**Abstract**

**Introduction (Rachel)**

Virtual Machines simulate a machine (abstract or real) that is typically different from the target machine it is being simulated on. Virtual machines can emulate the [computer architecture](http://en.wikipedia.org/wiki/Computer_architecture) and functions of a real world computer without extra hardware. This power and convenience has caused virtual machine applications to become increasingly more popular. This virtualization, however, may bring some new security risks to the table. Rootkits are a type of malicious software that is activated before the operating system each time a system is booted up. With the rise of virtual machine usage, virtual machine based rootkits (VMBRs) have also come into play. Virtual machine rootkits work by dropping a virtual machine monitor (VMM) underneath an OS installation. Virtual machine rootkits, like regular rootkits, may install hidden files, processes, user accounts, or other malicious data on a system. However, it does this on a separate OS, making it virtually invisible to the target OS and its anti-malware applications. This paper will evaluate some existing virtual machine rootkits and proposed defenses against them. It will also provide new insight on tactics for virtual machine rootkit defense.

**Virtual Machines (Mike)**

A virtual machine (VM) creates a completely isolated operating system on one computer that execute programs just like a physical computer. There are two different ways of creating a VM, or guest, on a host computer, virtualization and emulation. Virtualization is a faster implementation of a VM because it uses the same hardware as the host computer provides. One catch is that the VM’s hardware requirements must match the host’s provided set of chips. Emulation of a VM, on the other hand, adds a layer of indirection and translation that allows for the host to provide any type of chipset. Although this indirection layer helps separate the VM from the hardware of the host, it is significantly slower to its counterpart.

**VMBR Installation (Rachel)**

In order for a virtual machine rootkit to be installed, it must first obtain root privileges – or access to kernel mode. This is not a trivial task, but it can be done by exploiting a remote vulnerability, booting from a bad CD-ROM/DVD, or via malicious software installed with root privileges. Corrupt vendors could also sell this kind of information or exploit it themselves after selling the hardware to a paying customer. This root access allows the attacker to install the VMBR in persistent storage (typically disk). Microsoft used the following tactics in the installation of their subVirt VMBR: “When the target system is Windows XP, we store the VMBR state in the beginning of the first active disk partition. We relocate the data that was in these disk blocks to unused blocks elsewhere on the disk. When the target system is Linux, we disable swapping and use the swap partition to store persistent VMBR state. Both these installation procedures leave most of the target’s data in its original location on disk.” [1]

Root access is also necessary to modify the system boot records, which can allow the VMBR to run before the target OS. This modification can be done during the final stages of system shutdown when it won’t be detected by anti-malware applications that have already shut down. Installation of the rootkit will also happen in this phase of shutdown after the boot sequence has been modified. The boot sequence modification is OS Platform specific. On Windows XP it can be done in a LastChanceShutdown Notification event handler that uses the low-level disk driver to copy the modified boot sequence and bypass the file system layer. On Linux, the boot sequence can be modified in user-mode by changing the shutdown scripts so that the installation code runs after all other applications shut down, before the system shuts down entirely.

The installation places the target OS disk space onto a virtual disk. Once the boot sequence has been modified during the final stages of shutdown and the VMBR is installed, the rootkit will run before the target OS during the next system startup. The VMBR must translate the virtual disk to the equivalent location on physical disk. Microsoft’s subVirt modifies the VMM’s disk virtualization module on Windows VirtualPC and the hard-drive block device in Linux for VMware. At this point the VMBR has a very discrete control over the system.

1. Obtain root privileges/access to kernel mode
2. Install virtual machine rootkit on disk
3. Modify system boot records to ensure rootkit runs before the target OS
   1. This can be done during final stages of shutdown (when modifications to hard disk’s boot blocks won’t be detected by anti-malware applications)
   2. LastChanceShutdown Notification on Windows XP
   3. Modify shut-down scripts so installation runs after all other processes are killed
4. Target system’s disk space now contained in virtual disk, VMM translates address to equivalent location on physical disk  Target System no longer has access to physical disk and rootkit has control of the system

**Forms of Attack (Prashanta)**

1. No interaction with target system
   1. Ex. Spam relays, denial of service zombies, phishing web servers
2. Target system http://start.fedoraproject.org/observers
   1. Logging of hardware-level data via modification of VMM’s device emulation software
   2. Virtual machine introspection
3. Target system modifiers
   1. Ex. Modifying network communication, e-mails, or target applications
   2. Virtual machine introspection

**How a Rootkit Maintains Control Over the System (Mike)**

A rootkit can maintain control over a system by allowing the rootkit administrative privileges, also known as ‘root’ access, to the program. One main attributes of a rootkit is to have the ability to not be detected by either human or OS. There have been cases where a rootkit actually manipulated an antivirus’ code so that it left the antivirus useless. This is what makes a rootkit so dangerous and so persistent. By allowing the rootkit to go undetected and granting itself root access, the rootkit may do whatever it pleases. This type of control is more difficult in a virtual machine environment, but rootkits have adapted.

One of the main downfalls of a VMBR is that it is hosted on a virtual machine. This means that the rootkit is only running while the VM is running. The virtual machine rootkit is also vulnerable during a short period of time in-between powering up the system and when the virtual machine is started. The rootkit can minimize this vulnerability by emulating a system shutdown while it remains in a running state. The rootkit also may try to minimize the number of times a full system power-off occurs.

**Hooksafe or Other Detectors (Prashanta and Rachel)**

Automated detection

**How does the system behave when VMBR is injected(Mike)**

**Forms of Defense(Rachel and Prashanta)**

King et. al[1] have categorized the detection techniques into two types; first where the detection system is running below the VMBR and second where the detection system is running above the VMBR(i.e., within the target system).

1. Security software below the VMBR

Detectors running below VMBR can read physical memory or disk and look for signatures or anomalies that indicate the presence of VMBR, such as a modified boot sequence. One of the ways to gain control below the VMBR is to use secure hardware that can run low-layer security software beneath a VMBR.

The other way around to get control below VMBR is to boot from a safe medium such as CD-ROM, USB drive or network boot server. One problem with this could be that VMBRs can avoid booting from safe medium by emulating system shutdowns and reboots. Thus it would be wise to unplug the system before rebooting. As for detection, it is important to remember that in order to survive physical reboots, persistent storage must be used. If there is abnormal availability of these resources, a VMBR could be installed on the machine. [3]

The third way to gain control below the VMBR is to use a secure VMM. King et. al[1] implemented an enhanced version of secure boot which can prevent VMBR installation. The goal of their secure boot system was to provide attestation for existing boot components such as disk’s master boot loader and also to allow legitimate updates of these components.

2. Run detection within virtual machine on target

The original ideas for detecting a VMBR within the target OS were elementary in nature. These suggestions included looking for excess CPU overhead using clock that can’t be manipulated by rootkit (i.e. a pocket watch) and looking for excess memory usage (although this could be masked by the rootkit). [1] With further investigation, it has been pointed out that existing VMMs value functionality over transparency. [3] Virtual hardware is fundamentally different than underlying physical hardware. Detecting differences such as I/O paravirtualization and the inability to execute non-virtualizable instructions like SIDT, SGDT, and SSL can give away the presence of a VMM and its VMBR from within the target OS running on top of the VMM.

1. Run detection below the virtual machine rootkit, out of its control
   1. Secure hardware (Intel’s LaGrande, AMD’s platform for trustworthy computing, Copilot) allow for development of low-layer security software
   2. Boot from a safe medium, i.e. CD\_ROM,http://start.fedoraproject.org/ USB drive, or network boot server
      1. Note: device must be completely powered off first to avoid rootkit shutdown emulation
   3. Use secure VMM, one which gains control of system before operating system boots. This could perform a check to stop a rootkit from modifying the systems boot sequence.
      1. “Using a secure VMM, we implemented an enhanced version of secure boot which can prevent VMBR installations. The goal of our secure boot system is to provide attestation for existing boot components, such as the disk’s master boot record, the file system’s boot sector, and the OS’s boot loader and also to allow legitimate updates of these components. All attempted updates of these components are verified (by checking the cryptographic signature) before they are allowed to complete. The verification code resides in a separate virtual machine, so it is protected from malicious code running within the guest. We implement this secure boot system using a Virtual PC VMM and a Windows XP guest operating system.” [king06.pdf]

\* The methods described above are often unrealistic and not practiced frequently enough

1. Run detection within virtual machine on target system (if possible)
   1. Look for excess CPU overhead using clock that can’t be manipulated by rootkit
   2. Look for excess memory usage (although this can be masked by the rootkit)

**Keep Searching for Good Papers**