DOI:10.1145/2593686

Users' trust in cloud systems is undermined by the lack of transparency in existing security policies.

BY MIHIR NANAVATI, PATRICK COLP, BILL AIELLO, AND ANDREW WARFIELD

Cloud Security: A Gathering Storm

FRIDAY, 15:21. Transmogrifica headquarters, Palo Alto...

News has just come in that Transmogrifica has won a major contract from Petrolica to model oil and gas reserves in Gulf of Mexico. Hefty bonuses are on offer if the work is completed ahead of schedule.

Two people are seated in the boardroom.

"Let's order 150 machines right away to speed things along," says Andrea.

"Too expensive," says Robin. "And what will we do with all the machines when we're done? We'll be overprovisioned."

"Doesn't matter," says Andrea. "The bonus more than covers it, and we'll still come out ahead. There's a lot of money on the table."

"What about power? Cooling? And how soon can we get our hands on the machines? It's Friday. Let's not lose the weekend."

"Actually," says Andrea, "Why don't we just rent the machines through the cloud? We'd have things up and running in a couple of hours."

"Good idea! Get Sam on it, and I'll run it by security."

An hour and hundreds of clicks later, Transmogrifica has more than 100 nodes across North America, each costing less than a dollar per hour. Teams are already busy setting up their software stack, awaiting a green light to start work with live data.

Cloud computing has fundamentally changed the way people view computing resources; rather than being an important capital consideration, they can be treated as a utility, like power and water, to be tapped as needed. Offloading computation to large, centralized providers gives users the flexibility to scale available resources with changing demands, while economies of scale allow operators to provide required infrastructure at lower cost than most individual users hosting their own servers.

The benefits of such "utilification"[36] extend well beyond the cost of underlying infrastructure; cloud providers can afford dedicated security and reliability teams with expertise far beyond the reach of an average enterprise. From a security perspective, providers can also

» key insights

- **"Utilification" of computing delivers benefits in terms of cost, availability, and management overhead.**
- **Shared infrastructure opens questions as to the best defenses to use against new and poorly understood attack vectors.**
- **Lack of transparency concerning cloud providers' security efforts and governmental surveillance programs complicates reasoning about security.**

benefit from scale, as they can collect large quantities of data and perform analytics to detect intrusions and other abnormalities not easily spotted at the level of individual systems.

The value of such centralized deployment is evident from its rapid uptake in industry; for example, Netflix migrated significant parts of its management and encoding infrastructure to Amazon Web Services,[12] and Dropbox relies on Amazon's Simple Storage Service to store users' data.[11] Cloud desktop services (such as OnLive Desktop) have also helped users augment thin clients like iPads and Chromebooks with access to remote workstations in data centers.

Virtualization is at the forefront of this shift to cloud-hosted services—a technique for machine consolidation that helps co-locate multiple application servers on the same physical machine. Developed in the 1960s and rediscovered in earnest over the past decade, virtualization has struck a balance between the organizational need to provision and administer software at the granularity of a whole machine and the operational desire to use expensive datacenter resources as efficiently as possible. Virtualization cleanly decouples the administration of hosted software from that of the underlying physical hardware, allowing customers to provision servers quickly and accountably and providers to service and scale their datacenter hardware without affecting hosted applications.

Achieving a full range of features in a virtualization platform requires many software components. A key one is the hypervisor, a special class of operating systems that hosts virtual machines. While conventional OSes present system- and library-level interfaces to run multiple simultaneous applications, a hypervisor presents a hardware-like interface that allows simultaneous execution of many entire OS instances at the same time. The coarser granularity sandboxes provided by hypervisors involve a significant benefit; since virtual machines are analogous to physical machines, administrators can move existing in-house server workloads in their entirety, or the full software stack, dependencies and all, to the cloud, with little or no modification.

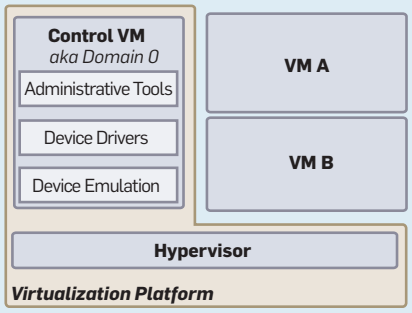
Virtualization has also proved itself an excellent match for various trends in computer hardware over the past decade; for example, increasingly parallel, multicore systems can be partitioned into a number of single- or dual-core virtual machines, so hardware can be shared across multiple users while maintaining isolation boundaries by allowing each virtual machine access to only a dedicated set of processors.

Multiplexing several virtual machines onto a single physical host allows cloud operators to provide low-cost leased computing to users. However, such convenience comes at a price, as users must now trust the provider to "get it right" and are largely helpless in the face of provider failures.

While arguable that such reliance is like relying on third parties for other infrastructure (such as power and network connectivity), there is one crucial difference: Users rely on the provider for both availability of resources and the confidentiality of their data, making critical both the security and the availability of the systems.

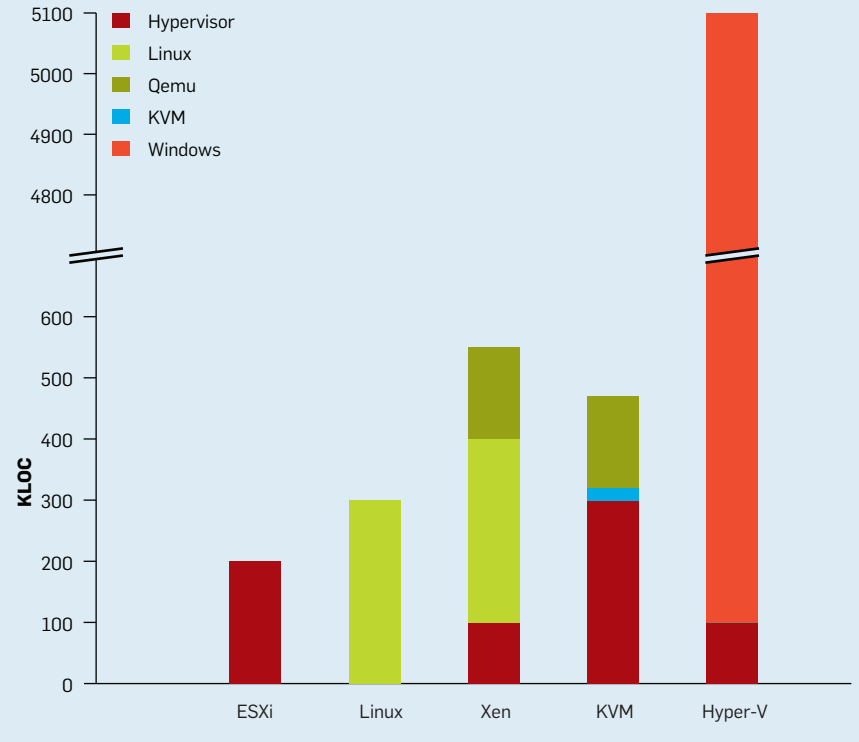
Despite the best effort of cloud providers, unexpected power outages, hardware failures, and software misconfigurations have caused several high-profile incidents[2–4] affecting the availability of large-scale Internet services, including Foursquare, Heroku, Netflix, and Reddit. Unlike outages, however, security exploits are not obvious from the outside and could go undetected for a long time. While the broad reporting of failures and outages is a strong incentive for providers to give clear explanations, there is little incentive to disclose compromises of their systems to their users. Moreover, cloud providers are legally bound to cooperate with law-enforcement agencies in some jurisdictions and may be compelled, often in secrecy, to reveal more about their users' activities than is commonly acknowledged.

Figure 1. Example TCB of a virtualization platform.



In virtualized environments, misbehaving "tenants" on a given machine can try to compromise one another or the virtualization platform itself. As the lowest software layer, responsible for isolating hosted virtual machines and protecting against such attacks, the virtualization platform is the underlying trusted layer in virtualized deployments. The trust customers place in the security and stability of hosting platforms is, to a large degree, trust in the correctness of the virtualization platform itself.

Figure 2. TCB size for different virtualization platforms, from Nova.[29] The Linux kernel size is a minimal system install, calculated by removing all unused device drivers, file systems, and network support.



In traditional nonvirtualized environments, securing systems involves patching and securing the OS kernel, often on a weekly basis. However, virtualized environments expose a larger attack surface than conventional nonvirtualized environments; even fully patched and secured systems may be compromised due to vulnerabilities in the virtualization platform while simultaneously remaining vulnerable to all attacks possible on nonvirtualized systems. OS bugs have been exploited to allow attackers to break process isolation and compromise entire systems, while virtualization-platform bugs risk exposing an opportunity for attackers within one virtual machine to gain access to virtual machines belonging to other customers. Exploits at either of these layers endanger both private data and application execution for users of virtual machines.

Worse, virtualization exposes users to the types of attacks typically absent in nonvirtualized environments. Even without a compromise of the virtualization platform, shared hardware could store sensitive state that is inadvertently revealed during side-channel attacks. Despite attempts by virtualization platforms to isolate hardware resources, isolation is far from complete; while each virtual machine may have access to only a subset of the processors and physical memory of the system, caches and buses are often still shared. Attacks that leak encryption keys and other sensitive data across independent virtual machines via shared caches are being explored.[26,40]

Modern cloud deployments require an unprecedented degree of trust on the part of users, in terms of both the intention and competence of service providers. Cloud providers, for their part, offer little transparency or reason for users to believe their trust is well placed; for example, Amazon's security whitepapers say simply that EC2 relies on a highly customized version of the Xen virtualization platform to provide instance isolation.[6] While several techniques to harden virtualized deployments are available, it is unclear which, if any, are being used by large cloud service providers.

Critical systems and sensitive data are not exclusive to cloud computing. Financial, medical, and legal systems

have long required practitioners comply with licensing and regulatory requirements, as well as strict auditing to help assess damage in case of failure. Similarly, aircraft (and, more recently, car) manufacturers have been required to include black boxes to collect data for later investigation in case of malfunctions. Cloud providers have begun wooing customers with enhanced security and compliance certifications, underlining the increasing need for solutions for secure cloud computation.[5] The rest of this article focuses on a key underpinning of the cloud—the virtualization platform—discussing some of the technical challenges and recent progress in achieving trustworthy hosting environments.

Meanwhile in Palo Alto...

Friday, 15:47. Transmogrifica headquarters, Palo Alto...

An executive enters the boardroom where Robin is already seated.

"Hello, Robin. I hear celebrations are in order. How much time do we have?"

"Hey Sasha, just who I was looking for," Robin says. "It's going to be tight. Andrea was just here, and we thought we'd buy virtual machines in the cloud to speed things up. Anything security would be unhappy about?"

"Well...," says Sasha, "it isn't as secure as in-house. We could be sharing the system with anyone. Literally anyone—who might love for us to fail. Xanadu, for instance, which is sore about not getting the contract? It's unlikely, but it could have nodes on the same hosts we do and start attacking us."

"What would it be able to do?," says Robin.

"In theory, nothing. The hypervisor is supposed to protect against all such attacks. And these guys take their security seriously; they also have a good record. Can't think of anything offhand, but it's frustrating how opaque everything is. We barely know what system it's running or if it's hardened in any way. Also, we're completely in the dark about the rest of the provider's security process. Makes it really difficult to recommend anything one way or the other."

"That's annoying. Anything else I need to know?"

"Nothing I can think of, though let me think it through a bit more."

Trusted Computing Base

The set of hardware and software components a system's security depends on is called the system's trusted computing base, or TCB. Proponents of virtualization have argued for the security of hypervisors through the "small is secure" argument; hypervisors present a tiny attack surface so must have few bugs and be secure.[23,32,33] Unfortunately, it ignores the reality that TCB actually contains not just the hypervisor but the entire virtualization platform.

Note the subtle but crucial distinction between "hypervisor" and "virtualization platform." Architecturally, hypervisors form the base of the virtualization platform, responsible for at least providing CPU multiplexing and memory isolation and management. Virtualization platforms as a whole also provide the other functionality needed to host virtual machines, including device drivers to interface with physical hardware, device emulation to expose virtual devices to VMs, and control toolstack to actuate and manage VMs. Some enterprise virtualization platforms (such as Hyper-V and Xen) rely on a full-fledged commodity OS running with special privileges for the functionality, making both the hypervisor and the commodity OS part of the TCB (see Figure 1). Other virtualization platforms, most notably KVM and VMware ESXi, include all required functionality within the hypervisor itself. KVM is an extension to a full-fledged Linux installation, and ESXi is a dedicated virtualization kernel that includes device drivers. In each case, this additional functionality means the hypervisor is significantly larger than the hypervisor component of either Hyper-V of Xen. Regardless of the exact architecture of the virtualization platform, it must be trusted in its entirety.

Figure 2 makes it clear that even the smallest of the virtual platforms, ESXi,[a] is comparable in size to a stock Linux kernel (200K LOC vs. 300K LOC). Given that Linux has seen several privilege-escalation exploits over the years, justifying the security of the virtualization platform strictly as a function of the size of the TCB fails to hold up.

A survey of existing attacks on virtualization platforms[20,27,37,38] reveals they, like other large software systems, are susceptible to exploits due to security vulnerabilities; the sidebar "Anatomy of an Attack" describes how an attacker can chain several existing vulnerabilities together into a privilege escalation exploit and bypass the isolation between virtual machines provided by the hypervisor.

Reduce Trusted Code?

One major concern with existing virtualization platforms is the size of the TCB. Some systems reduce TCB size by "de-privileging" the commodity OS component; for example, driver-specific domains[14] host device drivers in isolated virtual machines, removing them from the TCB. Similarly, stub domains[30] remove the device emulation stack from the TCB. Other approaches completely remove the commodity OS from the system's TCB,[10,24] effectively making the hypervisor the only persistently executing component of the provider's software stack a user needs to trust. The system's TCB becomes a single, well-vetted component with significantly fewer moving parts.

Boot code is one of the most complex and privileged pieces of software. Not only is it error prone it is also not used for much processing once the system has booted. Many legacy devices commodity OSes support (such as the ISA bus and serial ports) are not relevant in multi-tenant deployments like cloud computing. Modifying the device-emulation stack to eliminate this complex, privileged boot-time code[25] once it has executed significantly reduces the size of the TCB, resulting in a more trustworthy platform.

Prior to the 2006 introduction of hardware support for virtualization, all subsystems had to be virtualized entirely through software. Virtualizing the processor requires modification of any hosted OS, either statically before booting or dynamically through an on-the-fly process called "binary translation." When a virtual machine is cre-

a In 2009, Microsoft released a stripped-down version of Windows Server 2008 called Server Core[23] for virtualized deployments; while figures concerning its size are still not available to us, we do not anticipate the virtualization platform being significantly smaller than ESXi.

Anatomy of an Attack

Not all discovered vulnerabilities are exploitable; in fact, most exploits rely on chaining together multiple vulnerabilities. In 2009, Kostya Korchinsky of Immunity Inc., presented an attack that gave an administrator within a virtual machine running on a VMware hypervisor access to a physical host.[20]

This is notable for two reasons: It affected the entire family of VMware products, so both Workstation and ESX server were vulnerable, and it was reliable enough that Canvas, Immunity's commercially available penetration testing tool, included a "cloudburst" mode to exploit systems and deploy different payloads. Rather than remain an esoteric proof of concept, it was indeed a commercial exploit available to anyone.

The virtualization platform exposes virtual devices to guest machines through device emulation. The device emulation layer runs as a user-mode process within the host, acting as a translation and multiplexing layer between virtual and physical devices. Cloudburst exploited multiple vulnerabilities in the emulated video card interface to allow the guest arbitrary read-and-write access to host memory, giving it the ability to corrupt random regions of memory.

The emulated video card accepts requests from the guest virtual machine through a FIFO command queue and responds to these requests by updating a virtual frame buffer. Both the queue and the frame buffer reside in the address space of the emulation process on the host (`vmware-vmx`) but are shared with the video driver in the guest. The rest of the process's address space is private and should remain inaccessible to the guest at all times.

`SVGA_CMD_RECT_COPY` is an example of a request issued by the driver to the emulator, specifying the (*X*, *Y*) coordinates and dimensions of a rectangle to be copied along with the (*X*, *Y*) coordinates of the destination. The emulated device responds by copying the appropriate regions, indexed relative to the start of the frame buffer. However, due to incorrect boundary checking, the device is able to supply an extremely large or even negative *X* or *Y* coordinate and read data from arbitrary regions of the process's address space. Unfortunately, due to stricter bounds checking around the destination coordinates, arbitrary regions of process memory cannot be written to.

Emulating 3D operations requires the emulated device maintain some device state or contexts. The contexts are stored as an array within the process but are not shared with the guest, which requests updates to the contexts through the command queue. The `SVGA_CMD_SETRENDERSTATE` command takes an index into the context array and a value to be written at that location but does not perform bounds checking on the value of the index, effectively allowing the guest to write to any region of process memory, relative to the context array. This relative write can be further extended by exploiting the `SVGA_CMD_SETLIGHTENABLED` command that reads a pointer from a fixed location within the context and writes the requested value to the memory the pointer references. These two vulnerabilities can be chained to achieve arbitrary memory writes; as the referenced pointer lies within the context array, it is easily modified by exploiting the `SETRENDERSTATE` vulnerability.

When arbitrary reads and writes are possible, shell-code can be written into process memory, then triggered by modifying a function pointer to reference the shell-code. As no-execute protection prevents injected shell-code from being executed, the function pointer must first call the appropriate memory protection functions to mark these regions of memory as executable code pages; when this is done, however, the exploit proceeds normally.

ated, binary translation modifies the instruction stream of the entire OS to be virtualized, then executes this modified code rather than the original OS code.

Virtualizing memory requires a complex page-table scheme called "shadow page tables,"[7] an expansive, extremely complicated process requiring the hypervisor maintain page tables for each process in a hosted virtual machine. It also must monitor any modifications to these page tables to ensure isolation between different virtual machines. Advances in processor technology render this functionality moot by virtualizing both processor and memory directly in hardware.

Some systems further reduce the size of the TCB by splitting the functionality of the virtualization platform between a simple, low-level, system-wide hypervisor, responsible for isolation and security, and more complex, per-tenant hypervisors responsible for the remaining functionality of conventional virtualization platforms.[29,35] By reducing the shared surface between multiple VMs, such architectures help protect against cross-tenant attacks. In such systems, removing a large commodity OS from the TCB presents an unenviable trade-off; systems can either retain the entire OS and benefit from additional device compatibility or remove the entire OS and sacrifice device compatibility by requiring hypervisor-specific drivers for every device.[29]

No matter how small the TCB is made, sharing hardware requires a software component to mandate access to the shared hardware. Being both complex and highly privileged, this software is a real concern for the security of the system, an observation that begs the question whether it is really necessary to share hardware resources at all.

Researchers have argued that a static partitioning of system resources can eliminate the virtualization platform from the TCB altogether.[18] The virtualization platform traditionally orchestrates the booting of the system, multiplexing the virtual resources exposed to virtual machines onto the available physical resources. However, static partitioning obviates the need for such multiplexing in exchange for a loss of flexibility in resource allocation. Partitioning physical CPUs and memory is relatively straightforward; each virtual machine is assigned a fixed number of CPU cores and a dedicated region of memory that is isolated using the hardware support for virtualizing page tables. Devices (such as network cards and hard disks) pose an even greater challenge since it is not reasonable to dedicate an entire device for each virtual machine.

Fortunately, hardware virtualization support is not limited to processors, recently making inroads into devices themselves. Single-root I/O virtualization (SR-IOV)[21] enables a single physical device to expose multiple virtual devices, each indistinguishable from the original physical device. Each such virtual device can be allocated to a virtual machine, with direct access to the device. All the multiplexing between the virtual devices is performed entirely in hardware. Network interfaces that support SR-IOV are increasingly popular, with storage controllers likely to follow suit. However, while moving functionality to hardware does reduce the amount of code to be trusted, there is no guarantee the hardware is immune to vulnerability or compromise.

Eliminating the hypervisor, while attractive in terms of security, sacrifices several benefits that make virtual-

ization attractive to cloud computing. Statically partitioning resources affects the efficiency and utilization of the system, as cloud providers are no longer able to multiplex several virtual machines onto a single set of physical resources. As trusted platforms beneath OSes, hypervisors are conveniently placed to interpose on memory and device requests, a facility often leveraged to achieve promised levels of security and availability.

Live migration[9] involves moving a running virtual machine from one physical host to another without interrupting its execution. Primarily used for maintenance and load balancing, it allows providers to seamlessly change virtual to physical placements to better balance workloads or simply free up a physical host for hardware or software upgrades. Both live-migration and fault-tolerant solutions rely on the ability of the hypervisor to continually monitor a virtual machine's memory accesses and mirror them to another host. Interposing on memory accesses also allows hypervisors to "deduplicate," or remove redundant copies, and compress memory pages across virtual machines. Supporting several key features of cloud computing, virtualization will likely be seen in cloud deployments for the foreseeable future.

This single administrative toolstack is an artifact of the way hypervisors have been designed rather than a fundamental limitation of hypervisors themselves.

Small Enough?

Arguments for trusting the virtualization platform often focus on TCB size; as a result, TCB reduction continues to be an active area of research. While significant progress—from shrinking the hypervisor to isolating and removing other core services of the platform—has been made, in the absence of full hardware virtualization support for every device, the TCB will never be completely empty.

At what point is the TCB "small enough" to be considered secure? Formal verification is a technique to mathematically prove the "correctness" of a piece of code by comparing implementation with a corresponding specification of expected behavior. Although capable of guaranteeing an absence of programming errors, it does only that; while proving the realization of a system conforms to a given specification, it does not prove the security of the specification or the system in any way. Or to borrow one practitioner's only somewhat tongue-in-cheek observation: It "...only shows that every fault in the specification has been precisely implemented in the system."[31] Moreover, formal verification quickly becomes intractable for large pieces of code. While it has proved applicable to some microkernels,[19] and despite ongoing efforts to formally verify Hyper-V,[22] no virtualization platform has been shrunk enough to be formally verified.

Software exploits usually leverage existing bugs to modify the flow of execution and cause the program to perform an unauthorized action. In code-injection exploits, attackers typically add code to be executed via vulnerable buffers. Hardware security features help mitigate such attacks by preventing execution of injected code; for example, the no-execute (NX) bit helps segregate regions of memory into code and data sections, disallowing execution of instructions resident in data regions, while supervisor mode execution protection (SMEP) prevents transferring execution to regions of memory controlled by unprivileged, user-mode processes while executing in a privileged context. Another class of attacks called "return-oriented programming"[28] leverages code already present in the system rather than adding any new code and is not affected by these security enhancements. Such attacks rely on small snippets of existing code, or "gadgets," that immediately precede a return instruction. By controlling the call stack, the attacker can cause execution to jump between the gadgets as desired. Since all executed code is original read-only system code, neither NX nor SMEP are able to prevent the attack. While such exploits seem cumbersome and impractical, techniques are available to automate the process.[17]

Regardless of methodology, most exploits rely on redirecting execution flow in an unexpected and undesirable way. Control-flow integrity (CFI) prevents such an attack by ensuring the program jumps only to predefined, well-known locations (such as functions, loops, and conditionals). Similarly, returns are able to return execution only to valid function-call sites. This protection is typically achieved by inserting guard

conditions in the code to validate any control-flow transfer instructions.[1,13]

Software-based CFI implementations typically rely on more privileged components to ensure the enforcement mechanism itself is not disabled or tampered with; for example, the kernel can prevent user-space applications from accessing and bypassing the inserted guard conditions. However, shepherding the execution of hypervisors through CFI is more of a challenge; as the hypervisor is the most privileged software component, there is nothing to prevent it from modifying the enforcement engine. A possible workaround[34] is to mark all memory as read-only, even to the hypervisor, and fault on any attempted modification. Such modification is verified while handling the fault, and, though benign updates to nonsensitive pages are allowed, any attempt to modify the enforcement engine is blocked. Despite the difficulties, monitoring control flow is one of the most comprehensive techniques to counter code-injection exploits.

Shared Hardware Resources

A hypervisor provides strong isolation guarantees between virtual machines, preventing information leakage between them. Such guarantees are critical for cloud computing; their absence would spell the end for public-cloud deployments. The need for strong isolation is typically balanced against another operational requirement, that providers share hardware resources between virtual machines to provide services at the scale and cost users demand.

Side-channel attacks bypass isolation boundaries by ignoring the software stack and deriving information from shared hardware resources; for example, timing attacks infer certain system properties by measuring the variance in time taken for the same operation across several executions under varying circumstances. Timing attacks on shared instruction caches have allowed attackers to extract cryptographic keys from a co-located victim's virtual machine.[40]

These attacks are conceptually simple: The attacker fills up the i-cache, then waits for the victim to run. The exact execution within the victim's virtual machine determines which cache lines are evicted; the attacker deduces the execution pattern code based on the evicted cache lines and is able to extract the victim's cryptographic key. Moreover, combining such attacks with techniques to exploit cloud placement algorithms[26] could allow attackers to identify victims precisely, arrange to forcibly co-locate virtual machines, and extract sensitive data from them.

While providers have no incentive to undermine their users' operations (their business indeed depends on maintaining user satisfaction), the carelessness or maliciousness of a single, well-placed administrator could compromise the security of an entire system.

Modern hypervisors are helpless to prevent them, as they have no way to partition or isolate the caches, which are often shared between cores on the same processor on modern architectures. While researchers have proposed techniques to mitigate timing attacks (such as randomly delaying requests, adjusting the virtual machine's perception of time and adding enough noise to the computation to prevent information leakage), no low-overhead practically deployable solutions are available. Such mitigation techniques remain an active area of research.

What to Do?

The resurgence of hypervisors is a relatively recent phenomenon, with significant security advances in only a few years. However, they are extremely complex pieces of software, and writing a completely bug-free hypervisor is daunting, if not impossible; vulnerabilities will therefore continue to exist and be exploited.

Assuming any given system will eventually be exploited, what can we do? Recovering from an exploit is so fraught with risk (overlooking even a single backdoor can lead to re-compromise) it usually entails restoring the system from a known good backup. Any changes since this last backup are lost. However, before recovery can begin, the exploit must first be detected. Any delay toward such detection represents a window of opportunity for an attacker to monitor or manipulate the entire system.

Comprehensive logging and auditing techniques are required in several application domains, especially for complying with many of the standards cloud providers aim to guarantee.[5] Broadly speaking, such audit trails have helped uncover corporate impropriety, financial fraud, and even piece together causes of tragic accidents.

For cloud computing, such logs can help identify exactly how and when the system was compromised and what resources were affected.

Tracking information flows between virtual machines and the management tool stack allows logging unauthorized use of highly privileged administrative tools.[15] Not only is such use tracked, the specifics of the interaction are recorded for future audit. If a virtual machine's memory is read, the log stores the exact regions of accessed memory, along with their contents. Users can then assess the effects of the accesses and resolve them appropriately; for instance, if regions corresponding to a password or encryption keys are read, users can change the password or encryption keys before incurring any further damage.

Beyond this, advanced recovery solutions can help recover quickly from security breaches and minimize data loss. Built on top of custom logging engines,[16] they provide analytics to classify actions as either tainted or nontainted. Recovery is now much more fine grain; by undoing all effects of only the tainted actions, an attack can be reversed without losing all useful changes since the last backup. Alternatively, during recovery, all actions, including the attack, are performed against a patched version of the system. The attack will now fail, while useful changes are restored.

Back at Transmogrifica...

Friday, 16:35. Transmogrifica headquarters, Palo Alto...

Sasha enters the boardroom where Robin and Andrea are already seated.

"Robin, Andrea, how confidential is this Petrolica data we'll be processing?"

"Well," says Robin, glancing toward Andrea, "Obviously, it's private data, and we don't want anyone to have access to it. But it isn't medical- or legal-records-level sensitive, if that's what you're getting at. Why do you ask?"

"I hope you realize anyone with sufficient privileges in the cloud provider could read or modify all the data. The provider controls the entire stack. There's absolutely nothing we can do about it. Worse, we wouldn't even know it happened. Obviously I'm not suggesting it would happen. I just don't know the extent of our liability."

"Shouldn't the provider's SOC compliance ensure it's got steps in place to prevent that from happening?," says Andrea before Robin could respond. "Anyhow, I'll run it by legal and see how unhappy they are. We should probably be fine for now, but it's worth keeping in mind for any other projects."

Watching the Watchers

Isolating virtual machines in co-tenant deployments relies on the underlying hypervisor. While securing the hypervisor against external attacks is indeed vital to security, it is not the only vector for a determined attacker. Today's hypervisors run a single management stack, controlled by a cloud provider. Capable of provisioning and destroying virtual machines, the management toolstack can also read the memory and disk content of every virtual machine, making it an attractive target for compromising the entire system.

This single administrative toolstack is an artifact of the way hypervisors have been designed rather than a fundamental limitation of hypervisors themselves. While providers have no incentive to undermine their users' operations (their business indeed depends on maintaining user satisfaction), the carelessness or maliciousness of a single, well-placed administrator could compromise the security of an entire system.

Revelations over the past year indicate several providers have been required to participate in large-scale surveillance operations to aid law-enforcement and counterintelligence efforts. While such efforts concentrate largely on email and social-network activity, the full extent of surveillance remains largely unknown to the public. It would be naïve to believe providers with the ability to monitor users' virtual machines for sensitive data (such as encryption keys) are not required to do so; furthermore, they are also unable to reveal such disclosures to their customers.

Compliance standards also require restricting internal access to customer data while limiting the ability of a single administrator to make significant changes without appropriate checks and balances.[5] As the single toolstack architecture bestows unfettered access to all virtual machines on the administrators, it effectively hampers the ability of operators to provide the guarantees required by their customers, who, in turn, could opt for more private hosting solutions despite the obvious advantages of cloud hosting in terms of scale and security.

Recognizing this danger, some systems advocate splitting the monolithic administrative toolstack into several mini toolstacks[8,10] each capable of administrating only a subset of the entire system. By separating the provisioning of resources from their administration, users would have a private toolstack to manage their virtual machines to a much greater degree than with preprovisioned machines (see Figure 3). As a user's toolstack can interpose on memory accesses from only the guests assigned to it, users' can encrypt the content of their virtual machines if desired. Correspondingly, platform administrators no longer need rights to access the memory of any guest on the system, limiting their ability to snoop sensitive data.

"Nested virtualization," which allows a hypervisor to host other hypervisors in addition to regular OSes, provides another way to enforce privacy for tenants; Figure 4 outlines a small, lightweight, security-centric hypervisor hosting several private, per-tenant, commodity hypervisors.[39] Isolation, security, and resource allocation are separated from resource management. Administrators at the cloud provider manage the outer hypervisor, allocating resources managed by the inner hypervisors. The inner hypervisors are administered by the clients themselves, allowing them to encrypt the memory and disks of their systems without sacrificing functionality. Since device management and emulation are performed by the inner hypervisor, the outer, provider-controlled, hypervisor never needs access to the memory of a tenant, thereby maintaining the tenant's confidentiality.

While both split toolstacks and nested virtualization help preserve confidentiality from rogue administrators, the cloud provider itself remains a trusted entity in all cases. After all, an operator with physical access to the system could simply extract confidential data and encryp-

tion keys from the DRAM of the host or run a malicious hypervisor to do the same. While the former is a difficult, if not impossible, problem to solve, recent advances help allow users to gain some assurance about the underlying system.

Trust and Attestation

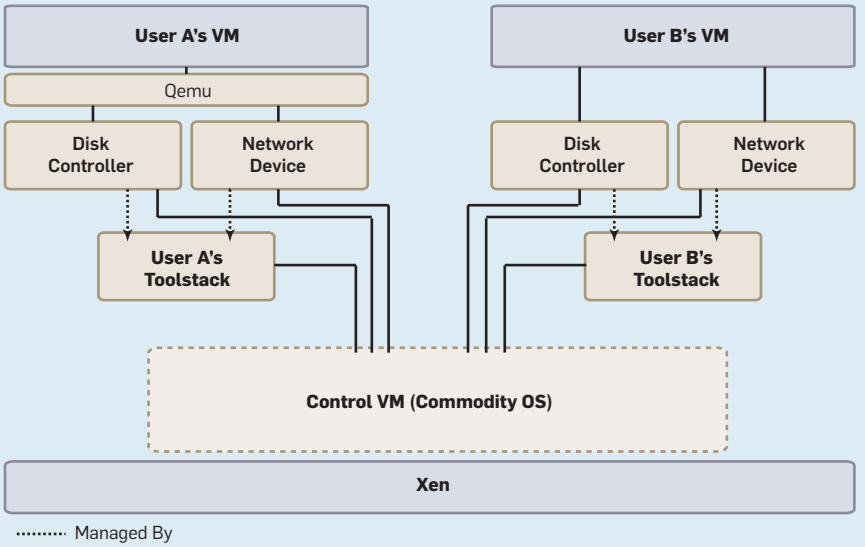
Consider the following scenario commonly seen in fiction: Two characters meet for the first time, with one, much to the surprise of the other, seeming to act out of character. Later, it becomes apparent that a substitution has occurred and the character is an imposter. This illustrates a common problem in security, where a user is forced to take the underlying system at its word, with no way of guaranteeing it is what it claims to be. This is particularly important in cloud environments, as the best security and auditing techniques are worthless if the platform disables them.

"Trusted boot" is a technology that allows users to verify the identity of the underlying platform that was booted. While ensuring the loaded virtualization platform is trusted, it makes no further guarantees about security; the system could have been compromised after the boot and still be verified successfully. Two techniques that provide a trusted boot are unified extensible firmware interface (UEFI) and trusted platform module (TPM). While differing in implementation, both rely on cryptographic primitives to establish a root of trust for the virtualization platform.

UEFI is a replacement for the BIOS and is the first software to be executed during the boot process. In a secure boot, each component of the boot process verifies the identity of the next one by calculating its digest or hash and comparing it against an expected value. The initial component requires a platform key in the firmware to attest its identity. TPM differs slightly in its execution; rather than verify the identity of each component while booting, a chain with the digest of each component is maintained in the TPM. Verification is deferred for a later time. Clients can verify the entire boot chain of a virtualization platform, comparing it against a known-good value, a process called "remote attestation."

Trusted-boot techniques give users a way to gain more concrete guarantees about the virtualization platform to which they are about to commit sensitive data. By providing a way to verify themselves, then allowing users to proceed only if they are satisfied that certain criteria are met, they help overcome one of the key concerns in cloud computing. However, this increased security comes at a cost; to realize the full benefit of attestation, the source code, or at least the binaries, must be made available to the attestor. While cloud providers have a strong incentive to bolster the confidence of their customers, such transparency conflicts with an operational desire to maintain some degree of secrecy regarding the software stack for competitive reasons.

Figure 3. Multiple independent toolstacks.



At the End of a Long Day...

Friday, 21:17. Transmogrifica headquarters, Palo Alto...

Robin and Andrea are reflecting on their long day in the boardroom.

"Well, we got green lights from security and legal. Sam's team just called to say they've got it all tested and set up. We should be starting any time now," says Andrea.

"The cloud scares me, Andrea. It's powerful, convenient, and deadly. Before we realize it, we'll be relying on it, without understanding all the risks. And if that happens, it'll be our necks. But it's been right for us this time around. Good call."

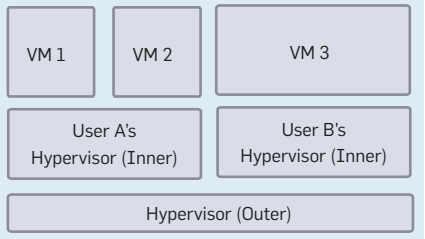
Epilogue. Transmogrifica completed the contract ahead of schedule with generous bonuses for all involved. Centralized computation was a successful experiment, and Transmogrifica today specializes in tailoring customer workloads for faster cloud processing. The 150-node cluster server remains unpurchased.

Conclusion

Cloud computing, built with powerful servers at the center and thin clients at the edge, is a throwback to the mainframe model of computing. Seduced by the convenience and efficiency of such a model, users are turning to the cloud in increasing numbers. However, this popularity belies a real security risk, one often poorly understood or even dismissed by users.

Cloud providers have a strong incentive to engineer their systems to

Figure 4. Nested virtualization.



provide strong isolation and security guarantees while maintaining performance and features. Their business case relies on getting this combination of trade-offs right. However, they have an equally strong incentive to keep their software and management techniques proprietary (to safeguard their competitive advantages) and not report bugs or security incidents (to maintain their reputations). While hypervisors have been studied extensively, and many security enhancements have been proposed in the literature, the actual techniques used or security incidents detected in commercial deployments are generally not shared publicly.

All this makes it difficult for users to evaluate the suitability of a commercial virtualization platform for an application. Given the overall profitability and growth of the business, many clients have sufficient trust to run certain applications in the cloud. Equally clear is that concern over security and liability is holding back other clients and applications; for example, Canadian federal law restricts health care and other sensitive data to machines physically located in Canada, while recent news of U.S.-government surveillance programs has prompted increased caution in adopting cloud services, particularly in Europe.

As with most types of trust, trust in a cloud provider by a client is based on history, reputation, the back and forth of an ongoing commercial relationship, and the legal and regulatory setting, as much as it is on technical details. In an effort to entice customers to move to the cloud, providers already provide greater transparency about their operations and proactively attempt to the meet the compliance standards required by the financial and the health-care industries. Given the competing demands of the cloud-infrastructure business, this trend toward transparency is likely to continue. C

References

1. Abadi, M., Budiu, M., Erlingsson, U., and Ligatti, J. Control-flow integrity. In *Proceedings of the 12th ACM Conference on Computers and Communications Security*. ACM Press, New York, 340–353.
2. Amazon. *Summary of the Amazon EC2 and Amazon RDS Service Disruption in the U.S. East Region*; http://aws.amazon.com/message/65648/
3. Amazon. *Summary of the December 24, 2012 Amazon ELB Service Event in the U.S. East Region*; http://aws.amazon.com/message/680587/
4. Amazon. *Summary of the October 22, 2012 AWS Service Event in the U.S. East Region*; https://aws.amazon.com/message/680342/
5. Amazon. *AWS Risk and Compliance*, 2014; http://media.amazonwebservices.com/AWS_Risk_and_Compliance_Whitepaper.pdf
6. Amazon. *Overview of Security Processes*, 2014; http://media.amazonwebservices.com/pdf/AWS_Security_Whitepaper.pdf
7. Bugnion, E., Devine, S., Govil, K., and Rosenblum, M. Disco: Running commodity operating systems on scalable multiprocessors. *ACM Transactions on Computer Systems 15*, 4 (1997), 412–447.
8. Butt, S., Lagar-Cavilla, H.A., Srivastava, A., and Ganapathy, V. Self-service cloud computing. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*. ACM Press, New York, 2012, 253–264.
9. Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I., and Warfield, A. Live migration of virtual machines. In *Proceedings of the Second USENIX Symposium on Networked Systems Design and Implementation*. USENIX Association, Berkeley, CA, 2005, 273–286.
10. Colp, P., Nanavati, M., Zhu, J., Aiello, W., Coker, G., Deegan, T., Loscocco, P., and Warfield, A. Breaking up is hard to do: Security and functionality in a commodity hypervisor. In *Proceedings of the 23rd ACM Symposium on Operating Systems Principles*. ACM Press, New York, 189–202.
11. Dropbox. *Where Does Dropbox Store Everyone's Data?*; https://www.dropbox.com/help/7/en
12. Edbert, J. How Netflix operates clouds for maximum freedom and agility. AWS Re:Invent, 2012; http://www.youtube.com/watch?v=s0rCGFetdtM
13. Erlingsson, U., Abadi, M., Vrable, M., Budiu, M., and Necula, G.C. XFI: Software guards for system address spaces. In *Proceedings of the Seventh Symposium on Operating System Design and Implementation*. USENIX Association, Berkeley, CA, 2006, 75–88.
14. Fraser, K., Hand, S., Neugebauer, R., Pratt, I., Warfield, A., and Williamson, M. Safe hardware access with the Xen virtual machine monitor. In *Proceedings of the First Workshop on Operating System and Architectural Support for the On-Demand IT Infrastructure*, 2004.
15. Ganjali, A. and Lie, D. Auditing cloud management using information flow tracking. In *Proceedings of the Seventh ACM Workshop on Scalable Trusted Computing*. ACM Press, New York, 2012, 79–84.
16. Goel, A., Po, K., Farhadi, K., Li, Z., and De Lara, E. The Taser intrusion recovery system. In *Proceedings of the 20th ACM Symposium on Operating Systems Principles*. ACM Press, New York, 2005, 163–176.
17. Hund, R., Holz, T., and Freiling, F.C. Return-oriented rootkits: Bypassing kernel code integrity protection mechanisms. In *Proceedings of the 18th Conference on USENIX Security*. USENIX Association, Berkeley, CA, 2009, 383–398.
18. Keller, E., Szefer, J., Rexford, J., and Lee, R.B. NoHype: Virtualized cloud infrastructure without the virtualization. In *Proceedings of the 37th Annual International Symposium on Computer Architecture*. ACM Press, New York, 2010, 350–361.
19. Klein, G., Elphinstone, K., Heiser, G., Andronick, J., Cock, D., Derrin, P., Elkaduwe, D., Engelhardt, K., Kolanski, R., Norrish, Sewell, T., Tuch, H., and Winwood, S. sel4: Formal verification of an OS kernel. In *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles*. ACM Press, New York, 2009, 207–220.
20. Kortchinsky, K. Cloudburst: A VMware guest to host escape story. Presented at Black Hat USA 2009; http://www.blackhat.com/presentations/bh-usa-09/KORTCHINSKY/BHUSA09-Kortchinsky-Cloudburst-SLIDES.pdf
21. Kutch, P. *PCI-SIG SR-IOV Primer: An Introduction to SR-IOV Technology.* Application Note 321211-002, Intel Corp., Jan. 2011; http://www.intel.com/content/dam/doc/application-note/pci-sig-sr-iov-primer-sr-iov-technology-paper.pdf
22. Leinenbach, D. and Santen, T. Verifying the Microsoft Hyper-V hypervisor with VCC. In *Proceedings of the Second World Congress on Formal Methods*. Springer-Verlag, Berlin, Heidelberg, 2009, 806–809.
23. Microsoft. *Windows Server 2008 R2 Core: Introducing SCONFIG*; http://blogs.technet.com/b/virtualization/archive/2009/07/07/windows-server-2008-r2-core-introducing-sconfig.aspx
24. Murray, D.G., Milos, G., and Hand, S. Improving Xen security through disaggregation. In *Proceedings of the Fourth ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*. ACM Press, New York, 2008, 151–160.
25. Nguyen, A. Raj, H., Rayanchu, S., Saroiu, S., and Wolman, A. Delusional boot: Securing hypervisors without massive reengineering. In *Proceedings of the Seventh ACM European Conference on Computer Systems*. ACM Press, New York, 2012, 141–154.
26. Ristenpart, T., Tromer, E., Shacham, H., and Savage, S. Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*. ACM Press, New York, 2009, 199–212.
27. Rutkowska, J. and Wojtczuk, R. Preventing and detecting Xen hypervisor subversions. Presented at Black Hat USA 2008; http://www.invisiblethingslab.com/resources/bh08/part2-full.pdf
28. Shacham, H. The geometry of innocent flesh on the bone: Return into libc without function calls (on the x86). In *Proceedings of the 14th ACM Conference on Computer and Communications Security*. ACM Press, New York, 552–561.
29. Steinberg, U. and Kauer, B. NOVA: A microhypervisor-based secure virtualization architecture. In *Proceedings of the Fifth European Conference on Computer Systems*. ACM Press, New York, 2010, 209–222.
30. Thibault, S. and Deegan, T. Improving performance by embedding HPC applications in lightweight Xen domains. In *Proceedings of the Second Workshop on System-Level Virtualization for High-Performance Computing*. ACM Press, New York, 2008, 9–15.
31. University of New South Wales and NICTA. seL4. http://www.ertos.nicta.com.au/research/sel4/
32. VMware. *Benefits of Virtualization with VMware*; http://www.vmware.com/virtualization/virtualization-basics/virtualization-benefits.html
33. VMware. VMware hypervisor: Smaller footprint for better virtualization solutions; http://www.vmware.com/virtualization/advantages/robust/architectures.html
34. Wang, Z. and Jiang, X. Hypersafe: A lightweight approach to provide lifetime hypervisor control-flow integrity. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*. IEEE Computer Society, Washington, D.C., 2010, 380–395.
35. Wang, Z., Wu, C., Grace, M., and Jiang, X. Isolating commodity hosted hypervisors with HyperLock. In *Proceedings of the Seventh ACM European Conference on Computer Systems*. ACM Press, New York, 2012, 127–140.
36. Wilkes, J., Mogul, J., and Suermondt, J. Utilification. In *Proceedings of the 11th ACM SIGOPS European Workshop*. ACM Press, New York, 2004.
37. Wojtczuk, R. A stitch in time saves nine: A case of multiple OS vulnerability. Presented at Black Hat USA 2012; http://media.blackhat.com/bh-us-12/Briefings/Wojtczuk/BH_US_12_Wojtczuk_A_Stitch_In_Time_WP.pdf
38. Wojtczuk, R. and Rutkowska, J. *Following the White Rabbit: Software Attacks against Intel VT-d Technology*, 2011; http://www.invisiblethingslab.com/resources/2011/SoftwareAttacksonIntelVT-d.pdf
39. Zhang, F., Chen, Chen, H., and Zang, B. Cloudvisor: Retrofitting protection of virtual machines in multi-tenant cloud with nested virtualization. In *Proceedings of the 23rd ACM Symposium on Operating Systems Principles*. ACM Press, New York, 2011, 203–216.
40. Zhang, Y., Juels, A., Reiter, M.K., and Ristenpart, T. Cross-VM side channels and their use to extract private keys. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*. ACM Press, New York, 2012, 305–316.

Mihir Nanavati (mihirn@cs.ubc.ca) is a Ph.D. student in the Department of Computer Science at the University of British Columbia, Vancouver.

Patrick Colp (pjcolp@cs.ubc.ca) is a Ph.D. student in the Department of Computer Science at the University of British Columbia, Vancouver.

William Aiello (aiello@cs.ubc.ca) is a professor in the Department of Computer Science at the University of British Columbia, Vancouver.

Andrew Warfield (andy@cs.ubc.ca) is an assistant professor in the Department of Computer Science at the University of British Columbia, Vancouver.