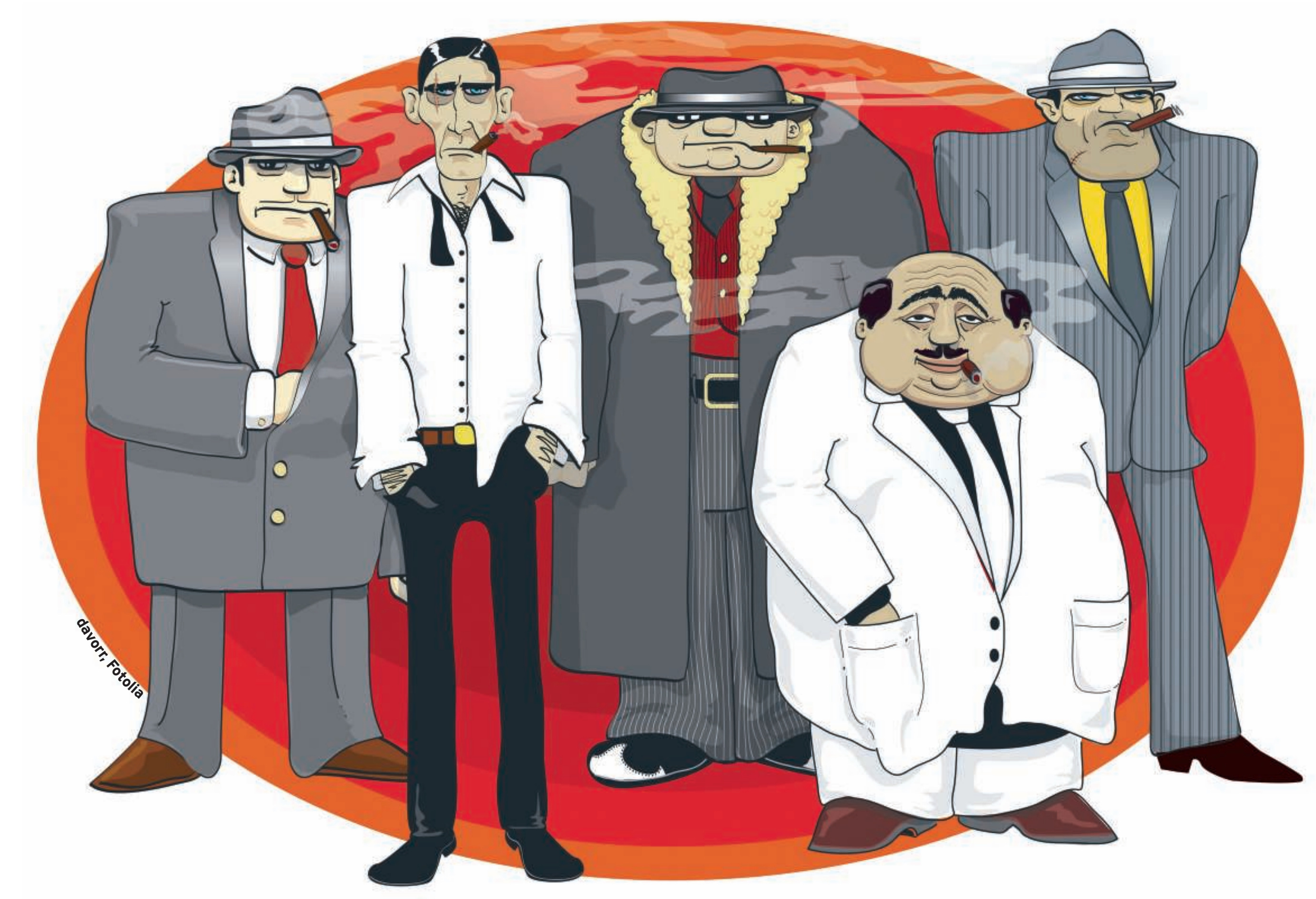


davorr, Fotolia

Virtualizing rootkits and the future of system security

VIRTUAL MALWARE

A new generation of rootkits avoids detection by virtualizing the compromised system – and the user doesn't notice a thing. **BY WILHELM DOLLE AND CHRISTOPH WEGENER**

In the typical cat-and-mouse game of attackers and defenders, the aim of the game is to gain or keep control of the operating system (see Figure 1). Legacy malware tries to escalate privileges and, if possible, to run in ring 0, the operating system's kernel mode. Once it gets there, the exploit, and thus the attacker, can manipulate the system.

Virtualization is often heralded as a big advance for system security. Multiple virtual systems can run on the same hardware without the ability to influence each other. This isolation prevents a number of standard attack techniques, but today's virtualization technologies also open a whole new frontier for attacks that never would have been possible in the past. Experts are already talking about a new generation of rootkits that will exploit the powers of virtualization to avoid detection.

Rootkits let an attacker secretly sustain privileged access to a computer. A rootkit can hide processes, network connections, files, and directories to remotely control the victim's PC, install backdoors, sniff network packets, or log keystrokes. Once the rootkit is running in kernel mode, it can filter and manipulate system call return values and very effectively hide files, directories, and processes.

A rootkit with access to kernel mode can easily terminate applications run in user mode (ring 3) by any normal user, including root. Once it has conquered the kernel, the rootkit is extremely difficult to identify and remove. Of course, the legitimate owner of the computer can also use kernel mode to set up an effective line of defense.

Virtualization essentially acts as another ring with even higher privileges than ring 0. Anyone who compromises the virtualization environment practically controls the whole physical environment on which the system runs. Malware hiding in this layer is even more difficult to discover and to remove than malware in kernel mode.

Researchers at the University of Michigan and from Microsoft Research demonstrated an initial proof of concept

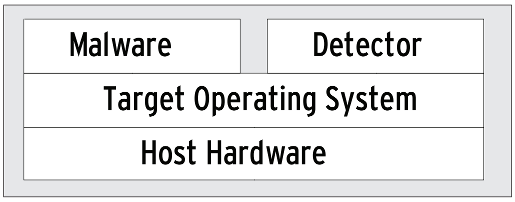


Figure 1: Detection software can only identify malware running at the same (or a higher) level, like the malware and the detector in this figure.

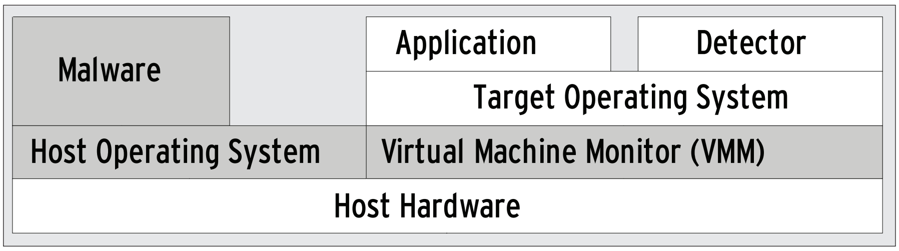


Figure 2: A rootkit that attacks the virtualization layer has wide-ranging privileges. The guest operating system can't terminate or uninstall the software.

rootkit dubbed SubVirt [1] in March 2006, thus spawning the first generation of rootkits to exploit virtualization. After infecting a computer, the rootkit installs itself below the existing system and runs on a virtual machine after rebooting.

To allow this to happen, SubVirt also modifies the boot sequence so that the BIOS no longer loads the Master Boot Record (MBR) belonging to the operating system directly but instead starts a virtual machine. The virtual machine then executes the BIOS and launches the operating system copied into the virtual environment via the MBR.

While users carry on working on their – virtual – operating systems, unaware of what has happened, SubVirt launches a second instance and performs all kinds of nasty tricks. The rootkit cannot be terminated or uninstalled by the guest system because the rootkit controls the virtual machine on which the victim's guest system is running. Security researchers refer to this technique as a virtual machine-based rootkit (VMBR).

Figure 2 shows the new situation; the gray areas are occupied by the rootkit. The attacker's ability to control the victim's system also improves because the rootkit can now use the Virtual Machine Monitor (VMM) to manipulate, forward, or block arbitrary data and hardware characteristics en route to the guest operating system, without leaving the slightest trace of evidence that could be detected by legacy methods.

The researchers demonstrated their ability to compromise both Windows XP and Linux machines, implementing proof of concept attacks with four different vectors, including a phishing web server, a keyboard logger, and spyware that scans the infected system for confidential data.

The technology used by the Windows version of SubVirt is based on Microsoft's Virtual PC software, and the Linux version is based on VMware. However, you do need administrative privileges for the system to install the rootkit, although an attacker could use any number of approaches to gain administrative status.

Discovery

Virtualization technologies such as VMware or Xen are so widespread that discovering an operating system is running in a virtual environment does not necessarily mean you have found a rootkit. Most diagnostic tools demonstrate the existence of the virtual environment on the basis of anomalies. They measure response times, with the assumption that the same command should take longer to complete in a virtual environment than natively – assuming identical hardware and an identical installation. The effect is caused by the virtual machine consuming CPU cycles itself.

This kind of automated timekeeping might be fine to detect a legitimate virtual machine; however, it does not rule out the existence of a rootkit because the rootkit would also control the internal clock. Also, the idea of using external hardware to measure response times manually does not scale very well.

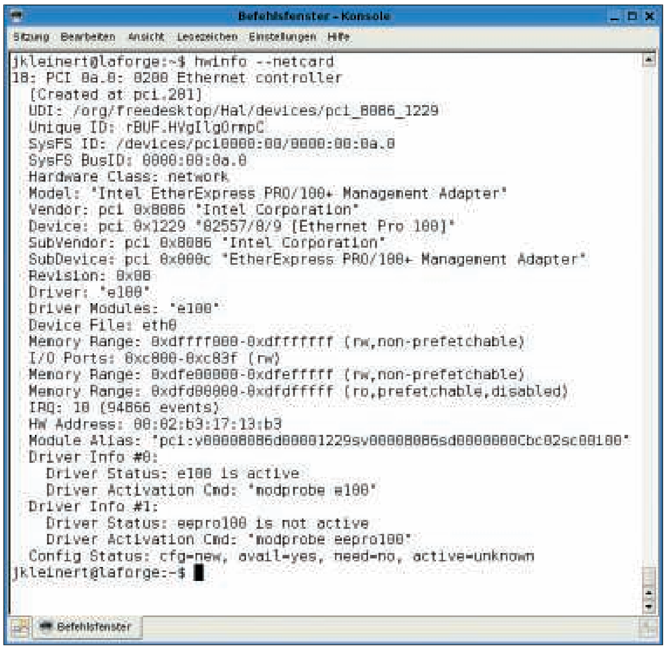
What really gives away an infected system is anomalies in the visible hardware configuration, which are typical for a virtual environment and particularly true of graphics and network adapters. By comparing the known physical configuration with the output from commands like *system-info*, *hwinfo*, or the */proc* filesystem, you can discover the differences (Figure 3). Windows admins will have to use the device manager or third-party tools.

The free disk space, or free memory, could also point to the existence of a virtual environment. For example, if you are unable to use the total physical size of your hard disk, the host system of a virtual machine might need this space itself. But be careful: The host system might also manipulate this data, in that it can control any kind of important output to the guest system.

Boot Externally and Scan the Disk

The rootkit described earlier, SubVirt, resides persistently on the hard disk; however, these changes are very difficult to identify on the running system. To reliably identify an infection, you might need to switch off the machine, boot from a different medium, and analyze the hard disk – of course, this method is problematic for many servers.

Tools developed specifically for this purpose give administrators another approach to detecting the existence of a virtualized operating system. For example, Joanna Rutkowska released Red Pill [2] late in 2004. It works because the SIDT, SGDT, and SLDT instructions exe-



```
jkleinert@laforge:~$ hwinfo --netcard
10: PCI 0a.0: 0200 Ethernet controller
  [Created at pci.281]
  UDI: /org/freedesktop/Hal/devices/pci_8086_1229
  Unique ID: rBUF.HVgIlg0rmpC
  SysFS ID: /devices/pci0000:00/0000:00:0a.0
  SysFS BusID: 0000:00:0a.0
  Hardware Class: network
  Model: "Intel EtherExpress PRO/100+ Management Adapter"
  Vendor: pci 0x8086 "Intel Corporation"
  Device: pci 0x1229 "82557/8/9 [Ethernet Pro 100]"
  SubVendor: pci 0x8086 "Intel Corporation"
  SubDevice: pci 0x000c "EtherExpress PRO/100+ Management Adapter"
  Revision: 0x08
  Driver: "e100"
  Driver Modules: "e100"
  Device File: eth0
  Memory Range: 0xdffff000-0xdfffffff (rw,non-prefetchable)
  I/O Ports: 0xc000-0xc03f (rw)
  Memory Range: 0xdfe00000-0xdfefffff (rw,non-prefetchable)
  Memory Range: 0xdfd00000-0xdfdfffff (ro,prefetchable,disabled)
  IRQ: 10 (94866 events)
  HW Address: 00:02:b3:17:13:b3
  Module Alias: "pci:v00008086d00001229sv00008086sd0000000Cbc02sc00i00"
  Driver Info #0:
    Driver Status: e100 is active
    Driver Activation Cmd: "modprobe e100"
  Driver Info #1:
    Driver Status: eepro100 is not active
    Driver Activation Cmd: "modprobe eepro100"
  Config Status: cfg=new, avail=yes, need=no, active=unknown
jkleinert@laforge:~$ █
```

Figure 3: Tools like "hwinfo" help to find the differences between physical hardware and the hardware the operating system identifies.

cuted by virtual systems return values different from those returned by a native CPU. For example, the SIDT instruction returns the address of the interrupt table.

As an alternative, Tobias Klein's Scoopy doo [3] and Jerry [4] tools will detect a VMware environment. If you are sure that you are running a system without virtualization software, positive findings by these tools are a real indication of an active VMBR.

CPU-Supported

This new generation of virtualizing rootkits might be dangerous, but, as you might expect, this technique also has some weaknesses. For example, the rootkit needs to reboot to become active, and the reboot is easy to detect. Rootkit programmers have cooked up other techniques – some based on the more recent development of hardware-based virtualization.

Either the whole system is virtualized – this is the case with IBM's logical partitions (LPAR [5]), for example – or virtualization is restricted to individual components, such as the processor via Intel VT (Virtualization Technology; formerly Vanderpool [6]) or AMD Virtualization (formerly Pacifica [7]).

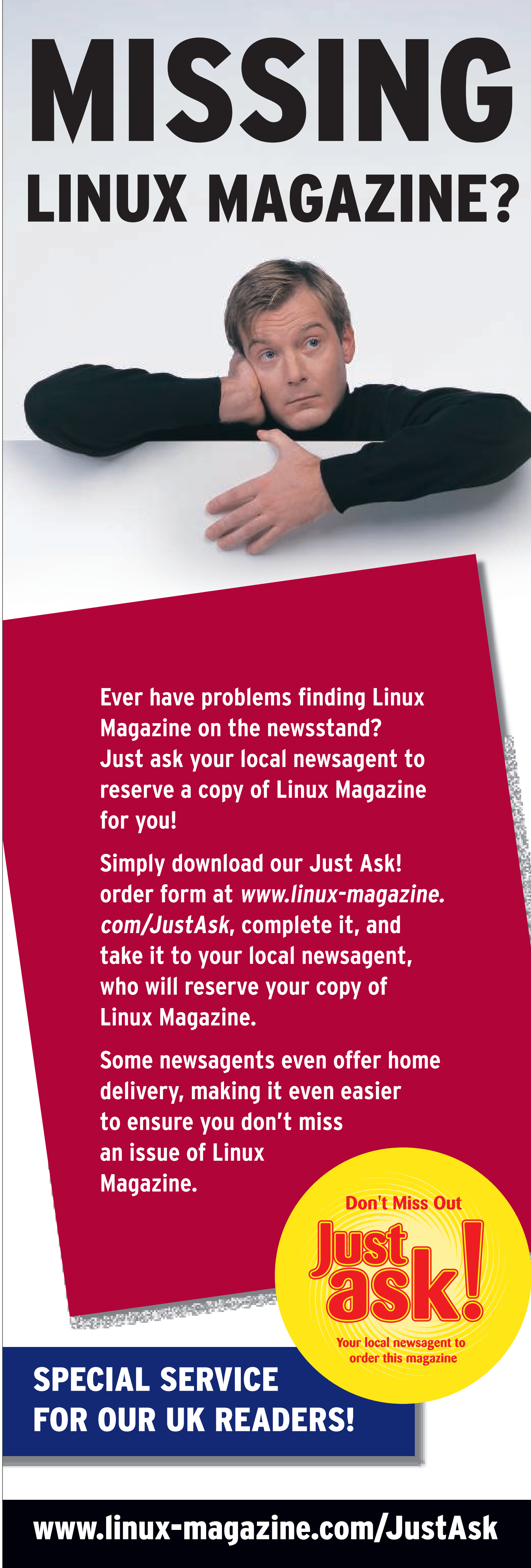
System or operating system virtualization relies on a VMM accepting instructions intended for the hardware from the guest systems (the virtual machines). Without processor support, the VMM has to capture and modify certain ring 0 instructions from the guest system, for example, to protect its own memory management against guest access.

In contrast, AMD and Intel's processor virtualization allows the VMM to send instructions of this kind directly to the processor. The CPU itself takes care of keeping the guest system's and the VMM's processes apart because their logic is inaccessible even to ring 0 processes. The ability to do without modification steps in the VMM helps the system achieve better performance.

Some researchers have started using hardware-based virtualization as a role model for a new generation of rootkits that benefit from the processor technology that allows them to insert an additional hypervisor between the visible hardware and the software. The hypervisor takes control of the system and converts the original operating system into a virtual guest on the fly. In contrast to software-based virtualization, this kind of hijacking does not need a restart, and that makes it all the more difficult to detect the intrusion.

Some rootkits use this kind of nesting technology, such as Blue Pill [8] by Joanna Rutkowska, which was released in 2006 for AMD-V, or Vitriol [9], which is suitable for Intel VT thanks to Dino Dai Zovi. In 2007, Rutkowska and Alexander Tereshkin relaunched Blue Pill, completely rewriting detect and adding a number of features [10]. Once the new Blue Pill is running with administrative privileges, it enables SVM (Secure Virtual Machine) mode on the more recent AMD CPUs and set up the VMCB (Virtual Machine Control Block), which takes control of the infected OS in guest mode.

Until the next reboot, the rootkit itself works at a level below the hypervisor layer. In contrast to SubVirt, Blue Pill does not reside permanently on disk and thus does not survive a cold start. On the other hand, it doesn't leave behind any traces that could



be discovered offline in the course of the forensic investigation.

The Internet is full of bulky and controversial discussions about how easy it is to identify a second-generation rootkit. Of course, command run-time measurements like those described earlier will be less reliable here because the overhead is smaller (or nonexistent) thanks to the host system. Because Joanna Rutkowska originally announced her Blue Pill as "undetectable Malware," people were quick to prove her wrong. Many suggestions as to detecting the rootkit by means of timing analysis were put forward. The approaches at best only managed to confirm that the operating system was running in a virtual environment, which does not necessarily mean a rootkit infection.

INFO

[1] "SubVirt: Implementing Malware with Virtual Machines" by Samuel T. King, Peter M. Chen, Yi-Min Wang, et al. *Proceedings of the 2006 IEEE Symposium on Security and Privacy, http://www.eecs.umich.edu/Rio/papers/king06.pdf*

[2] "Red Pill … Or How To Detect VMM Using (Almost) one CPU Instruction" by Joanna Rutkowska. *http://*
i

[3] Scoopy doo: *http://www.trapkit.de/research/vmm/scoopydoo/index.html*

[4] Jerry: *http://www.trapkit.de/research/vmm/jerry/index.html*

[5] IBM LPAR: *http://en.wikipedia.org/wiki/LPAR*

[6] Intel Virtualization Technology: *http://www.intel.com/technology/platform-technology/virtualization/index.htm*

[7] AMD Virtualization technology: *http://multicore.amd.com/us-en/AMD-Multi-Core/Quad-Core-Advantage/At-Work-AMD-Opteron/Virtualization.aspx*

[8] Blue Pill: *http://theinvisiblethings.blogspot.com/2006/06/introducing-blue-pill.html*

[9] Vitriol: *http://www.theta44.org/software/HVM_Rootkits_ddz_bh-usa-06.pdf*

[10] Blue Pill redesign: *http://bluepillproject.org*

[11] "IsGameOver() Anyone?" by Joanna Rutkowska and Alexander Tereshkin. Invisible Things Lab, 2007, *http://bluepillproject.org/stuff/IsGameOver.ppt*

[12] TCG: *http://www.trustedcomputinggroup.org/home*

[13] The Matrix: *http://www.whysanity.net/monos/matrix3.html*

To avoid discovery, Rutkowska and Tereshkin have additionally developed a program called Blue Chicken [11], which detects timing analyses and temporarily removes the malware from virtual memory, thus preventing any timing anomalies. The race between the tortoise and the hare – that is, the game of hiding and discovering rootkits – is very much in full swing.

Prevention

Faced with the difficulty in identifying a rootkit once it has been installed, the need to safeguard and monitor the boot process and the VMM becomes critical. The admin's best approach is to avoid an infection.

From a technical point of view, the models suggested by the Trusted Computing Group (TCG [12]) seem like a good place to start. The key component is a Trusted Platform Module (TPM), a hardware chip that implements the TCG model on the computer's motherboard. The TPM chip provides cryptographic functions and operations, which are addressable through the BIOS and the operating system, to assess how trustworthy a system is.

On top of this, the TPM chip can be used to perform integrity checks during the boot process. To allow this to happen, special routines for critical system components calculate cryptographic hashes and store them in the TPM chip's platform configuration registers (PCRs). The corresponding evaluation instance compares the hash values calculated with the stored reference values and declares the current system configuration to be valid, permitted, or trustworthy.

Incorrect hash values indicate unauthorized changes to the system and trigger suitable responses from the evaluating instance, such as cancellation of the boot process or kernel panic. If the evaluation and the possible response occur while the system is booting, this is referred to as a secure boot. If this happens later – typically being handled by the operating system – this is referred to as a trusted boot.

These techniques allow the administrator to verify the integrity of the whole system from the boot processes to the VMM. The TPM chip and parts of the BIOS act as a core root of trust for measurement. When the system boots, the TPM chip helps the BIOS verify parts of itself and stores the hash values in the TPM platform configuration registers. After this has happened, the BIOS investigates the master boot record on the boot device and hands over control to the bootloader.

If the bootloader has been instructed to do so, it will carry on measuring values. Suitable targets are the bootloader configuration file, the initial RAM disk, the kernel file, the VMM file, and so on. If this is done consistently, the result is a chain of trust from the BIOS to the VMM. The VMM is the master of all the guest systems and can verify their integrity and respond appropriately, depending on what you want to achieve, without the guest systems influencing the process. This all sounds fine in theory, but taking care of the details can be time consuming.

To achieve a demonstrably secure boot, the reference values for each element in the chain of trust must reside in trustworthy memory. The only place to

NeoWare

The names of the two rootkits mentioned in this article, Red Pill and Blue Pill, are lifted from the movie "Matrix" (1999). In once scene, Morpheus (Laurence Fishburne) gives the hacker Neo (Keanu Reeves) a choice, which he has to take by swallowing either the red or the blue pill. The scene plays in an old high-rise building.

Morpheus: Unfortunately, no one can be told what the Matrix is. You have to see it for yourself. [Bends forwards towards Neo.] This is your last chance. After this, there is no turning back. [In his left hand, Morpheus shows a blue pill.] You take the blue pill and the story ends. You wake in your bed and believe whatever you want to believe. [A red pill is shown in his other hand.] You take the red pill and you stay in Wonderland and I show you how deep the rabbit-hole goes. [Long pause; Neo begins to reach for the red pill.] Remember – all I am offering is the truth, nothing more. [Neo takes the red pill and swallows it with a glass of water.] (Source [13]) store the BIOS' checksum and hash value at the start of the train of trust is in the TPM chip's own NVRAM. Then the BIOS must store the MBR reference values in its own NVRAM, and the bootloader must have the reference values for the bootloader configuration file, and so on – the verifying instance guarantees the trustworthiness of the next link in the chain, including the reference values stored in that link.

Best References

Secure boot imposes some fairly stringent constraints on reference value management. This also means that the admin has to replace the reference values stored in the TPM chip after flashing the BIOS with new firmware, and if the bootloader codes change, the reference value in the BIOS must be changed, and so on. All of these transactions have to be secure, and this is only possible with trustworthy and authorized instances.

If an action fails, the administrator also needs some kind of backup and recovery mechanism to boot the system. And the question of whether users should be able to influence this is another hard nut to crack.

The whole infrastructure is very low level; for example, Intel's VT follows the model with its "Secure Extensions" and "Secure Boot" implementations. Because of technical complexities, production deployment of this feature on PC systems is unlikely to be worthwhile in the near future. Embedded systems, mobile phones, and PDAs, on which reference value management is easier to handle, seem more likely candidates.

A question still remains as to who the origin of trust should be: the owner of the system, the manufacturer, or even a third party? The owner is always in danger of losing control of the system because, say, a rootkit compromises all of these measures, or a manufacturer wants to dictate the software a user can install on the system.

Virtualization is increasingly making inroads into daily server operations. The virtual environment offers many security benefits through isolation and parallel contexts, but virtualization also introduces new vectors for malware.

Initial concepts impressively demonstrate how rootkits can exploit virtualization to hide malicious processes. In the future, more implementations can be expected. So far, none of these state-of-the-art virtualizing rootkits has appeared in the wild, but it makes very good sense to keep an eye on security in the virtual world. ■

THE AUTHORS

Wilhelm Dolle, CISA, CISM, CISSP (*http://www.dolle.net*), is Senior Security Consultant with HiSolutions in Berlin and is a BSI-licensed ISO 27001 basic protection auditor. He has written for periodicals and also serves as a university lecturer.

Christoph Wegener, CISA and CBP, has a Ph.D. in physics and has worked as a freelancer since 1999 for IT security and Open Source/Linux at Wecon.it consulting (*http://www.wecon.net*). He has published various articles and has acted as a technical consultant for expert services for publishing companies.