A Technique for Remote Detection of VMware Workstation

Christopher Jämthagen and Martin Hell
Department of Electrical and Information Technology
Lund University, Sweden
E-mail: {christopher.jamthagen,martin.hell}@eit.lth.se

I. Introduction and Motivation

Virtualization is a technology often used in testing and analyzing potential malware. Any changes made on the OS by the malware can easily be undone by reverting back to a previously saved snapshot of the system configuration. This will not only make the analysis more efficient, but it is also an easy way of separating the environment exposed to the malware from the rest of the computer. This technique, however, comes with a drawback. If the malware can detect that it is being executed inside a virtual machine, it can adapt to this and simply run harmless code. If it is *not* being executed in a virtual machine, the malware will run the intended malicious code. This will make analysis of the code much more difficult, and in the worst case, malware will not be detected in the analysis. When running in a virtual machine, there are several ways to detect this fact, see e.g., [1]. Since there are many properties of the execution environment that are accessible to the program, it is very difficult to prevent this detection.

In this paper we will take a closer look on remote detection techniques of virtual machines. More specifically, we will try to determine if network traffic originates from a virtual machine by just looking at the packages sent. In a potential attack, a webpage hosted by a web server could attempt to exploit a vulnerability in the browser. The code run on a webpage can be analysed in a virtual machine and if no malware found, the webpage is considered harmless. If the web server can detect that the remote machine running the browser is run in a virtual machine, the webpage can simply choose to not run the exploit. However, if the browser is not run in a virtual machine, the exploit can be launched.

II. VMware and NAT

VMware has many virtulization products and we focus on VMware Workstation, which utilizes a type II hypervisor, meaning that it runs as a regular application ontop of the host operating system. The hypervisor may run several guest operating systems. In our setting, the host will have a private IP address and connects to an external network through Network Address Translation (NAT). Though there are several different variants of NAT, the most commonly used is the Network Address Port Translation (NAPT). In NAPT, the NAT router translates the IP address and the transport identifier (e.g., TCP or UDP port) such that several private hosts can communicate with an external network. The source IP address of outgoing packets is then typically changed to the (external) IP address of the NAT router.

Our remote virtual machine detection technique is applicable if the host/guest combination is either Linux/Windows or Windows/Linux-based. It is based on the following two observations:

- VMware's NAT implementation changes some guest OS specific values in outgoing TCP/IP headers to host OS specific values.
- We have not found this behaviour in other NAT implementations, such as home routers.

The identification field in the IP header (IPid) is used to reassemble fragmented packets. This field is typically set in one of three ways. A global counter can be used, incrementing the value for each packet sent. This is used in Windows operating systems. Another option is to use one counter for each connection, which is implemented by the Linux kernel. A third option is to use a PRNG to set the values. This is used in e.g., OpenBSD. The TTL for a packet is also different depending on the operating system, typically Linux OSes set the TTL to 64 while Windows sets it to 128.

In the VMware Workstation NAT, both the IPid and the TTL characteristics are changed to the characteristic used by the host operating system, even though packets originates from the guest OS. This can be exploited for remote virtual detection by setting up a web server accepting HTTP requests. If there is a discrepancy between the user-agent in the HTTP header, the IPid algorithm and/or the TTL value, we can conclude that one or more fields have been rewritten on the way, and that the likelihood of VMware workstation being used is looking pretty good.

To implement this three things are needed:

- A web server that allows connections on two distinct ports. Two ports are needed to test the IP ID discrepancy problem.
- A custom sniffer that can catch packets destined to the web server and analyze them.
- Some sort of database that can mediate the result of the analysis to the webserver, so that it can take the right decision of what content should be sent to a certain host.

References

[1] Ed Skoudis and Tom Liston. Thwarting virtual machine detection. http://handlers.sans.org/tliston/ThwartingVMDetection_Liston_Skoudis.pdf