Split Personality Malware Detection and Defeating in Popular Virtual Machines

Anjana V. kumar
Depatment of computer Science
NITK, Surathkal Mangalore
Phone Number: +91 7204856833
anjanavk12@gmail.com

Kalpa Vishnani
Depatment of computer Science
NITK, Surathkal Mangalore
Phone Number: +91 8951800668
kalpavishnani@gmail.com

K. Vinay Kumar
Depatment of computer Science
NITK, Surathkal Mangalore
Phone Number: +91-824-2474000 Extn-3403
vinay@nitk.ac.in

ABSTRACT
Virtual Machines have gained immense popularity amongst the Security Researchers and Malware Analysts due to their pertinent design to analyze malware without risking permanent infection to the actual system carrying out the tests. This is because during analysis, even if a malware infects and destabilizes the guest OS, the analyst can simply load in a fresh image thus avoiding any damage to the actual machine. However, the cat and mouse game between the Black Hat and the White Hat Hackers is a well established fact. Hence, the malware writers have once again raised their stakes by creating a new kind of malware which can detect the presence of virtual machines. Once it detects that it is running on a virtual machine, it either terminates execution immediately or simply hides its malicious intent and continues to execute in a benign manner thus evading its own detection. This category of malware has been termed as 'Split Personality' malware or 'Analysis Aware' malware in the Information Security jargon. This paper aims at defeating the split personality malware in popular virtual machine environment. This work includes first the study of various virtual machine detection techniques and then development of a method to thwart these techniques from successfully detecting the virtual machines-VirtualBox, VirtualPC and VMware.

Categories and Subject Descriptors
D.3.3 [**Security**]: Malware analysis, VM detecting malwares, Defeating split personality malwares.

General Terms
Security

Keywords
malware, analysis aware, split personality, VirtualBox, VirtualPC, VMDetectGuard, pin tool, detection, masking.

1. INTRODUCTION
Malware Analysts are increasingly relying on Virtual Machine Environment (VME), debuggers and sandboxes in their analysis work. Hence attackers and their malicious codes have a significant stake in detecting the presence of these malware analysis tools. Virtualization, by its very nature, creates systems that have different characteristics from the real machines. From a theoretical perspective, any difference between the virtual and the real could lead to a fingerprinting opportunity for attackers. Thus, Malware writers have developed a new class of malware called Analysis Aware Malware or Split Personality Malware. This class detects the presence of malware analysis tools such as Virtual Machines (VM), debuggers and sandboxes and then either terminates execution or hides its malicious nature by executing like a benign application. As a result, a casual malware analyst may misunderstand it as a genuine application. Analysts are hence forced to analyze such VM-aware malware by executing them on physical host, which increases the cost of analysis. For this reason, it is important to have the capability and tools needed to not only detect these VM-detecting malware, but to defeat the detection techniques they use.

The Split personality malwares/programs detect the virtual machine using several methods [1] like hardware fingerprinting, registry checks, existing files etc. Previously efforts mainly focused on detecting the Split Personality malware and once detected they resort to analyzing them on a native machine to bring out their malicious nature. This defeats the purpose of using virtual machines for analysis.

The latest work done in this aspect was mainly concentrated on VMware. Here we have further extended the VMDetectGuard tool [1] to mask the detection of other virtual machines. We present the effective results obtained by means of this tool. According to Gartner research [2] the popular virtual machines in market, as per their market shares are VMware, Virtual PC, and Virtual Box. Hence in the work carried out here, the above mentioned virtual machines are taken into consideration

2. RELATED WORK
The latest work done in masking the detection of virtual machines proposes a method [1] using the PIN API. This work was concentrated on VMware in Windows operating system. This includes detecting the VM detection attempts and further tricking the malware into believing that it is running on a native machine even when it is actually running on VMware. This method takes in a potentially malicious binary and changes the return value of functions that try to detect the presence of the virtual machine. But this method's applicability is limited to VMware and to binaries that do not call any other binaries from within. We have extended this work to include other virtual machines.

Guizani et al.[3] described a method that used dynamic binary instrumentation, to detect the specific instructions used in

SIN'12, October 25-27, 2012, Jaipur, India

instruction check and VMware communication channel test, and also change the output of these instructions so that the attempt to detect the virtual machine monitor fails. But again they concentrated only VMware and also only on two detection mechanisms. This formed the basis of the method proposed by Vaishani et. al. [1].

Carpenter et al.[4] propose two mitigation techniques. They aim at tricking the malware by, changing the configuration settings of the .vmx file present on the host system and, altering the magic value to break the guest-host communication channel. But this results in breaking the communication channel between guest and host, which affects many genuine applications using this channel.

The other work done mainly concentrates on detecting this category of malware. Once they are detected they propose the analysis of the malware in the native machine. Huang et al. [5] proposed a method to retrieve malware behavioural information, in real operation system environment, and then to quickly restore back to the point before analysis so as to analyze another malware sample. The existing malware analysis tools [6] like "Multiple path exploration", "Norman Sandbox" and "Ether" etc. are not capable of masking VM detection.

So, here we conclude that there is no single method that covers different virtual machines, and effectively thwarts VM detection by split personality malware.

3. METHODS OF DETECTION OF VIRTUAL MACHINES

The methods of detection of virtual machines can be generalized into seven,

1. Hardware fingerprinting
2. registry check
3. process and file check
4. memory check
5. timing analysis
6. Communication channel check
7. Invalid instruction check

Among the above listed detection techniques, all except invalid instruction check [8] have been discussed in [1]. This check is specific to VirtualPC by Microsoft. Virtual PC uses a bunch of invalid instructions to allow the interfacing between the native machine and the Virtual PC software, i.e. for guest-host communication. There are certain Opcodes that are invalid in the native machine and will raise an exception. But these Opcodes does not raise any exception in VirtualPC. This method of detection can be used in detecting virtual pc.

4. OUR STUDY ON DETECTION TECHNIQUES

the work done here two virtual machines- VirtualPC and VirtualBox are considered and their detection mechanisms are studied. The virtual machines were decided based on their market shares. This study was then added to the VMware study done in [1]. These are used by the analysis aware malwares to flag the detection of VM.

In table 1 and 2 we have given values obtained in virtual machine as well native machine. The criteria chosen depend on the virtual machine under consideration. Table 1-2 shows few detection techniques and sample values in both VirtualBox and VirtualPC.

5. ALGORITHM TO MASK VM DETECTION

We have used Intel PIN tool [7] for instrumenting each call/instruction made by the binary under test. The steps we have followed for masking VM detection is

I. Create a complete list of all the instructions to be tracked in all the three virtual machines.

We have taken into consideration instructions like,

- __access (File and Process Check)
- LoadLibraryA (File and Process Check)
- Process32Next (File and Process Check)
- __emit (Invalid Opcode Check) etc.

We have prepared a list of instructions which indicate the possible attempt of VM detection. The above listed are a few examples. We have created a complete list of possibly all such instructions after a thorough analysis and research. These instructions are compared against the instruction calls made by the binary under test at the runtime

II. Determine the underlying virtual machine.

The underlying virtual machine is determined to be VMware, VirtualBox, or VirtualPC. This is done so that only the required checks need to be tracked for each virtual machine under consideration

III. Run the binary sample under test.

IV. Determine the OS and the binary as 32/64 bit.

PIN is different for 32 and 64 bit OS as well as for 32 and 64 bit binaries. Hence this has to be determined before deciding which masking method to be invoked.

V. Do dynamic binary instrumentation of the binary under test.

Intel PIN tool provides a large API that can be used for dynamic binary instrumentation. We can view all the instructions called, their arguments, their return values and can also modify these values at runtime.

VI. Provide fake values to instructions revealing the identity of the VM.

Based on the values returned in Step4 the corresponding method is invoked. Each call by the binary is compared with our list of instructions (Step 1). If an instruction match occurs, eg. Say LoadLibraryA() is the instruction that matched, then the argument "LPCTSTR lpFileName" value is checked to see if it is a VM specific library. If so then the argument value is changed to an invalid library name. This function will hence return NULL, and will fail preventing the detection of VM.

This step returns if the binary under test is split personality or not, also each call made is saved, for future analysis.

Similarly other instructions like _emit, Process32Next, STR are also checked, to get their arguments and return value, and if found to be VM detecting instructions a false value is provided such that it appears to be returned from a native machine. Thus this method prevents the detection of virtual machine, as well as tricks the binary into believing that it is running on a native machine.

VII. Provide the results to malware analyst.

The following results are provided at the end of each test:

- If binary is split personality malware or not.
- The log files giving:
  - Entire call trace
  - Criteria for categorizing as split personality.
  - Instructions called that matched the predefined list.

6. IMPLEMENTATION

The criteria and the algorithm explained in the above section are used to check if a binary is possibly split personality malware or not. , We thus made use of these concepts to extend the tool, VMDetectGuard for masking the detection in other virtual machines too. This tool takes in the binary to be tested and provides the user with options to instrument the binary in two different modes, Masking mode and Non-masking mode. The non-masking mode allows running the binary as it is, without any false values being provided. This mode provides only the entire call trace of the binary. i.e. This mode is as good as running the binary in the absence of our tool, just with the difference that we log all the function calls made and the instructions executed by the binary. On the other hand, the masking mode provides false values when the binary tries to detect VM presence, and tells the user if the binary is split personality or not. This mode provides all the log files explained in the previous section. Thus, by running a binary in both the modes, an analyst could to compare the behaviours of the same binary with and without the presence VMDetectGuard.

7. RESULTS

In order to test the effectiveness of our tool, we ran various VM detecting malware samples [9] (both, proof of concept samples and live malware captured from the internet) in the presence as well as absence of the tool, to observe if there were any notable changes in their behaviour. The result of our analysis is shown in Table 3. The screenshots of the proof of concept sample VBDetect in Virtual box, and VMDetect in VirtualPC (both ran with and without the tool) is given in Figure 1-4.

Figure 1. Running VmDetect in VirtualPC

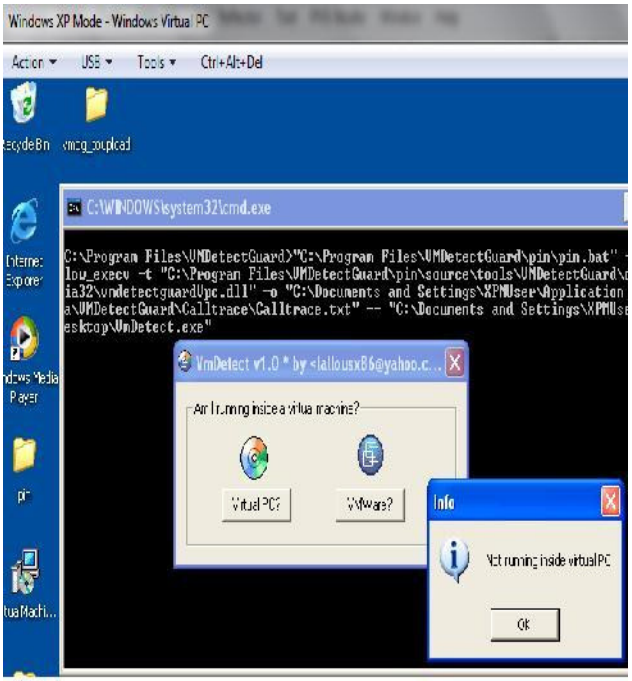

Figure 2. Running VmDetect under masking tool

8. CONCLUSION

We found lack of research in the field of split personality malwares. Malware analysis has been made difficult by the analysis aware malwares as they detect the underlying virtual machine and either behave benignly or do not run. Most of the previous work was concentrated on detection of the split personality malwares. During our study we could not find any full-fledged tool to counter Split Personality malware.

In this paper we present a method that tricks the analysis aware malwares into believing that they are running on a native machine. The tool VMDetectGuard now works with three virtual machines- VirtualBox, VirtualPC and VMware. It works for both 32 bit and 64 bit binaries running on both 32/64 bit operating

systems. We have designed the tool in such a way that it first detects and then masks the VM detection of the binary under test. We have also been successful in masking detection even if the split personality binary is called from within another binary and so on.

Although we have tested VMDetectGuard for several VM Detecting malware, we are still in the testing phase to ensure the completeness of our solution. Moreover, we are currently working on improving the performance of the tool. We are carrying out its performance evaluation to make it more efficient.

Currently our work supports only native binaries in Windows OS, in VirtualBox, VirtualPC and VMware; we are working to support the managed binaries too.

Table 1. Few virtualpc detection techniques

	In VirtualPC	**In Native Machine**
Hardware Fingerprinting		
BIOS	American Megatrenda	L900781
Graphics Card	Virtual PC Integration Components S3 Trio32/64	NVDIA GeForce 310
Baseboard Manufacturer	Microsoft co-orporation	LENOVO
System Name	VIRTUALXP	User-think
USB Controller	USB Virtualisation Bus Driver	Intel® 5 Series /3400 …
Registry Check		
SCSI: HARDWARE\\DEVICEMAP\\Scsi\\Scsi Port 0\\Scsi Bus 0\\Target Id 0\\Logical Unit Id 0	Virtual HD	Hitachi HDS721050CLA362
Control class for usb : SYSTEM\\ControlSet001\\Control\\Class\\{36FC9E60-C465-11CF-8056-444553540000}\\0000	USB Virtualisation Bus Driver	Intel® 5 Series /3400 …
Control class for graphics: SYSTEM\\ControlSet001\\Control\\Class\\{4D36E968-E325-11CE-BFC1-08002BE10318}\\0000	Virtual PC Integration Components S3 Trio32/64	NVDIA GeForce 310
Controlset for cd/dvd drive: SYSTEM\\CurrentControlSet\\Enum\\IDE	Disk Virtual_HD____1._1__	Registry not found
Invalid Opcode	Did not raise exception	Raised exception
File Check		
Vpcubus Driver	Present	Not Present
Vpcgbus Driver	Present	Not Present
Vpcuhub Driver	Present	Not Present

Table 2. Few VirtualBox detection techniques

	Virtual Box running windows	Host Windows Machine
Hardware Fingerprinting		
BIOS	0	L900781
Graphics Card	Virtual Box Graphics Adapter	NVDIA GeForce 310
N/W adapter	AMD PCNET Family PCI Ethernet Adapter	WAN Miniport(SSTP) …
Processor	Null	CPU1
USB Controller	Std Open HCD USB Host Controller	Intel® 5 Series /3400 …
Registry Check		
Dsdt: : HARDWARE\\ACPI\\DSDT	VBOX __	Registry not present
Scsi P0 : HARDWARE\\DEVICEMAP\\Scsi\\Scsi Port 0\\Scsi Bus 0\\Target Id 0\\Logical Unit Id 0	VBOX HARDDISK	Hitachi HDS721050CLA362
Scsi P1: HARDWARE\\DEVICEMAP\\Scsi\\Scsi Port 1\\Scsi Bus 0\\Target Id 0\\Logical Unit Id 0	VBOX CD-ROM	Null
Vedio Bios Version: HARDWARE\\DESCRIPTION\\System\VideoBiosVersion	Oracle VM VirtualBox Version 4.1.2 VGA Bios	Version 70.18.3E.00.05
System Bios Version: HARDWARE\\DESCRIPTION\\System\SystemBiosVersion	VBOX-1	LENOVO-133
Instruction Check		
STR (store task register)	28 0	40 00
File Check		
VBOXHook	Present	Not Present
VBOXTray	Present	Not Present
VBOXService	Present	Not Present

Table 3. VM Detection samples and their behaviours

Binary	Detection Technique Used	Run without tool	Run under tool
Virtual Box			
VBDetect: calls others binaries for individual checks within.	• Registry Check • File and Process Check • Instruction Check	Detected VirtualBox	Did not detect VirtualBox

Rebhip	• Registry Check • File and Process Check	Runs benignly	Runs maliciously
VirtualPC			
VPCDetect: calls others binaries for individual checks within.	• Registry Check • File and Process Check • Invalid Opcode Check	Detected VirtualPC	Did not detect VirtualPC
Backdoor.Win32.SdBot.fmn	• File and Process Check • Invalid Opcode Check	Displays a message, "This application cannot run under a Virtual Machine	Ran maliciously
VMDetect	Invalid Opcode Check	Detects VirtualPC	Does not detect VirtualPC
Trojen.Karsh-252	Invalid Opcode Check	Displays a message, "This application cannot run under a Virtual Machine	Ran Maliciously

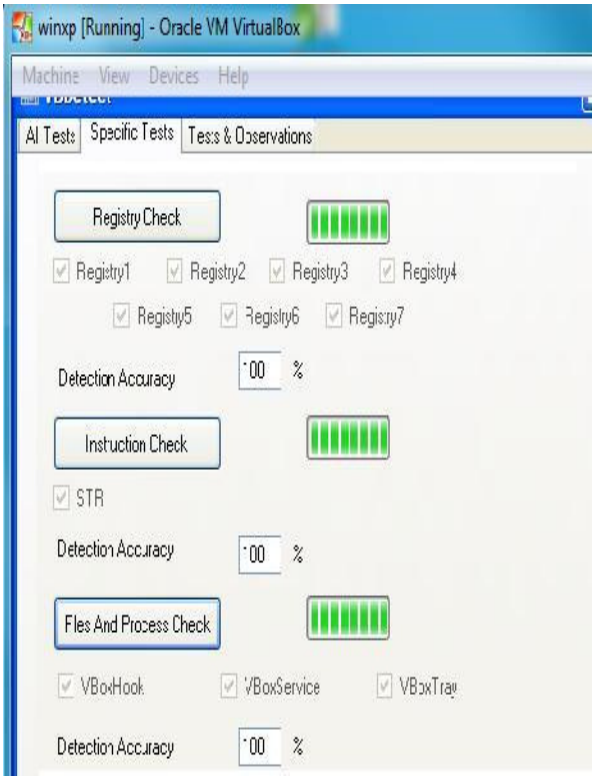

Figure 3. Running detection checks in Virtual Box

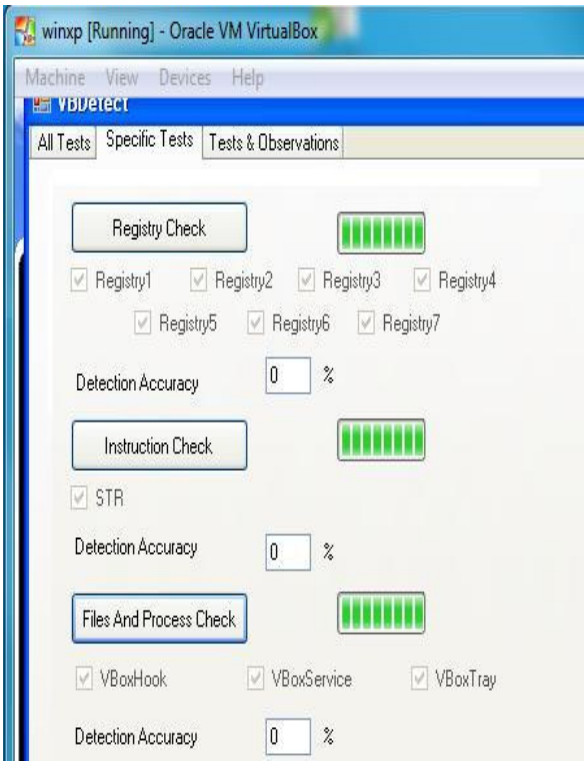

Figure 4. Running detection chechks under the masking tool.

9. REFERENCES

[1] K. Vishnani, A. R. Pais, R. Mohandas National Institute Of Technology Karnataka, India,.. 2011 Detecting & Defeating Split Personality Malware. SECURWARE 2011: The Fifth International Conference On Emerging Security Information, Systems And Technologies.

[2] Gartner research, ID Number G00170437. 2012 [Online]. DOI: http://www.mendeley.com/research/virtual-machines-market-share-through-2012/#page-1.

[3] W. Guizani , J . Y . Marion , and R.Plantey 2009. Server-Side Dynamic Code Analysis. Analysis,.

[4] M. Carpenter, T . Liston, and Skoudis 2007. Hiding Virtualization from Attackers and Malware. IEEE Security and Privacy, June, pp. 62-65.

[5] H D Huang, C. S. Lee, H.Y. Kao, Y.L. Tsai, J.-Gong Chang, 2011 Nat. Center for High- Performance Comput., Nat. Appl. Res. Labs., Tainan, Taiwan Intelligent Agent (IA), Malware behavioral analysis system: TWMAN, 2011 IEEE Symposium on 11-15 April 2011.

[6] M. Egele Vienna University Of Technology, T. Scholte, SAP Research, S. Antipolis, E. Kirda, Institute Eurecom, Sophia Antipolis And C. Kruegel, University Of California, Santa Barbara, A Survey On Automated Dynamic Malware Analysis Techniques And Tools, ACM Computing Surveys.

[7] C.K. Luk, R. Cohn, R.t Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. Janapa Reddi, K. Hazelwood. 2005. Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation, Programming Language Design and Implementation (PLDI), Chicago, IL, June 2005, pp. 190-200.

[8] VmDetect (2005), “Detect if your program is running inside a Virtual Machine - CodeProject” [Online]. Available: http://www.codeproject.com/KB/system/VmDetect.aspx

[9] Virtual machine detection malwares [Online] DOI: http://www.offensivecomputing.net/