When Virtual is Harder than Real: Security Challenges in Virtual Machine Based Computing Environments

Tal Garfinkel Mendel Rosenblum
{talg,mendel}@cs.stanford.edu
Stanford University Department of Computer Science

Abstract

As virtual machines become pervasive users will be able to create, modify and distribute new "machines" with unprecedented ease. This flexibility provides tremendous benefits for users. Unfortunately, it can also undermine many assumptions that today's relatively static security architectures rely on about the number of hosts in a system, their mobility, connectivity, patch cycle, etc.

We examine a variety of security problems virtual computing environments give rise to. We then discuss potential directions for changing security architectures to adapt to these demands.

1 Introduction

Virtual machines allow users to create, copy, save (checkpoint), read and modify, share, migrate and roll back the execution state of machines with all the ease of manipulating a file. This flexibility provides significant value for users and administrators. Consequently, VMs are seeing rapid adoption in many computing environments.

As virtual machine monitors provide the same interface as existing hardware, users can take advantage of these benefits with their current operating systems, applications and management tools. This often leads to an organic process of adoption, where servers and desktops are gradually replaced with their virtual equivalents.

Unfortunately, the ease of this transition is deceptive. As virtual platforms replace real hardware they can give rise to radically different and more dynamic usage models than are found in traditional computing environments.

This can undermine the security architecture of many organizations which often assume predictable and controlled change in number of hosts, host configuration, host location, etc. Further, some of the useful mechanisms that virtual machines provide (e.g. rollback) can have unpredictable and harmful interactions with existing security mechanisms.

Virtual computing platforms cannot be deployed securely simply by dropping them into existing systems. Realizing the full benefits of these platforms demands a significant re-examination of how security is implemented.

In the next section we will elaborate on the capabilities that virtual machines provide, new usage models they give rise to, and how this can adversely impact security in current systems. In section 3 we will explore how virtual environments can evolve to meet these challenges. We review related work in section 4 and offer conclusions in section 5.

2 Security Problems in Virtual Environments

A virtual machine monitor (VMM) (e.g. VMware Workstation, Microsoft Virtual Server, Xen), provides a layer of software between the operating system(s) and hardware of a machine to create the illusion of one or more virtual machines (VMs) on a single physical platform. A virtual machine entirely encapsulates the state of the *guest operating system* running inside it.

Encapsulated machine state can be copied and shared over networks and removable media like a normal file. It can also be instantiated on existing networks and requires configuration and management like a physical machine. VM state can be modified like a physical machine, by executing over time, or like a file, through direct modification.

Scaling Growth in physical machines is ultimately limited by setup time and bounded by an organization's capital equipment budget. In contrast, creating a new VM is as easy as copying a file. Users will frequently have several or even dozens of special purpose VMs lying around e.g. for testing or demonstration purposes, "sandbox" VMs to try out new applications, or for particular applications not provided by their regular OS (e.g. a Windows VM running Microsoft Office). Thus, the total number of VMs in an organization can grow at an explosive rate, proportional to available storage.

The rapid scaling in virtual environments can tax the security systems of an organization. Rarely are all administrative tasks completely automated. Upgrades, patch management, and configuration involve a combination of automated tools and individual initiative from administrators. Consequently, the fast and unpredictable growth that can occur with VMs can exacerbate management tasks and significantly multiply the impact of catastrophic events, e.g. worm attacks where all machines should be patched, scanned for vulnerabilities, and purged of malicious code.

Transience In a traditional computing environment users have one or two machines that are online most of the time. Occasionally users have a special purpose machine, or bring a mobile platform into the network, but this is not the common case. In contrast, collections of specialized VMs give rise to a phenomenon in which large numbers of machines appear and disappear from the network sporadically.

While conventional networks can rapidly "anneal" into a known good configuration state, with many transient machines getting the network to converge to a "known state" can be nearly impossible.

For example, when worms hit conventional networks they will typically infect all vulnerable machines fairly quickly. Once this happens, administrators can usually identify which machines are infected quite easily, then cleanup infected machines and patch them to prevent re-infection, rapidly bringing the network back into a steady state.

In an unregulated virtual environment, such a steady state is often never reached. Infected machines appear briefly, infect other machines, and disappear before they can be detected, their owner identified, etc. Vulnerable machines appear briefly and either become infected or reappear in a vulnerable state at a later time. Also, new and potentially vulnerable virtual machines are created on an ongoing basis, due to copying, sharing, etc.

As a result, worm infections tend to persist at a low level indefinitely, periodically flaring up again when conditions are right.

The requirement that machines be online in conventional approaches to patch management, virus and vulnerability scanning, and machine configuration also creates a conflict between security and usability. VMs that have been long dormant can require significant time and effort to patch and maintain. This results in users either forgoing regular maintenance of their VMs, thus increasing the number of vulnerable machines at a site, or losing the ability to spontaneously create and use machines, thus eliminating a major virtue of VMs.

Software Lifecycle Traditionally, a machine's lifetime can be envisioned as a straight line, where the current state of the machine is a point that progresses monotonically forward as software executes, configuration changes are made, software is installed, patches are applied, etc. In a virtual environment machine state is more akin to a tree: at any point the execution can fork off into N different branches, where multiple instances of a VM can exist at any point in this tree at a given time.

Branches are caused by undo-able disks and checkpoint features, that allow machines to be rolled back to previous states in their execution (e.g. to fix configuration errors) or re-run from the same point many times, e.g. as a means of distributing dynamic content or circulating a "live" system image.

This execution model conflicts with assumptions made by systems for patch management and maintenance, that rely on monotonic forward progress. For example, rolling back a machine can re-expose patched vulnerabilities, reactivate vulnerable services, re-enable previously disabled accounts or passwords, use previously retired encryption keys, and change firewalls to expose vulnerabilities. It can also reintroduce worms, viruses, and other malicious code that had previously been removed.

A subtler issue can break many existing security protocols. Simply put, the problem is that while VMs may be rolled back, an attackers' memory of what has already been seen cannot.

For example, with a one-time password system like S/KEY, passwords are transmitted in the clear and security is entirely reliant on the attacker not having seen previous sessions. If a machine running S/KEY is rolled back, an attacker can simply replay previously sniffed passwords.

A more subtle problem arises in protocols that rely on the "freshness" of their random number source e.g. for generating session keys or nonces. Consider a virtual machine that has been rolled back to a point after a random number has been chosen, but before it has been used, then resumes execution. In this case, randomness that must be "fresh" for security purposes is reused.

With a stream cipher, two different plaintexts could be encrypted under the same key stream, thus exposing the XOR of the two messages. This could in turn expose both messages if the messages have sufficient redundancy, as is common for English text. Non-cryptographic protocols that rely on freshness are also at risk, e.g. reuse of TCP initial sequence numbers can allow TCP hijacking attacks [2].

Zero Knowledge Proofs of Knowledge (ZKPK), by their very nature, are insecure if the same random nonces are used multiple times. For example, ZKPK authentication protocols, such as Fiat-Shamir authentication [5] or Schnorr authentication [12], will leak the user's private key if the same nonce is used twice. Similarly, signature systems derived from ZKPK protocols, e.g. the Digital Signature Standard (DSS), will leak the secret signing key if two signatures are generated using the same randomness [1].

Finally, cryptographic mechanisms that rely on previous execution history being thrown away are clearly no longer effective, e.g. perfect forward secrecy in SSL. Such mechanisms are not only ineffective in virtual environments, but constitute a significant and unnecessary overhead.

Diversity Many IT organizations tackle security problems by enforcing homogeneity: all machines must run the most current patched software. VMs can facilitate more efficient usage models which derive benefit from running unpatched or older versions of software. This creates a range of problems as one must try and maintain patches or other protection for a wide range of OSes, and deal with the risk posed by having many unpatched machines on the network.

For example, at many sites today users are simply supplied with VMs running their new operating environment and applications are gradually migrated to that environment, or conversely, legacy applications are run in a VM. This can mitigate the need for long and painful upgrade cycles, but leads to a proliferation of OS versions. This makes patch management more difficult, especially in the presence of older, deprecated versions of operating systems.

Virtual machines have also changed the way that software testing takes place. Previously one required a large number of usually dedicated test machines to test out a new piece of software, one for each different OS, OS version (service pack), patch level, etc. Now each developer or tester can simply have their own collection of virtual test machines. Unfortunately, if these machines are not secured they rapidly become a cesspool of infected machines.

Mobility VMs provide mobility similar to a normal file; they can easily be copied over a network or carried on portable storage media. This can give rise to host of security problems.

For a normal platform, the trusted computing base (TCB) consists of the hardware and software stack. In a VM world, the TCB consists of all of the hosts that a VM has run on. Combined with a lack of history, this can make it very difficult to figure out how far a compromise has extended, e.g. if a file server has been compromised, any VM that was on the server may have been backdoored by an attacker. Determining which VMs were exposed, subsequently copied, etc. can be quite challenging.

Similar problems arise with worms and viruses. Infecting a VM is much like infecting a normal executable. Further, direct infection provides access to every part of a machines state irrespective of protection in the guest OS.

Using VMs as a general-purpose solution for mobility [10, 11] poses even more significant issues. Migrating a VM running on someone's home machine of unknown configuration into a site's security perimeter is a risky proposition at best.

From a theft standpoint, VMs are easy to copy to a remote machine, or walk off with on a storage device. Similar issues of proprietary data loss due to laptop theft are consistently cited as one of the largest sources of financial loss due to computer crime [9].

That VMs are such coarse grain units of mobility can also magnify the impact of theft. Facilitating easy movement of one's entire computing environment (e.g. on a USB keychain) makes users more inclined to carry around all of their (potentially sensitive) files instead of simply the ones they need.

Identity In traditional computing environments there is often an ad-hoc identity associated with a machine. This can be as simple as a list of MAC addresses, employee names, and office numbers. Without such mechanisms it can be extremely difficult to establish who is responsible for a machine, e.g. who to contact if the machine turns malicious or who is responsible for its origin/current state.

Unfortunately, these static methods are impractical for VMs. The dynamic creation of VMs makes the use of MAC addresses infeasible. Often VMs just pick a random MAC address (e.g. in VMware Workstation), in the hope of avoiding collisions.

Identifying machines by location/Ethernet port number is also problematic since a VM's mobility makes it difficult to establish who owns a VM running on a particular physical host. Further, there are often multiple VMs on a physical host, thus shutting off the port to a machine can end up disabling non-malicious VMs as well.

Establishing responsibility is further complicated as VMs have more complicated "ownership histories" than normal machines. A specialized virtual machine may be passed around from one user to the next, much like a popular shell script. This can make it very difficult to establish just who made what changes to get a machine into its present state.

Data Lifetime A fundamental principle for building secure systems is minimizing the amount of time that sensitive data remains in a system [6]. A VMM can undermine this process. For example, the VMM must log execution state to implement rollback. This can undermine attempts by the guest to destroy sensitive data (e.g. cryptographic keys, medical documents) since data is never really "dead," i.e. data can always be made available again within the VM.

Outside the VM, logging can leak sensitive data to persistent storage, as can VM paging, checkpointing, and migration, etc. This breaks guest OS mechanisms to prevent sensitive data from reaching disk, e.g. encrypted swap, pinning sensitive memory, and encrypted file systems.

As a result sensitive files, encryption keys, passwords, etc. can be left on the platform hosting a VM indefinitely. Because of VMs' increased mobility, such data could easily be spread across several hosts.

Similar Problems in Traditional Computing Environments Some existing platforms exhibit security problems similar to those found in virtual environments. Laptops are known for making it difficult to maintain a meaningful network perimeter by transporting worms into internal networks, and sensitive data (e.g. source code) out, thus making the firewall irrelevant. Undo features like Windows Restore introduce many of the same difficulties as rollback in VMs. Transience occurs with dual boot machines, and other occasionally used platforms.

These examples can lend insight into the impact of VMs. However, they differ in a variety of ways. Most of these technologies are deployed in limited parts of IT organizations or see infrequent use; as virtualization is adopted, these dynamic behaviors become the common case. Similar characteristics manifest by other platforms (e.g. mobility, transience) tend to be more extreme in VMs as VMs are software state. Finally, VMs tend to magnify problems with the rapid growth and novel uses they facilitate.

Notably, adapting virtual computing environments to meet these challenges also provides a solution for mobile platforms.

3 Towards Secure Virtual Environments

The dynamic usage models facilitated by virtual platforms demand a dedicated infrastructure for enforcing security policies. We can provide this by introducing a ubiquitous virtualization layer, and moving many of the security and managment functions of guest operating systems into this layer.

Ubiquity allows administrators to flexibly reintroduce the constraints that virtualization relaxes on mobility and data lifetime. Moving security and management functions (e.g. firewalling, virus scanning, backup) from the guest OS to the virtualization layer allows delegation to a central administrator. It also permits management tasks to be automated and performed while VMs are offline, thus aiding issues of usability, scale and transience.

We will briefly outline what such a layer would look like and how it can address the challenges raised in the prior section.

Outlining a Virtualization Layer The heart of a virtualization layer is a high assurance virtual machine monitor. On top of it would run a secure distributed storage system, and components replacing security and management functions traditionally done in the guest OS.

Enforcing policies such as limiting VM mobility and connectivity requires that the virtualization layer on a particular machine be trusted by the infrastructure. Virtualization layer integrity could be verified either through normal authentication and access controls, or through dedicated attestation hardware e.g. TCPA.

Policy at this layer could limit replication of sensitive VMs and control movement of VMs in and out of a managed infrastructure. Document control style policies could prevent certain VMs from being placed onto removable media, limit which physical hosts a VM could reside on, and limit access to VMs containing sensitive data to within a certain time frame.

User and machine identities at this layer could be used to reintroduce a notion of ownership, responsibility and machine history. Tracking information such as the number of machines in an organization and their usage patterns could also help to gauge the impact of potential threats.

Encryption at this layer could help address data lifetime issues due to VM swapping, checkpointing, rollback, etc.

VMM Assurance A VMM's central role is providing secure isolation. The need to preserve this property is sometimes seen as an argument against moving functionality out of the guest operating system. However, such arguments overlook the inherent flexibility available in a VMM. In essence, a virtual machine monitor is nothing more than a microkernel with a hardware compatibility layer. As such, it can support arbitrary protection models for services running at the virtualization layer.

For example, firewall functionality running outside of a guest OS would be hosted in its own protection domain (e.g. a paravirtualized VM), and could utilize a special purpose operating system affording better assurance, greater efficiency, and a more suitable protection model than common OSes.

Other requirements for building a high assurance VMM (e.g. device driver isolation) have been explored elsewhere [7].

3.1 Benefits

Moving security and management functions out of the guest OS provides a variety of benefits including:

- **Delegating Management**

  A virtual environment provides maximum utility when users can focus on using their VMs however they please, without having to worry about managing them.

  Moving security functionality out of guest OSes makes it easier to delegate management responsibilities to automated services and site administrators. It also obviates the need for homogeneous systems where every machine runs a common management suite (e.g. LANDesk), or where an administrator must have an account on every machine.

  As administrators can externally modify VMs, tasks not moved outside of the VM can still be delegated while VMs are offline. Much of the required scanning, patching, configuration, etc. can be done by a service running on the virtualization layer that would periodically scan and maintain archived VMs.

  In a virtualization layer, VMs are first-class objects, instead of merely a collection of bits (as in today's file systems). Thus, operations that today require reconfiguration could be provided transparently, e.g. users should be able to copy VMs just as they would a normal file, without having to bring them online and reconfigure. The infrastructure could appropriately update hostname, cryptographic keys, etc. to reflect the new machine identity.

  Suspended VMs could be executed in a "sandboxed" environment to allow certain configuration changes to anneal and ensure that they do not break the guest.

- **Guest OS Independence**

  Moving security and managment components to the virtualization layer makes them independent of the structure of the guest operating system. Thus, these components can provide greater assurance, as they can largely specify their own software stack and protection model and are isolated from the guest OS. In contrast, today's host-based firewalls, intrusion detection and anti-virus software are tightly coupled with the fragile monolithic operating systems they try to protect, making them trivial to bypass.

  This flexibility opens the door for the adoption of more secure and flexible operating systems as a foundation for infrastructure services. Further, because the infrastructure can now authenticate and trust components running at network end-points, it can now delegate responsibility to these end-points, thus making policies such as trustworthy network quarantine (i.e. limiting network access based on VM contents) feasible.

- **Lifecycle Independence** Moving security relevant state out of the guest OS solves many difficulties caused by rollback.

  This can be accomplished by moving security mechanisms out of the guest completely, into e.g. an external login mechanism, or by modifying guests to store state such as user account information, virus signatures, firewall rules, etc. in dedicated storage that would operate independent of rollback. A combination of both approaches is likely necessary.

  For protocol related issues, making guest software lifecycle independant is likely the easiest path forward, and seems possible without major changes to today's systems.

  As a first step, lifecycle dependent algorithms could be replaced with lifecycle independent variants, e.g. ZKPK based signature schemes (such as DSS) can be replaced with lifecycle independent signatures schemes such as RSA.

  Guest software must have also some way of being notified when a VM has been restarted, so that it can refresh any keys it is currently holding, perhaps a variation on existing approaches for notifying applications when a laptop has awakened from hibernation. Finally, randomness (e.g. data from Linux's `/dev/random`) should be obtained directly from the VMM instead of relying on state/events within the VM.

- **Securely Supporting Diversity**

  A virtual infrastructure should allow users to use old unpatched VMs with diverse OSes much as they would be able to use old or non-standard files without having to change them. This avoids problems such as patches breaking VMs and being unable to secure deprecated versions of software where patches are no longer available.

  Enforcing policy from outside of VMs facilitates this through the use of vulnerability specific protection as an alternative to software modification. For example, vulnerability specific firewall rules, such as Shields [13], can allow users to run unpatched versions of applications and operating systems while still accessing as much network functionality as is safely possible.

  Finally, today greater diversity requires supporting N different versions of security software (e.g. firewall, intrusion detection). While specialized policy is still required for scanning particular OSes, putting management at the virtualization layer eliminates this redundant infrastructure.

There are many challenges to building an architecture that securely allows the full potential of VMs to be realized. However, we believe the direction forward is clear. Moving security relevant functionality out of guest operating system to a ubiquitous virtualization layer provides a more secure and flexible model for managing and using VMs.

4 Related Work

A body of existing work has already examined the security benefits of moving intrusion detection [8], and logging [3, 4] out of the guest e.g. to leverage the isolation provided by a VMM and ability to interpose on all system events. Other work has also examined the benefits of trust and flexible assurance provided by placing components such as the firewall outside of the VM [7].

A variety of recent projects have examined how virtualization can enhance manageability [11], mobility [10] and security [3, 4, 7, 8]. Unfortunately, this work has only considered single hosts or assumed an entirely new organizational paradigm (e.g. utility computing), overlooking how virtual machine technology impacts security in current organizations.

Some of the problems presented here are beginning to be addressed by VMware ACE, such as controlling VM copying, preventing the spread of VM contents (encrypted virtual disks and suspend files) and some support for network quarantine.

In the absence of other published studies on VMs within organizations, we have relied on our own experience and anecdotes from others while formulating this work.

5 Conclusions

We expect end-to-end virtualization to become a normal part of future computing environments. Unfortunately, simply providing a virtualization layer is not enough. The flexibility that makes virtual machines such a useful technology can also undermine security within organizations and individual hosts.

Current research on virtual machines has focused largely on the implementation of virtualization and its applications. We believe that further attention is due to the security risks that accompany this technology and the development of infrastructure to meet these challenges.

Acknowledgments

We are very greatful for generous feedback given to us by Dan Boneh, Jeff Mogul, Armando Fox, Emre Kiciman, Peter Chen, and Martin Casado. This material is based upon work supported in part by the National Science Foundation under Grant No. 0121481.

References

[1] M. Bellare, S. Goldwasser, and D. Micciancio. "Pseudo-random" number generation within cryptographic algorithms: The DDS case. In *CRYPTO*, pages 277–291, 1997.

[2] S. M. Bellovin. Security problems in the TCP/IP protocol suite. *SIGCOMM Comput. Commun. Rev.*, 19(2):32–48, 1989.

[3] P. M. Chen and B. D. Noble. When virtual is better than real. In *(HOTOS-VIII)*, Schloss Elmau, Germany, May 2001.

[4] G. W. Dunlap, S. T. King, S. Cinar, M. A. Basrai, and P. M. Chen. Revirt: Enabling intrusion analysis through virtual-machine logging and replay. In *OSDI*, 2002.

[5] U. Fiege, A. Fiat, and A. Shamir. Zero knowledge proofs of identity. In *STOC '87: Proceedings of the nineteenth annual ACM conference on Theory of computing*, pages 210–217, New York, NY, USA, 1987. ACM Press.

[6] T. Garfinkel, B. Pfaff, J. Chow, and M. Rosenblum. Data lifetime is a systems problem. In *Proc. 11th ACM SIGOPS European Workshop*, september 2004.

[7] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: A virtual machine-based platform for trusted computing. In *Proceedings of the 19th Symposium on Operating System Principles(SOSP 2003)*, October 2003.

[8] T. Garfinkel and M. Rosenblum. A virtual machine introspection based architecture for intrusion detection. In *Proc. Network and Distributed Systems Security Symposium*, February 2003.

[9] L. Gordon, M. L. W. Lucyshyn, and R. Richardson. CSI/FBI computer crime and security survey. `http://www.gocsi.com`, 2004.

[10] M. Kozuch and M. Satyanarayanan. Internet suspend/resume. In *Forth IEEE Workshop on Mobile Computing Systems and Applications*, pages 40–, 2002.

[11] C. Sapuntzakis and M. S. Lam. Virtual appliances in the Collective: A road to hassle-free computing. In *(HOTOS-XI)*, May 2003.

[12] C.-P. Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.

[13] H. J. Wang, C. Guo, D. R. Simon, and A. Zugenmaier. Shield: Vulnerability-driven network filters for preventing known vulnerability exploits. In *Proc. of ACM SIGCOMM*, August 2004.