

[94] M. Rosenblum, "VMWare's Virtual Platform: A virtual machine monitor for commodity PCs", 1999

[95] J. Watson, "VirtualBox: bits and bytes masquerading as machines", Linux Journal, vol. 166, pp. 1, 2008

[96] P. Barham, "Xen and the art of virtualization," Proceedings of the 19th ACM Sym-posium on Operating systems principles, pp. 164-177, 2003

[97] A. Balachandran, G.M. Voelker, P. Bahl, "Wireless hotspots: current challenges and future directions", Mobile Networks and Applications, vol. 10 no. 3, pp. 265-274, 2005

[98] N. Piscataway, "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications", IEEE P802.11 D3, 1996

List of publications that related to this dissertation

Journals

1. M. Noorafiza, K.K. Ishak, H. Maeda, M. Shiratori, T. Kinoshita, and R. Uda, "Characteristic Patterns of Timestamps from Android Operating System on Mobile Device and Virtual Machine", IAENG International Journal of Computer Science, vol. 43, no. 2, pp. 212-218, June 2016

2. M. Noorafiza, H. Maeda, T. Kinoshita R.Uda, "Virtual Machines Detection Methods Using IP Timestamps Pattern Characteristic", International Journal of Computer Science & Information Technology (IJCSIT) vol. 8, no. 1, pp. 1-15, February 2016

International Conference Proceedings

1. M. Noorafiza, H. Maeda, M. Shiratori, T. Kinoshita R. Uda, "Vulnerability Analysis using Network Timestamps in Full Virtualization Virtual Machine", in Proceedings of the 1st International Conference on Information Systems Security and Privacy (ICISSP-2015), pp. 83-89, February 2015

List of publications that published during PhD. candidacy but not related to this dissertation

Journals

1. T. Kinoshita, M. Noorafiza, K. Katsumata, "Performance Evaluation Technique for Computer Systems with Finite Input Source," International Journal of Computer Applications (IJCA), accepted for publication, 2016

2. J. Ishii, M. Noorafiza, S. Tezuka, R. Uda, T. Kinoshita, "Confidential Information Poisoning Methods by Considering the Information Length in Electronic Portable Devices", Information Processing Society of Japan, vol.54, no.10, pp. 1-16, October 2013

International Conference Proceedings

1. M. Noorafiza, H. Maeda, T. Kinoshita, R. Uda, "Virtual Machine Remote Detection Method using Network Timestamps in Cloud Computing", Proceedings of the 8th International Conference for Internet Technology and Secured Transactions (ICITST 2013), pp. 380-385, December 2013

2. K. Katsumata, M. Noorafiza, S. Ito, I. Koike, T. Kinoshita, "Queuing Network Approximation Technique for Evaluating Performance of Computer Systems Acquiring Different Memory resource with Finite Input Source", Proceedings of 31st International Conference on Computers and Their Applications (CATA 2016), pp. 43-49, April 2016

3. M. Hirose, M. Noorafiza, M. Takaya, I. Koike, T. Kinoshita, "Optimum Singularity Size in Data Deduplication Technique", Proceedings of the International Conference on Scientific Computing (CSC 2015), pp. 101-105, July 2015

4. M. Hirose, M. Shiratori, M. Noorafiza, R. Tsuboi, I. Koike, T. Kinoshita, "Queuing Network Approximation Technique for Evaluating Performance of Computer Systems with Finite Input Source", Proceedings of the International Conference on Scientific Computing (CSC 2015), pp. 9-15, July 2015

5. M. Takaya, M. Ogiwara, M. Noorafiza, C. Itaba, I. Koike, T. Kinoshita, "Queuing Network Approximation Technique for Evaluating Performance of Computer Systems with Memory Resource used by Multiple job types", Proceedings of the International Conference on Parallel & Distributed Processing Techniques & Applications (PDPTA 2014), pp. 41-46, July 2014

6. M. Noorafiza, I. Koike, H. Yamasaki, A. Rizalhasrin, T. Kinoshita, "Block Length Optimization in Data Deduplication Technique", Proceedings of the International Conference on Scientific Computing (CSC 2013), pp. 216-220, July 2013

7. J. Ishii, M. Noorafiza, S. Tezuka, R. Uda, T. Kinoshita, “Confidential Information Poisoning Methods by Considering the Information Length in Electronic Portable Devices”, Proceedings of the 26th IEEE International Conference on Advanced Information Networking and Applications (AINA 2012), pp. 78-84, March 2012

8. M. NoorAfiza, T. Kinoshita, A. Tanabe, “Queuing Network Approximation Technique for Evaluating Performance of Computer Systems with Multiple Memory Resource Requirements”, Proceedings of the International Conference on Parallel & Distri-buted Processing Techniques & Applications (PDPTA 2012), pp. 758-763, July 2012

Award

1. Best Paper Award at the 31st International Conference on Computers and Their Applications (CATA 2016), April 2016.
   Paper Title: “Queuing Network Approximation Technique for Evaluating Performance of Computer Systems Acquiring Different Memory resource with Finite Input Source”