Editors: David Ahmad, drma@mac.com
Iván Arce, ivan.arce@coresecurity.com

Ghost in the Virtual Machine

The machine does not isolate man from the great problems of nature but plunges him more deeply into them.
—*Antoine de Saint-Exupery*

Iván Arce
Core Security Technologies

Virtual machines (VMs) and virtualization technologies aren't new to the computing world—they've been used for at least 40 years. Recently, the availability of virtualization software for low-cost computer equipment and the promise of both tangible reductions on total cost of ownership and rapid return on investment on virtualization projects have moved many organizations to adopt it as a key component of their IT strategy.

In this installment of Attack Trends, I'll look at this technology trend with an eye toward security and analyze past and present advances in offensive security tools and techniques.

The early days

The origins of VMs go back almost as far as modern computing itself. Born to optimize usage of expensive computing resources and provide users with a fully dedicated and interactive computer system, the early 1960s experiences in time-sharing systems at MIT, IBM, Bolt Beranek and Newman (now BBN Technologies), and the University of California, Berkeley, laid the foundations on which VMs were developed.

Memory protection, segmentation and paging, virtual memory and storage, the implementation of microprocessors with multiple execution modes, and the needs arising from multiple users multi-programming on a single computing system all hastened the development of time-sharing systems, such as the Compatible Time-Sharing System[1] at MIT. In time, this system led to Multics (Multiplexed Information and Computing Service) and eventually to modern-day Unix operating systems (OSs); the same foundational concepts from those early systems (www.bitsavers.org/pdf/mit/lcs/tr/MIT-LCS-TR-003.pdf) led to the creation of the IBM System/360 family of computers in the 1960s[2] and to the System/370 in the early 1970s. Current VM concepts derive from these pioneering systems—in fact, the CP/CMS OS developed at IBM (later reimplemented and renamed VM/370) provided the same functional components found in today's most popular virtualization software. The control program (CP), which controlled allocation and isolation, and managed the computing resources for multiple VMs, is functionally equivalent to today's VM monitors (VMMs), the systems required to implement a generic VM environment. The conversational monitor system (CMS) is a full-fledged single-user OS that supports personal use of a dedicated (virtual) computer, the equivalent of a "guest" OS in a modern VM environment.

Communications between users on different VMs were first accomplished in VM/370 by using an underlying directory structure that the CP managed but were later assimilated into the design goals of a dedicated OS, the remote spooling and communications system (RSCS), which was capable of running on a VM.[3]

In 1981, at the dawn of the PC era, R.J. Creasy provided an insightful account of the origins of VM/370 in IBM's *Journal of Research and Development*[4] and its potential future application for personal computing. Today, the z/VM operating system that runs on zSeries of IBM mainframe is a direct descendant of VM/370.

VM architectures

Successful implementation of virtualization technologies in CP/CMS and other OSs in the '70s prompted researchers to formalize the requirements that computer system architectures must meet to support VMs.

In 1974, Gerald Popek and Robert P. Goldberg[5] presented requirements in terms of three simple properties that outline the essential properties of a VMM:

- *Efficiency.* The VMM shouldn't impose noticeable performance degradation when executing programs in the virtual environment. Popek and Goldberg further translated this to mean that the real processor must directly execute a substantial (dominant) subset of the virtual processor's instruction set without VMM mediation. This rules out emulators, higher-level language interpreters, and other hybrid solutions from the realms of a "true" VM system.
- *Resource control.* A VMM must be able to manage the system's resources, allocating resources to virtualized programs and then reac-

quiring control of them as needed. Virtualized programs shouldn't be able to interfere with each other's resources without the VMM's intervention and must not be able to alter the allocation and management of resources that the VMM uses.
- *Equivalence.* A VMM should provide a virtual environment that's essentially identical to the real one for which existing programs are designed. This translates to a virtual hardware abstraction layer that's undistinguishable from the real hardware. A virtualized program running in the presence of a VMM shouldn't perform differently than when it's run directly on real hardware without a VMM, with two notable exceptions: when it's acceptable to have differences caused by timing or if there's a lack of available resources due to the VMM allocation policy.

To analyze the suitability of the instruction set architecture (ISA) a given system implements, Popek and Goldberg characterized instructions according to how they affect system state:

- *Privileged.* These instructions can only execute when the processor is in one specific "mode" of operation (the authors defined two necessary modes: supervisor and user). Privileged instructions generate a "trap" (or recoverable fault) when they don't execute in the required processor mode.
- *Sensitive.* These instructions attempt to change the allocation of the overall system's resources (control sensitive instructions) or whose behavior or results depend on the configuration of allocated resources in the processor's current mode (behavior-sensitive instructions).

Using this model, which explicitly excluded I/O instructions and devices, the authors defined in their first theorem the requirement that an ISA must meet to be virtualizable:

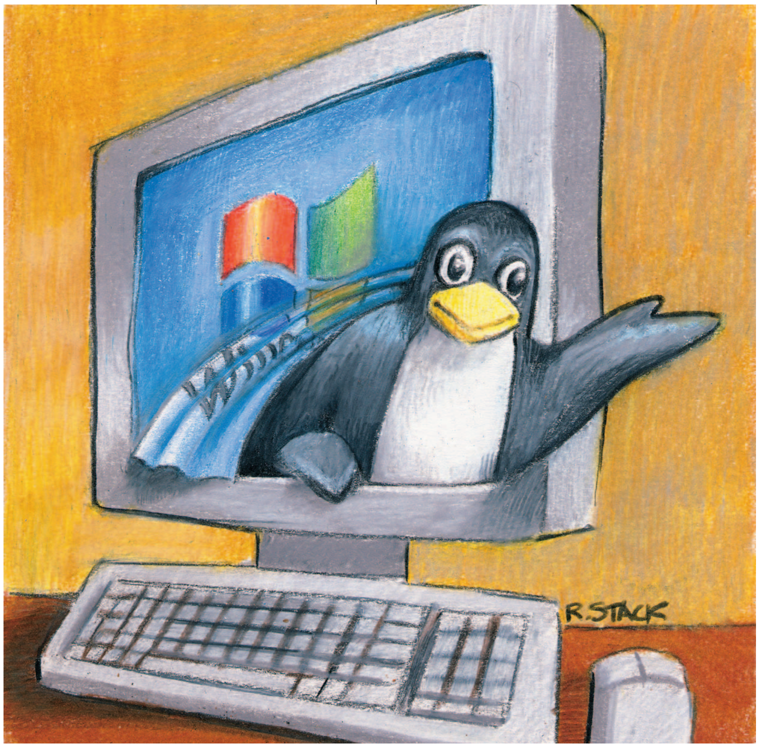
Theorem 1: For any conventional third-generation computer, a VMM may be constructed if the set of sensitive instructions is a subset of the set of privileged instruction.

Computers (or rather, ISAs) that meet this requirement are deemed *virtualizable*. At the time, only a handful of systems could comply with the requirement; the authors identified a specific system, the PDP-11, as unsuitable for virtualization due to its use of memory-mapped I/O devices and I/O instructions.

The trend toward virtualization decayed, and during the 1980s and '90s, when PCs became affordable and UNIX systems dominated in multi-user/multi-programming computer environments, VMs became more of a research curiosity. As a result, Popek and Goldberg's formalization work and its relevance to information security would remain unexplored for decades.

In the mid '90s and the early start of this century, a new wave of virtualization ideas gained steam, driven both by the complexity and mounting operational costs of managing the large number of computers needed for massive parallel processing, and to meet the computing requirements of modern, Internet-borne business organizations. IT-centric organizations that calculated the total cost of ownership identified power consumption, asset management and inventory control, and physical space requirements as factors more relevant to the resulting bottom line than the cost per processing or storage unit. Thus, server consolidation projects through virtualization technology have become increasingly attractive.

Additionally, the demand for software development environments to support multiplatform interoperability and portability also highlighted the advantages of applying virtualization concepts to high-level programming languages, such as Sun Microsystems' Java and Microsoft's Common Language Runtime (CLR) for the .NET framework. The virtualization craze is back and we should



analyze its potential impact on information security.

The Truman Show

The idea of using VMs to mitigate and contain information security threats isn't new. MIT professors Stuart Madnick and John J. Donovan identified the potential use of VMs for security purposes as part of research conducted for Project MAC in 1973.[6] In it, they characterized system security failures as a subset of reliability failures and provided a mostly theoretical probabilistic analysis comparing non-virtual and VM environments.

However, their analysis didn't model purposely malicious programs (or malicious VMs) or profile potential attackers. Although a bit naïve for today's threat landscape, their work is possibly the first analytical approach to security using VMs—it outlined VM advantages and suggested redundancy and independent hardware crosschecks to address weaknesses.

The possibility of achieving effective isolation between users (and their programs) using VMs seems promising to the security field, but the exceptions in Popek and Goldberg's third requirement hint at the potential of covert channels based on timing. Xerox PARC's Butler Lampson pointed this out in a note that appeared in *Communications of the ACM* in 1973[7] and MITRE's Steven Lipner followed up at the ACM Symposium on Operating Systems Principles in 1975.[8] In his note, Lipner proposed addressing the issue of covert channels due to measurable timing differences between virtualized and non-virtualized environments by implementing the notion of a "virtual time," which would isolate each VM's perception of the "real time" of the underlying hardware or of other VMs. Nonetheless, Lipner's conclusions were discouraging:

> "Closing the covert channels seems at a minimum very difficult, and may very well be impossible in a system where physical resources are shared. Ad hoc measures can probably be of value here."

Setting aside possible confinement problems due to covert channels and the limitations imposed by the constraint of sharing finite physical resources, the question of whether a secure VMM can be built using low-cost modern hardware still seems highly relevant.

John Scott Robin, a researcher with the US Air Force, and Cynthia Irvine, at the US Navy's Naval Postgraduate School, sought to find an answer to this question in 2000. Their research studied the feasibility of implementing a secure VMM using the Intel Pentium processor and resulted in the identification of 17 instructions (out of the 250 analyzed) that are sensitive but not privileged, and therefore make the Intel Pentium family of processors fail the criteria for virtualizable ISAs that Popek and Goldberg set 36 years earlier (see www.nps.navy.mil/cs/facultypages/faculty/irvine/Publications/Publications2000/VMM_Usenix00.pdf).

Although the Intel Pentium family (IA-32) isn't "truly" virtualizable, various techniques are available to build virtual environments reasonably close to the ideal of what a VM system should be. Obviously, VMM software running on current Intel Pentium processors must handle the problem of an ISA that has sensitive but unprivileged instructions and the diverse universe of I/O devices and device drivers of modern computers and OSs.

The security-conscious reader might already be asking whether the architectural foundations and the implementation of these virtualization technologies are good enough to withstand a determined attacker or if their weaknesses could turn into security incidents. Let's examine this more closely by way of a modern allegory. The movie *The Truman Show* (www.imdb.com/title/tt0120382/) provides a suitable setting. In it, Truman Burbank, the protagonist, discovers that his life is actually a TV show and that the entire universe known to him is simply a carefully prepared studio set. After realizing this, he embarks on a voyage to break out from the virtual world and into the "real world" outside the TV set.

In the last issue of *IEEE S&P*, Matthew Carpenter, Tom Liston, and Ed Skoudis provided an account of the current techniques used to detect the presence of a VM environment and the possible countermeasures to keep a hypothetical threat source (here, "Truman") unknowingly happy within the confines of his VM environment ("Hiding Virtualization from Attackers and Malware," vol. 5, no. 3, pp. 62–65). They showed the most common techniques for VM detection among the offensive computing crowd. Not surprisingly, these were directly related either to the Intel Pentium processor's failure to provide an instruction set that supports true virtualization (by exploiting unprivileged sensitive instructions) or to implementation problems when handling I/O instructions and devices—something Popek and Goldberg identified as a troublesome area more than three decades ago.

Hax0r The Matrix!

To complement the analysis from Carpenter, Liston, and Skoudis, we need to assess the possibilities that an attacker would succeed with any of these threats:

- subvert the VMM to escape the VM environment and gain control of the computer's resources;
- subvert the VMM to directly or indirectly alter other VMs running concurrently; or
- confine a non-virtualized OS to a VM using virtualization technology, thus gaining and maintaining

control of the computer's resources and remaining undetected to the formerly "real" OS.

A handful of recently disclosed vulnerabilities can give us some insight, but the origins of offensive anti-VM analysis date back to the 1970s when a team at IBM attempted penetration of a VM/370 system.[9] The group reported success: they seemingly completely penetrated the system by exploiting a series of implementation flaws mostly concentrated in the handling of I/O facilities, which led to the realization of the first two threats on the list.

In hindsight, the methodology the group developed for the exercise, their practical approach to the problem, and their clever selection of attacks seem remarkably accurate. The root causes that allowed successful penetration might not be surprising given the current threat landscape and computing environments, but they continue to be valid more than 30 years later. In fact, at this year's CanSecWest Applied Security Conference in Vancouver, Canada, Travis Ormandy, a security researcher at Google, unveiled serious implementation flaws that can lead to the security compromise of all the most popular software packages that provide virtualization capabilities today (http://taviso.decsystem.org/virtsec.pdf).

In 2006, researchers from Microsoft and the University of Michigan demonstrated that the threat of malware that use virtualization technology to defeat detection and analysis is a reality[10] and that realization of the third threat on our list is also possible. In the same year, security researchers Joanna Rutkoswka and Dino Dai Zovi showed similar techniques with malware that uses the technology to support hardware virtualization in the new families of processors from AMD and Intel respectively (https://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Rutkowska.pdf and https://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Zovi.pdf). Their work demonstrates that the realization of the third threat in our list is possible as well.

More information on virtual machines

A comprehensive view of virtual machines (VMs) and taxonomy of VM architectures can be found in the preface to Jim Smith and Ravi Nair's *Virtual Machines: Versatile Platforms for Systems and Processes*, available online at www.cs.uiuc.edu/homes/kingst/spring2007/cs598stk/papers/smith01.pdf. Additionally, a comprehensive survey of virtualization technologies currently in use is available at www.kernelthread.com/publications/virtualization/.

Past and present experiences point to a seemingly invariant conclusion: virtualization continues to be a promising technology to address information security needs, but it will also continue to fall short of delivering on the ideal of a robust, trustworthy, and mythically flawless computing environment.

As the trend toward virtualization accelerates and the technology becomes available to larger user communities at lower costs, the information security community will benefit from the additional hands-on experience to improve security tools and techniques. But malicious attackers certainly won't be spectators in the advancements of those security toolkits. As security practitioners, it's paramount that we adopt a practical and realistic approach to the new wave of virtualization, that we assess the risks with precision, and help align expectations accordingly. □

References

1. F.J. Corbató et al., *The Compatible Time-Sharing System, A Programmer's Guide*, MIT Press, 1963.
2. R.J. Adair et al., "A Virtual Machine System for the 360/40," *IBM Corporation, Cambridge Scientific Center Report 320-2007*, May 1966.
3. E.C. Hendricks and T.C. Hartmann, "Evolution of a Virtual Machine Subsystem," *IBM Systems J.*, vol. 18, no. 1, 1979, pp. 111–142.
4. R.J. Creasy, "The Origin of the VM/370 Time-Sharing System," *IBM J. Research and Development*, vol. 25 no. 5, 1981, pp. 483–490.
5. G.J. Popek and R.P. Goldberg, "Formal Requirements for Virtualizable Third-Generation Architectures," *Comm. ACM*, vol. 17, no. 7, 1974, pp. 412–421.
6. S.E. Madnick and J.J. Donovan, "Application and Analysis of the Virtual Machine Approach to Information System Security and Isolation," *Proc. Workshop on Virtual Computer Systems*, ACM Press, 1973, pp. 210–224.
7. B.W. Lampson, "A Note on the Confinement Problem," *Comm. ACM*, vol. 16 no. 10, 1973, pp. 613–615.
8. S.B. Lipner, "A Comment on the Confinement Problem," *Proc. 5th ACM Symp. Operating Systems Principles* (SOSP 75), ACM Press, 1975, pp. 192–196.
9. C.R. Attanasio, P.W. Markstein, and R.J. Phillips, "Penetrating an Operating System: A Study of VM/370 Integrity," *IBM Systems J.*, vol. 15 no. 1, 1976, pp. 102–116.
10. S.T. King et al. "SubVirt: Implementing Malware with Virtual Machines," *Proc. 2006 IEEE Symp. Security and Privacy*, 2006, IEEE CS Press, pp. 314-327.

***Iván Arce** is chief technology officer and cofounder of Core Security Technologies, an information security company based in Boston. Previously, he worked as vice president of research and development for a computer telephony integration company and as information security consultant and software developer for various government agencies and financial and telecommunications companies. Contact him at ivan.arce@coresecurity.com.*