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

M. Noorafiza, K.K. Ishak, H. Maeda, M. Shiratori, T. Kinoshita and R. Uda

***Abstract*— 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 system in mobile devices. But as the increase usage of mobile devices, so does the increase of number of malware attacks and security issues in mobile devices. Mobile devices limited resources, i.e., processing, memory, battery power, and lack of storage, prevents the integration of advanced security monitoring solutions in the mobile devices. One of the solutions in addressing this problem is by delegating security monitoring and malware detection for mobile devices to virtual machines (VM) in cloud computing facilities. VMs provide ease of use through their on-demand characteristics and give huge benefits in terms of lowering costs and improving scalability. However, such solution could create critical vulnerability if the malware could detect the VM environment. Upon the detection of VM environment, the malware may not execute its malicious programs, therefore hiding itself from being detected. The malware would only execute malicious programs once detecting the environment is on mobile device. This would have serious consequences for mobile device users, as any applications that have passed a malware detection system on the VM are considered safe and may gain the user's trust. In this paper, we propose a VM and mobile device environments detection method by analyzing the characteristic patterns of ICMP and IP timestamps received from Android OS running on VMs and mobile device. Based from our findings, we could showed that by comparing the characteristic patterns of ICMP and IP timestamps between VM and mobile device environment, the environment could be distinguish between each other, thus enabling the detection of VM environment.**

***Index Terms*— Android, ICMP and IP timestamp, mobile device, virtual machine detection, network security, malware, mobile device.**

Manuscript received 24th January 2016; revised March 7th, 2016.

M. Noorafiza is currently PhD candidate in Tokyo University of Technology, Tokyo, Japan (corresponding author: +81426372631; e-mail: noorafiza.matrazali@gmail.com).

K.K. Ishak is currently PhD candidate in Tokyo University of Technology, Tokyo, Japan (e-mail: kkishak@gmail.com).

H. Maeda completed his Bachelor of Science and Master of Science in Computer Science from Tokyo University of Technology, Tokyo, Japan (e-mail: g211203527@edu.teu.ac.jp).

M. Shiratori is currently undergraduate student in Tokyo University of Technology, Tokyo, Japan (e-mail: m.shiratori@edu.teu.ac.jp).

T. Kinoshita is a Professor in Tokyo University of Technology majoring in Computer Security, System Programing and Queuing Theory (e-mail: kinoshi@stf.teu.ac.jp).

R. Uda is Assistant Professor in Tokyo University of Technology (e-mail: uda@stf.teu.ac.jp).

I. Introduction

Mobile devices are now able to perform many of the operations that had been exclusively done on PCs. Mobile devices use the same architecture as traditional computers; thus they have the same vulnerabilities and security issues faced by PCs [1]. Operating system (OS) that is widely used in mobile devices is Android. According to Gartner Report [2] the Android OS's market share was 79% in August 2013 and it will keep increasing. Android is an open, programmable software framework which is vulnerable to typical mobile device attacks that can make the mobile devices unusable. Moreover, 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 PCs.

With the integration of mobile devices and cloud computing technology with virtual machines (VMs) as the main underlying technology, the lack of resources available in mobile devices for security solutions could be addressed by delegating security monitoring and malware detection to VMs in cloud computing facilities [3],[4].

However, despite the attractiveness of this idea, we argue 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.

There are also high possibilities that development and testing process of the applications for Android, including security checking will be done using emulator in the VM on the cloud computing environment. We argue that the security testing against various malicious codes might not give the true results because the malicious operation may not show their behavior once they had detected that the running environment is VM. As the result, after the application is released, the mobile device 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 VM environment. Therefore data and private information that are stored and communication through the mobile device and smartphone might be revealed to malicious third party. It may also cause expensive billing due to unapproved SMS/MMS subscription

services via smartphones [5].

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 [6].

In this paper, we present the analysis of characteristic patterns of ICMP and IP timestamps from Android OS running on mobile device and VM environment. From the findings, we showed that mobile device and VM environment could be distinguished by examining characteristic patterns of ICMP and IP timestamps characteristic patterns. Such characteristic could be exploited by malware in hiding its malicious programs upon detecting the VM environment.

II. Research Background

In parallel with the growth of mobile devices usage, there has been a significant increase in malware aimed at gathering personal information from mobile devices. This information could later be used by the malware owner or other third party for their personal profit such as for marketing and selling services on the web or profiteering from online banking information [7], [8]. This growing threat points out the need for users to protect their mobile devices by using anti-virus or anti-malware applications that include intrusion detection system (IDS) and intrusion prevention system (IPS) [9]. 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 [10]. In order to conserve mobile resources while improving protection from malware threats, an off-device in-cloud network service could be implemented [11]. Through this approach, security services are delegated to VMs in the cloud system for scanning and protecting mobile device applications, thus 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.

Defense against malicious software for mobile devices also involves in scanning and preventing malicious applications from being published to users. Basic security measures such as application review need to be applied for all the applications that will be released in the application marketplace [12]. In Android case, Google is implementing automated antivirus system called Google Bouncer to remove malicious applications uploaded on to the marketplace. Such system utilizes VMs as their core environment.

VM is one of the underlying technologies in the information technology industry. The VMs are implemented on hypervisor hosts. There are 2 main types of VM hypervisor. Type 1 hypervisors, or bare-metal implementations, run directly on the server hardware without any host operating systems beneath them, whereas Type 2 hypervisors run on top a traditional operating system. Type 2 hypervisors are easy to install and deploy because much of the hardware configuration work such as networking and storage is handled by the underlying operating system [13].

However, even with the significant merit of using VMs as a defense against malware, the idea is still vulnerable due to the possibility of the malware in detecting the system on which it is or will be operating and thereby distinguishing the VM environment. The issue of VM detection has been widely discussed by researchers [14]-[16]. There are a number of techniques for detecting the existence of a VM [17]. Detection method that is done once the detection program is installed and executed on a host is considered the last method that will be used. This is because if the program or software is installed in a host, its existence might be detected and a signature will be generated that may result in their existence being revealed.

Through VM detection, attackers 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 [11], [15]. 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. We believe those intensive studies should also look into VM detection methods and the capability of malware to differentiate VM or mobile device environment.

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 functioning well. Therefore security tests aimed against 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 capturing the correct signature data [18]. As a result when the applications are released, mobile devices that install the applications might be compromised.

In this research, we are analyzing the characteristic of IP and ICMP timestamps patterns from the Android OS that are running on VM or mobile device. Through the analyzed data, we are proposing a VM or mobile device detection method by distinguishing the different in the IP and ICMP timestamps patterns received from the Android OS on the VM and mobile device.

III. Related Work

Various researches 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 [15] 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 were 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 [14] [15]. 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 [19]. 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 [20] wherein the TCP timestamp was used as a covert channel to reveal the target host's physical clock skew. Meanwhile in [21], discrepancies between two different kinds of timestamp, ICMP and IP in one packet were used to determine the presence of a VM.

In this paper, we extend our scope of studies to explore the distinguishable differences of timestamps pattern between mobile devices that use Android as OS both on mobile device and Android that emulated in VM. Since blocking ICMP timestamps is not available by default on Android platforms, the detection using timestamps pattern could prove to be a vulnerability for the mobile devices [22].

IV. Proposed Detection Method and Methodology

A. *Detection method using Timestamp*

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 as shows in Figure 1.

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. Timestamp is used in various network protocols, such as IP, ICMP and TCP. IP and ICMP timestamp options are variable-length data that are stored in the header and are associated with a particular extension type. One of the options allows the sender to request timestamp values from any target machine which handles the packet by specifying its IP address.

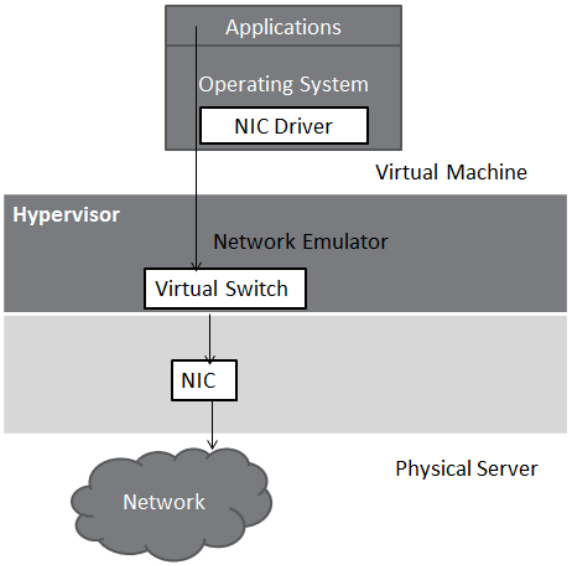

Fig. 1: Virtual network path

While stamping 2 timestamps t_a and t_b in one packet, such as ICMP (t_a) and IP (t_b) timestamps, the timestamps could be deviated because VM might switch operation to another guest operation between the 2 timestamps as shown in Figure 2. This creates time lag of ICMP and IP timestamps in the same packet. In order to verify the scenario of deviated 2 timestamps in 1 packet, we include timestamp request option in the header and send the packet to get timestamps reply. The data structure of the packet with IP header is shown in Figure 3, while the structure of the IP timestamp option packets is shown in Figure 4.

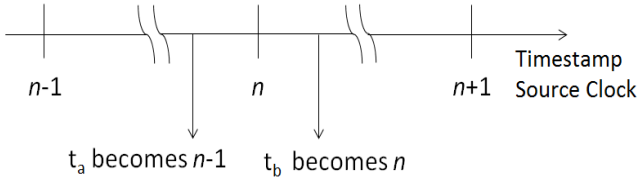

Fig 2: Relationship between source clock and two timestamp operations when timestamp discrepancy occurs

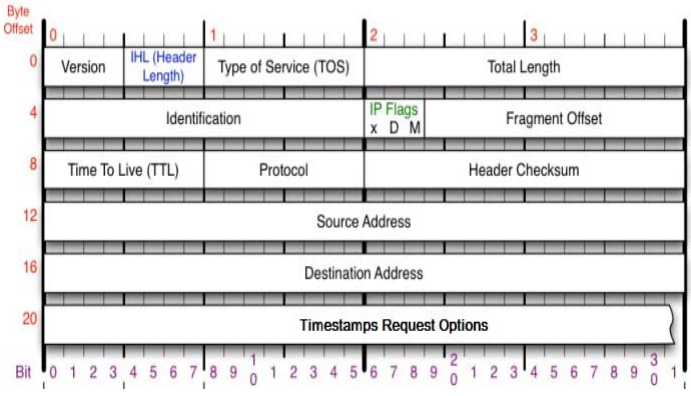

Fig 3: Data structure of IP packets header

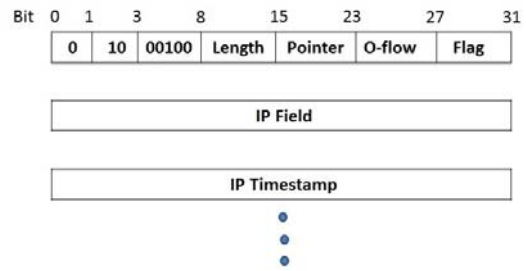

Fig 4: Structure of IP timestamp option

In our previous study [23], we examined timestamps behavior for type 2 VM hypervisors. In the study, we showed that the replied IP timestamp information received from VMs exhibit different behaviors compared the IP timestamps from a real machine (PC). In [24], we proved that the IP timestamp patterns for the type 1 hypervisor also show distinguishable differences between real machines (PC) and VM.

On the other hand, in mobile devices case, timestamp discrepancy could occurs 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 ICMP and IP timestamps for Android OS running on mobile device and VMs are not addressed comprehensively yet. Thus in this research, we conduct tests 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, we predict 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 differentiate.

B. Measurement infrastructure

In this experiment, we sent packets that request both ICMP and IP timestamps from measurer machine to the target machines which includes Android OS operated as emulator in mobile device and VM running Android OS. The experiment environment is shown in Figure 5.

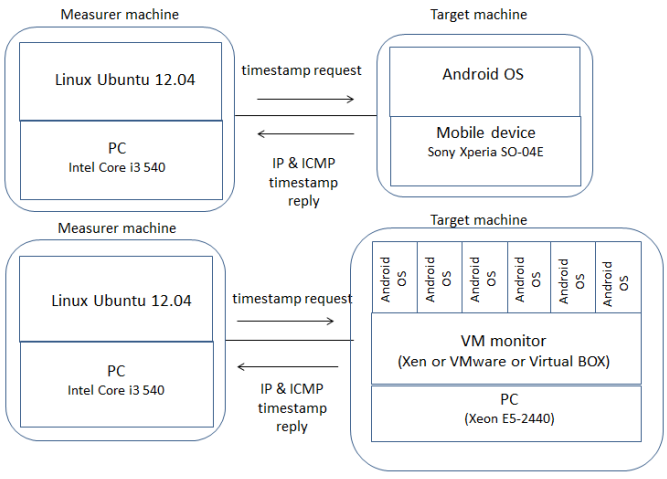

Fig 5: Experimental environment

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 setup to send packets with the timestamp option to the measurement target machines. High-performance Dell Power Edge server with Intel Xeon CPU E5-2440 was used to host the VM target machine. Major hypervisor products, [25], i.e., VMWare[26], Oracle VirtualBox[27] and Xen[28] were implemented in the 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 setup 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. The measurer machine also 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 we setup in our laboratory. C language scripts were written to send packets to request for ICMP and IP packets reply with timestamp option from the client machine to the target machines. We sent non-suspicious packets 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 ICMP and IP 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 [29]. 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. We also examine deviation of ICMP and IP timestamps in 1 packet. From the analyzed data, graphs of the timestamps difference in value, rate for the occurrence and ICMP and IP deviation were plotted to investigate the characteristic pattern differences of ICMP and IP timestamps from each target machine respectively.

C. 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 [20]. Therefore, for our experiments, we used a controlled environment that was setup 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. We did not address the latency issue 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.

V. Result Analysis

We analyzed the collected data to understand the time-stamping pattern behaviors of the target machines. Table 1 shows a sample of a portion of ICMP and IP timestamp data for the 15 count sequence, *n* until the (*n*+1) th – *n* th packet. 1,000,000 ICMP and IP timestamp data were collected from all the target machines. Based from 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 6(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.

Figure 7 shows the compilation of distribution patterns for all 4 tested Android versions.

TABLE I: 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	0
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

Based from 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. While for Android KitKat 4.4.4, and Android Lollipop 5.0.2 the peak is 2, 3 and 4. Based from this results, we could observe that pattern characteristic for 4 versions for Android in Wi-Fi environment are quite similar, where the peak of reoccurrence rate for the difference of timestamps value in the sequence are 2, 3 and 4. Figure 8 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. From Figure 8, we could observe that 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.

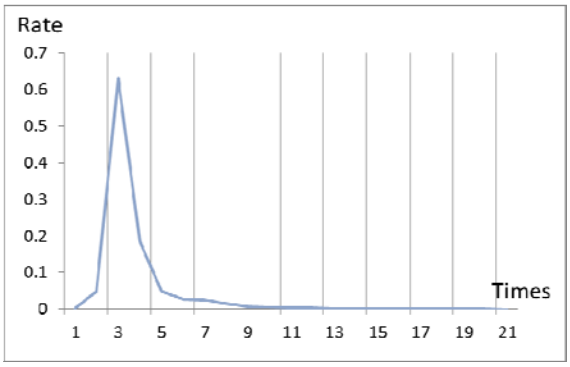

Fig. 6 (a): Android Ice Cream sandwich 4.0.4

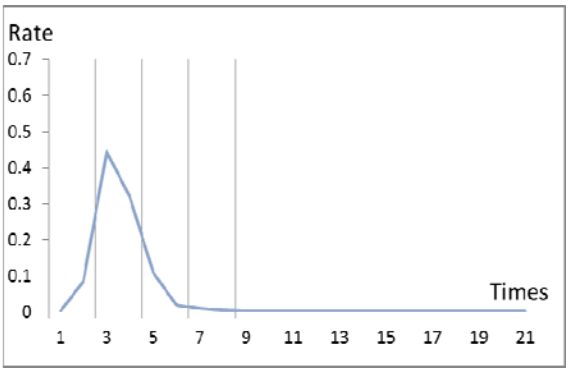

Fig. 6 (b): Android Jelly Bean 4. 2.2

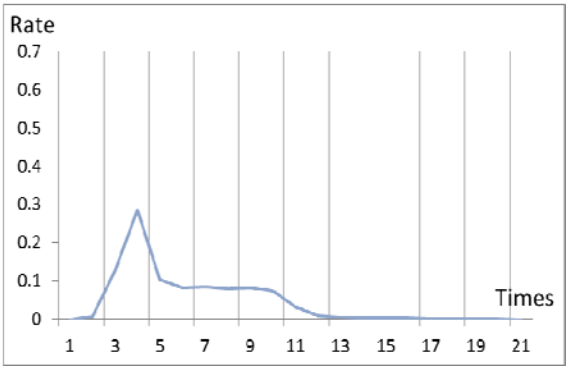

Fig 6 (c): Android KitKat 4.4.4

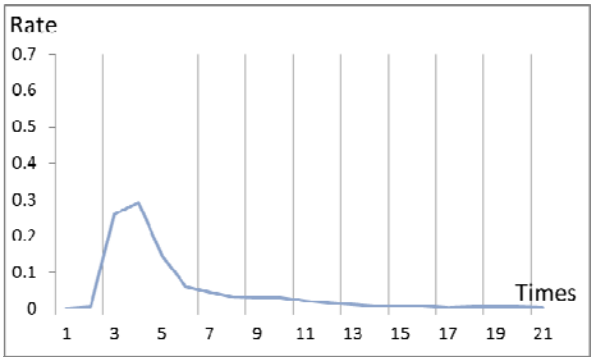

Fig. 6 (d): Android Lollipop 5.0.2

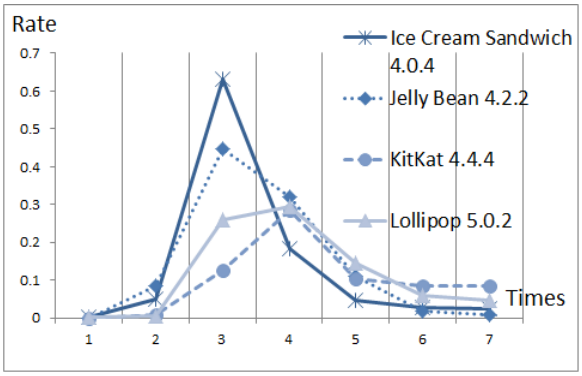

Fig. 7: Timestamps difference distribution for 4 versions of Android

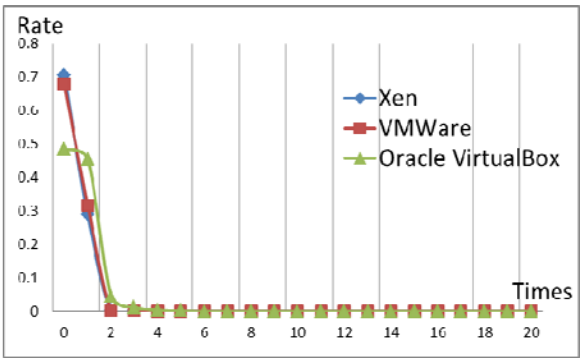

Fig. 8: Timestamps differences when Android installed as emulator on different types of VMs

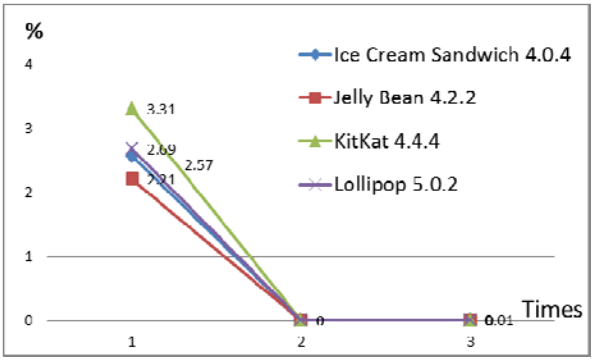

Fig. 9: IP and ICMP timestamps differences for 4 versions of Android

From the distributions graphs in Figure 7 and 8, we could notice a compelling different of the characteristic patterns of ICMP timestamps from Android OS running on mobile device environment and VMs.

Further data analysis also shows that the data for ICMP and IP timestamps from the mobile device replicated the phenomenon as per study completed in [21], where different ICMP and IP timestamps in same packet could be observed. As displayed in Figure 9, 3.31% of the packets from the Android KitKat 4.4.4 give difference value of 1 between the value of ICMP and IP 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.

VI. Discussion

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

The results also showed that ICMP and IP timestamps were deviate 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 ICMP and IP timestamps, we had proved our hypothesis that ICMP and IP 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.

In this research also, we 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.

VII. Conclusion and Future Work

Wireless local area networks (WLANs) or hotspots [30] or commonly known as “Wi-Fi”[31] 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 [32]-[34].

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 [35], [36].

In this research we have shown that ICMP and IP 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 ICMP and IP 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 infected mobile device is connected to corporate internal network, the detection method using ICMP and IP 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 ICMP and IP of the running environment.

In conclusion, from our results in this research, we 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 ICMP and IP timestamp characteristic pattern that was shown in this research.

REFERENCES

[1] N. Leavitt, "Malicious code moves to mobile devices," Computer, 2000(12): p. 16-19.

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

[3] M.R Rahimi, et al., "Mobile Cloud Computing: A Survey, State of Art and Future Directions," *Mobile Networks and Applications*,. vol. 19, no. 2, pp. 133-143. 2014.

[4] S. Zonouz, et al., "Secloud: A cloud-based comprehensive and lightweight security solution for smartphones," *Computer. Security*, vol. 37, pp. 215-227, 2013.

[5] D. Dagon, T. Martin, and T. Starner, "Mobile phones as computing devices: The viruses are coming!" *Pervasive Computing, IEEE*, vol. 3, no. 4, pp. 11-15, 2004.

[6] A. Gostev, and D. Maslenikov, "Mobile malware evolution: An overview," *Kaspersky Labs Report on Mobile Viruses*, 2006.

[7] S. Abraham, and I. Chengalur-Smith, "An overview of social engineering malware: Trends, tactics, and implications," *Technology in Society*, vol. 32, no 3, pp. 183-196, 2010.

[8] J. Hong, "The state of phishing attacks," *Communications of the ACM*, vol. 55, no.1, pp. 74-81, 2012.

[9] J. Oberheide, et al., "Virtualized in-cloud security services for mobile devices," in *ACM Proceedings of the First Workshop on Virtualization in Mobile Computing*, 2008.

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

[11] T. Garfinkel, M.Rosenblum, "A Virtual machine Introspection-Based Architecture for Intrusion Detection", in *Network and Distributed System Security Symposium, The Internet Society*, 2003.

[12] G. Suarez-Tangil, et al., "Evolution, detection and analysis of malware for smart devices," *IEEE Communications Surveys & Tutorials*, vol. 16, no 2, pp. 961-987, 2014.

[13] M. Portnoy, *Virtualization essentials*, John Wiley & Sons, vol. 19, 2012.

[14] T. Garfinkel, K. Adams, A. Warfield, J. Franklin, "Compatibility is not transparency: VMM detection myths and realities," in *Proceedings of the 11th USENIX workshop on Hot topics in operating systems*, USENIX Association: San Diego, CA, pp. 1-6, 2007.

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

[16] T. Raffetseder, C. Kruegel, and E. Kirda, "Detecting system emulators," in *Springer Information Security*, pp. 1-18, 2007.

[17] C. Thompson, M. Huntley, and C. Link, "Virtualization detection: New strategies and their effectiveness". Available: http: Ilwww-users. cs. umn. edu/ cthomp/papers/vmm-detect-20. 1.

[18] H.J. Liao, C-H.R. Lin, Y-C. Lin, K-Y.Tungal, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16-24, 2013.

[19] 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.

[20] T. Kohno, "Remote physical device fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 2, pp. 93-103, 2005.

[21] M. Shimamura, K. Kono, "Remote Virtual Machine Monitor Detection Using Network Timestamp," I*nformation Processing Society of Japan(IPSJ)*, vol. 50, no. 8 (Japanese), pp. 1870-1882, 2009.

[22] M. Cristea, and B. Groza, "Fingerprinting Smartphones Remotely via ICMP Timestamps," *Communications Letters,* IEEE, vol. 17, no. 6, pp. 1081-1083, 2013.

[23] M. Noorafiza, H.Maeda., R. Uda, T. Kinoshita, "Virtual machine remote detection method using network timestamp in cloud computing," in *International Conference on Information Science and Technology (ICIST)*, IEEE, 2013.

[24] M. Noorafiza, H.Maeda., R. Uda, T. Kinoshita, M. Shiratori, "Vulnerability Analysis using Network Timestamps in Full Virtualization Virtual Machine," in *1st International Conference on Information Systems Security and Privacy (ICISSP 2015)*, SCITEPRESS Digital Library, 2015

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

[26] M. Rosenblum, VMWare's Virtual Platform: A virtual machine monitor for commodity PCs. 1999(Hot Chips 11).

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

[28] P. Barham, et al., "Xen and the art of virtualization," in *Proceedings of the nineteenth ACM symposium on Operating systems principles*, ACM: Bolton Landing, NY, USA., pp. 164-177, 2003.

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

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

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

[32] P. Klasnja, et al, "When i am on wi-fi, i am fearless: privacy concerns & practices in eeryday wi-fi use," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2009.

[33] H.S. Choi, and D. Carpenter, "Connecting to Unfamiliar Wi-Fi Hotspots-A Risk Taking Perspective", 2013.

[34] T. Kindberg, et al. "Measuring trust in wi-fi hotspots," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2008.

[35] B. Tokuyoshi, "The security implications of BYOD," "Network Security", vol 4. pp. 12-13, 2013.

[36] G. Thomson, "BYOD: enabling the chaos," *Journal of Network Security*, vol.2, no. 2, pp. 5-8, 2012.