What Does the Future Hold for Hypervisor Security?

Marc Lacoste, Orange Labs

1. INTRODUCTION

Advances in virtualization technologies is one of the key factors fueling growth of cloud computing. The foundational element of any compute cloud is the component performing system virtualization. The *hypervisor* abstracts a host in the form of *Virtual Machines (VMs)*, providing execution environments duplicating the behavior and state of the physical machine. It also achieves VM multiplexing and isolation. Unfortunately, due to numerous reports of bugs or exploits, many concerns have been raised regarding its security. Its almost arbitrary privileges over VMs and its rising complexity make it a juicy target for attackers, a series of novel threats being directed at the hypervisor specifically [Pearce et al. 2013]. Reducing this large attack surface is a real security challenge.

Architecture has a decisive influence on hypervisor security, to make it "secure by design". Hypervisor architecture has been changing for two reasons: new architectures have been proposed to mitigate the previous threats; virtualization has also been expanding outside the data center, leading to new forms of hypervisors. Evolutions are substantial and happening in very short timeframes. To protect efficiently emerging virtualized systems, it is critical to have an overall view of such evolutions: are counter-measures proposed really addressing upcoming security threats? Despite the growing number of papers already published, such a bird's eye view seems to be missing.

This paper aims to take a step back and sketch a big picture by abstracting hypervisor evolution, present and future, into a consistent roadmap. We find three major disruptions that may strongly influence hypervisor security. Along evolution paths driven by each disruption, we identify some main steps corresponding to different types of hypervisor architectures. We discuss key challenges, benefits, and limitations of each design approach. We also attempt to give an idea of what hypervisor security could look like in the future.

2. A BIG PICTURE

Many technological trends induce strong evolutions in hypervisor architecture. Among those trends are availability of ever more small-scale devices with compute, storage, and network capabilities; rising software complexity and commoditization of hardware for dedicated processing; and fall of frontiers between virtualized systems that tend to be increasingly distributed and seamlessly interconnected.

Changes in the hypervisor landscape may be analyzed along two dimensions (see Figure 1): (1) the *scale* of the virtualized system, ranging from small (e.g., chip, embedded systems), medium (e.g., data center), to very large (e.g., federated clouds); and (2) the *abstraction level* of security mechanisms in the virtualization stack, i.e., closer to VMs, or to the hardware.

Three main trends might shape the future of hypervisor security: (1) a virtualization extension to *the embedded space*, for increasingly resource-constrained devices; (2) a "vertical" *migration of security* mechanisms *towards the hardware*; and (3) a "horizontal" extension to *multi-cloud infrastructures* with distributed, networked hypervisors.

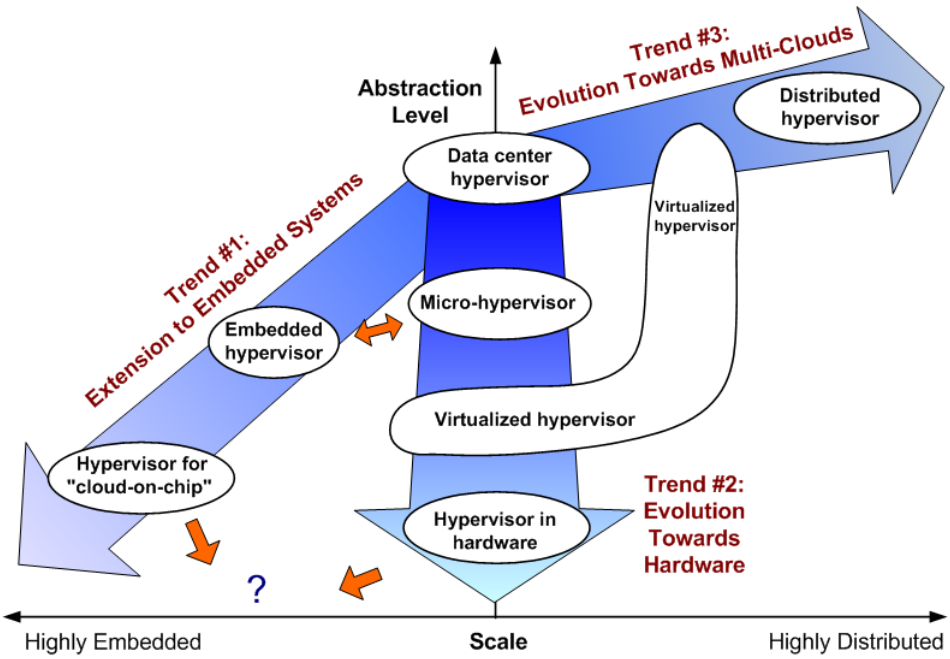

Fig. 1. Evolution of the hypervisor landscape.

3. EXTENDING VIRTUALIZATION TO THE EMBEDDED SPACE

3.1. Embedded Hypervisors

From its early days in the data center setting, virtualization has made a lot of headway, with increasingly wider inroads in the embedded space. Lightweight hypervisors have started to appear on mobile phones, sensors, automotive systems, or in avionics. Such systems are feature-rich platforms built from very heterogeneous sub-sytems, with rising complexity and code size, strong hardware diversity, many security issues, and increasingly open architectures.

The key properties to guarantee are: (1) *resource abstraction* to overcome hardware heterogeneity; (2) *isolation* to contain interferences between sub-systems, e.g., cascasding failures or vulnerabilities; (3) *performance* with strong expectations on communication latency between sub-systems; (4) *TCB (Trusted Computing Base) minimization* to reduce the attack surface and raise the assurance level; (5) *real-time (RT) behavior* enforced at the scheduler level; (6) *architectural flexibility* to enable code reuse through increased modularity; and (7) *fine-grained control* over resources.

Embedded virtualization brings benefits to achieve several such properties, for instance abstraction and isolation by multiplexing and isolating VMs. However, current hypervisor architectures show strong limitations for many others. A main flaw is the huge TCB size, leading to reliability, security, and administration issues. Flexibility is rather poor, as for device sharing. RT guarantees are also difficult due to two-level system scheduling, i.e., threads at VM-level, and VMs at hypervisor-level. Finally, resource control remains coarse-grained through heavyweight VMs.

Micro-kernel design has thus been increasingly popular to meet such challenges. Reducing the kernel to its fundamental elements, and making the rest of the OS deprivileged outside results in an extremely minimal TCB that may be formally verified. Strong isolation and high-perfromance communication between address spaces is also achieved through very efficient IPCs. Micro-kernels increasingly support virtualization: privileged instructions are intercepted and emulated in a dedicated system service. Control is finer than in the hypervisor, both by the granularity of isolation (lightweight threads), and by a unique control point of RT behavior in the scheduler. Device drivers are run in user space above the micro-kernel, allowing great flexibility for device sharing patterns.

Such architectures are very similar, although the hypervisor is driven by abstraction, to match closely virtual to real resources, while micro-kernels are guided by TCB minimization. They tend to converge: the *micro-visor* allows to run both VMs and threads over a lightweight execution environment [Heiser and Leslie 2010]. Like micro-kernels, it has a minimal TCB, and high-performance asynchronous IPCs between VMs, applications, or user-mode drivers. Like hypervisors, it allows virtualization of CPU, memory, or I/O through a hardware abstraction layer (HAL). Arbitrary control over resource partitioning in security domains or RT scheduling of threads or VMs is also possible. Finally, the micro-visor offers great architectural flexibility: drivers may be virtualized in a VM, or executed as system services in their own address space.

3.2. Towards the "Cloud-on-Chip"

With continuous progress of on-chip integration, virtualization may be expected to reach deeper in the embedded device, extending to massively multi-core hardware architectures. In this area referred to as "*cloud-on-chip*" by Intel, key concerns are to strictly limit resource sharing (e.g., CPU state, memory) between multiple cores, and to handle scalability (e.g., synchronization, fair resource allocation). Single hypervisor designs fail to achieve such goals.

Architectures supporting *multiple hypervisors on the same chip* might thus be the next step [Shi et al. 2012]. Multiple independent *security realms* are introduced. Each realm is controlled by a single hypervisor running its own VMs with dedicated cores and memory, without inter-realm sharing. Realms realize a distributed system enforcing a two-level resource management policy: hypervisors control their VMs, while a HAL performs hypervisor multiplexing and strong physical isolation. Expected benefits are: increased resilience, avoiding bugs or attack propagation from a hypervisor (or one of its VMs) to other realms; and better scalability, fully taking advantage of hardware capabilities at a finer grained level.

4. MIGRATING IAAS SECURITY TOWARDS THE HARDWARE

In parallel, another strong trend is to move IaaS security mechanisms increasingly closer to the hardware, as shown in Figure 2. This evolution in proposed security solutions corresponds to ever more precise answers to the key question: should the hypervisor be trusted or not?

4.1. Trusted Hypervisors

In-VM security. The hypervisor was first assumed to be part of the TCB. VM security was then guaranteed using mechanisms embedded in the VM, by analogy with a non-virtualized system. Placing security mechanisms within the VM benefits from proximity with the resource to protect, to increase accuracy of detection of unauthorized usage, and relevance of subsequent reactions. However, it may compromise stealth and transparency: in-VM programs may detect or alter the security component.

Hypervisor-based VM security. Going below, *VM introspection (VMI)* [Garfinkel and Rosenblum 2003] showed how to leverage the capabilities of the hypervisor to analyze or modify VM behavior, triggering a very active stream of research. Comparing monitoring information gathered from VM and hypervisor layers enables to enforce integrity policies and detect violations. Monitoring can also be performed in an *"out-of-VM" appliance*, or dedicated security VM. VMI greatly improves security by clear separation of protection mechanism from protected resource. However, this security is based on the hypervisor running at a higher level of privilege, and being trusted.

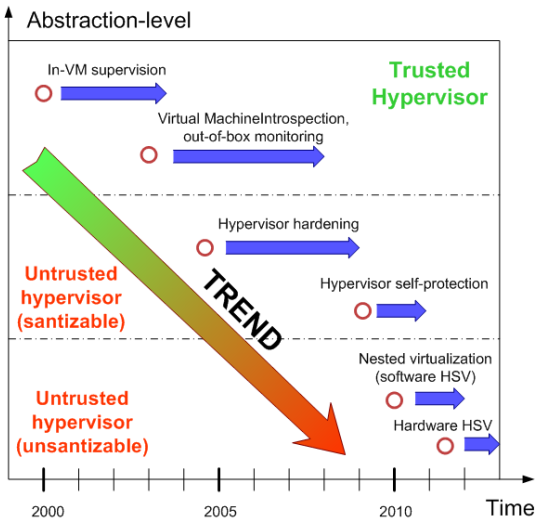

Fig. 2. Placement of IaaS security mechanisms.

4.2. Micro-Hypervisors

The extended privileges of the hypervisor over VMs pose severe risks for VM confidentiality, integrity, or availability in case of compromise. Recently published attacks targeting directly the hypervisor radically changed the way of thinking about the hypervisor as fully trusted. Attack vectors are buggy or vulnerable *device drivers* which enable kernel exploitation due to poor confinement. In such *VM escape* attacks, the driver is used as a stepping stone to inject malicious code, breach VM isolation, subvert the hypervisor, and corrupt other parts of the virtualized system. Improving hypervisor security is particularly difficult due to the size and complexity of the TCB that is too large to be free from bugs or vulnerabilities. Two types of approches have thus been explored: (1) *TCB hardening*; and (2) *TCB minimization*.

TCB hardening. A large variety of techniques were proposed to design mechanisms to protect "by hand" the hypervisor from subversion. One can for instance mention: trusted computing architectures [Azab et al. 2010] and language techniques [Wang and Jiang 2010] that provide strong code integrity and authenticity guarantees; or sandboxing [Coker 2007] to confine malicous driver code by strict control of communications between driver and device, kernel, or user space. However, such techniques remain limited to pure integrity detection or simple containment, with few solutions for sanitizing a corrupt part of the hypervisor.

TCB reduction. Architectural solutions were also created to make the hypervisor more minimal and flexible, targetting its two most vulnerable elements: (1) the core hypervisor itself; and (2) the management VM (e.g., Xen Dom0).

A first class of solutions aims *to reduce the hypervisor size*. The new hypervisor architecture should be more minimal, drawing inspiration from evolution in OS architecture (e.g., extensible, micro-, exo-, component-based kernels). Within the hypervisor are distinghuished security-critical modules from non-sensitive code. The aim is to expel as much code as possible from the TCB and target ultra-thin hypervisors, also known as *micro-hypervisors*. For instance, some components of the hypervisor (e.g., device drivers, VM address space management) may be virtualized outside the micro-hypervisor, isolated, and restarted in case of compromise [Steinberg and Kauer 2010]. The extreme case is to remove the virtualization layer altogether [Szefer et al. 2011]. Unfortunately, although promising, such architectures usually require extensive code rewriting. They also provide a very limited number of resource management services unlike an operational IaaS infrastructure. They are thus hard to apply to mainstream hypervisors today.

A second class of solutions aims *to improve management VM modularity*. Such solutions are meant to be applied to legacy hypervisors [Colp et al. 2011]. The approach is: (1) to componentize the administration VM into a number of *service VMs*; (2) to separate their privileges, e.g., only the most critical service VMs have full Dom0 administration privileges; and (3) to limit resource sharing between service VMs. This approach does not limit the number of services provided by the IaaS infrastructure. It also introduces more extensibility in a very monolithic architecture. It finally provides fine-grained control over infrastructure system services (e.g., live migration, advanced resource management) provided to cloud customers.

Automated hardening. In previous solutions, administration of security remains complex: protection policies are often hardcoded, and must be configured manually. Dynamic, reactive protection strategies are thus difficult to set up. Automating hypervisor security hardening was thus also explored, e.g., by applying autonomic computing concepts to build *self-protecting hypervisors* [Wailly et al. 2012]. Protection is regulated through several feedback loops that supervise different parts of the hypervisor through well-defined hooks. Interactions such as memory accesses between a driver and its hypervisor environment may thus strictly be monitored and controlled automatically. However, coming up with an efficient implementation turned out to be quite hard in practice.

4.3. Virtualized Hypervisors

Automated hypervisor sanitization may fail, not only due to overly high system complexity, but also because infection might be too advanced for recovery. *Hypervisor-secure virtualization (HSV)* architectures have therefore been explored to protect VMs even in case of complete hypervisor compromise [Jin et al. 2011; Szefer and Lee 2012]. The hypervisor is no longer part of the TCB. It is protected by a trusted security layer underneath, realized in hardware or software. In the hypervisor, resource management is decoupled from security: the hypervisor can still allocate memory pages, or launch, schedule, or kill VMs; but it no longer handles VM isolation through direct updates of VM nested page tables. Such resource protection is delegated to the security layer that keeps track of memory page ownership by VMs or the hypervisor.

The software approach for HSV is based on *nested virtualization* [Ben-Yehuda et al. 2010]. An extra hypervisor (L_0) implements the security layer below the hypervisor to protect (L_1), with applications such as rootkit detection at hypervisor or VM levels. L_0 security is hardware enforced. Several virtualization levels may be multiplexed on a single architectural support: for the CPU, a nested trap-and-emulate mechanism and hardware virtualization extensions (e.g., Intel VT-x) enable to run L_1 just like a regular VM; for memory, different alternatives are available for multi-level page table collapsing depending on hardware capabilities. For instance, VM security may be achieved in spite of an untrusted L_1 by delegating protection to L_0 to perform interposition between the VMs and L_1 [Zhang et al. 2011]: a `VMExit` (resp. `VMEntry`) event will be trapped and processed before transferring back control to L_1 (resp. to the VM). Such processing includes safe register context switching, memory isolation enforcement, and memory page encryption/decryption to prevent VM data from being read by untrusted L_1.

4.4. The Hypervisor in Hardware

In the future, IaaS security will be about protecting VMs even if several layers below are fully compromised. Hardware mechanisms might be in the best position to provide such guarantees as last line of defence. Hardware HSV architectures should therefore gain momentum. A small hardware controller (HC) is introduced as unique security manager, e.g., for memory virtualization. Only the HC is authorized to perform updates to nested page tables, checking memory mapping permissions using hardware-defined *Page Ownership Tables (POT)*. POTs are separated from page tables so that the hypervisor may still perform memory allocation and VM scheduling transparently. Hardware and hypervisor modifications remain minimal. Other alternatives based on a secure execution mode (e.g., ARM TrustZone), or on secure processor architectures (e.g., cloud-on-chip) may also be considered.

Besides stronger security, such hardware designs should be expected to offer better performance than pure software solutions. However, their cost, traditionally very high, might no longer be a barrier in the future: resulting changes in micro-architecture remain generally small, and providers may be willing to pay for the extra silicon to get an enhanced assurance level. Current developments of virtualization over multi-core chips should confirm whether this is a durable trend.

5. DISTRIBUTING VIRTUALIZATION FOR MULTI-CLOUDS

The last major disruption in hypervisor architecture is the hypervisor going distributed. This evolution is driven by a change of vision in cloud resource delivery: the cloud moves from being *provider-centric* to *user-centric*. Today, the cloud landscape is centered around *Cloud Service Providers (CSP)* operating multiple heterogeneous clouds. Each CSP provides its own feature-rich offerings for customer VMs, such as a great diversity of hypervisor-based infrastructure services. Unfortunately, deficiencies in lack of control over the infrastructure and in interoperability result in many such services being not deployed at all. For instance, non-flexible hypervisors prevent fine-grained cloud customization. Similarly, services are hypervisor-specific, and not compatible across CSPs.

Instead, there is growing interest for a more homogeneous vision of the cloud, *closer to user requirements*: a cloud resource distribution plane, the *super-cloud (SC)* layer, is introduced to decouple production (by CSPs) from consumption (by users) of cloud resources [Williams et al. 2013]. The super-cloud spans CSPs to achieve independence from providers and from their hypervisor services. It is also an extensibility layer: new players (SC providers) operating the distribution layer may provide their own services enabling the customer to deploy self-service, fully customizable clouds ranging from SaaS to the full IaaS.

Realizing this vision should make hypervisor architecture both more distributed and flexible. Distribution starts with multi-IaaS platforms, operating more at the VM level above provider hypervisors, and extending towards fully-networked hypervisors. Nested virtualization looks also highly promising in that respect: it enables to define at system-level a multi-cloud abstraction layer for hypervisor interoperability [Williams et al. 2012]. The next step is to leverage this layer to deploy flexibly SC services, such as security supervision. The CSP (resp. SC provider) then controls the lower (resp. upper) hypervisor. Other design alternatives are also available for flexibility such as extensible hypervisors or modular management VMs, separating the distributed IaaS into user- and provider-controlled modules or domains, isolated by a privilege separation approach [Butt et al. 2012]. The research field is only to starting to expand in that direction [Liu and Mao 2013].

6. CONCLUSION

Damage caused by exploitation of a virtualization vulnerability may be as serious as impact of other cloud security threats, more widespread and less advanced. The hypervisor is thus more than ever a keystone component of virtualized infrastructure security. The two main hypervisor protection challenges remain vulnerabilities induced by its rising complexity and the rapid evolution of threats. New hypervisor architectures look promising to meet such challenges but are far from mature. Architectures may be seen as moving on a 360° radar, vertically striving for cross-layer protection, and horizontally to guarantee end-to-end security, either in the small in very limited devices, or in the large across federated clouds. Deep transformations may be expected as virtualization becomes pervasive, moving towards the embedded setting, changing towards hardware-enforced security, or going large-scale.

Looking back at Figure 1, one sees that architectural trends are clearly diverging. This may reflect the fact that in the future, there may not be one, but multiple "good" architectures for a secure hypervisor. Hypervisor design is still a very young research domain, as for operating systems architecture quite a few decades ago. A few years from now, one may thus expect a similar wealth in the hypervisor design spectrum to find the right architectural trade-offs between TCB minimality, interoperability, flexibility, abstraction, performance, and other requirements. Hypervisor security is also a very fast moving research domain that should gain prominence as the cloud expands. It remains thus critical to monitor closely such evolutions to protect effectively virtualized systems of the future.

REFERENCES

Ahmed M. Azab, Peng Ning, Zhi Wang, Xuxian Jiang, Xiaolan Zhang, and Nathan C. Skalsky. 2010. HyperSentry: Enabling Stealthy In-Context Measurement of Hypervisor Integrity. In *ACM Conference on Computer and Communications Security (CCS)*.

Muli Ben-Yehuda, Michael D. Day, Zvi Dubitzky, Michael Factor, Nadav Har'El, Abel Gordon, Anthony Liguori, Orit Wasserman, and Ben-Ami Yassour. 2010. The Turtles Project: Design and Implementation of Nested Virtualization. In *9th USENIX Conference on Operating Systems Design and Implementation (OSDI)*.

Shakeel Butt, H. Andrés Lagar-Cavilla, Abhinav Srivastava, and Vinod Ganapathy. 2012. Self-Service Cloud Computing. In *ACM Conference on Computer and Communications Security (CCS)*.

George Coker. 2007. Xen Security Modules (XSM). In *Xen Summit*.

Patrick Colp, Mihir Nanavati, Jun Zhu, William Aiello, George Coker, Tim Deegan, Peter Loscocco, and Andrew Warfield. 2011. Breaking Up is Hard to Do: Security and Functionality in a Commodity Hypervisor. In *ACM Symposium on Operating Systems Principles (SOSP)*.

Tal Garfinkel and Mendel Rosenblum. 2003. A Virtual Machine Introspection-Based Architecture for Intrusion Detection. In *Network and Distributed Systems Security Symposium (NDSS)*.

Gernot Heiser and Ben Leslie. 2010. The OKL4 Microvisor: Convergence Point of Microkernels and Hypervisors. In *ACM SIGCOMM Asia-Pacific Workshop on Systems (APSys)*.

Seongwook Jin, Jeongseob Ahn, Sanghoon Cha, and Jaehyuk Huh. 2011. Architectural Support for Secure Virtualization under a Vulnerable Hypervisor. In *Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*.

Changbin Liu and Yun Mao. 2013. Inception: Towards a Nested Cloud Architecturee. In *5th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*.

Michael Pearce, Sherali Zeadally, and Ray Hunt. 2013. Virtualization: Issues, Security Threats, and Solutions. *ACM Computing Surveys* 45, 2 (2013).

Weidong Shi, Jong Hyuk Lee, Taeweon Suh, Dong Hyuk Woo, and Xinwen Zhang. 2012. Architectural Support of Multiple Hypervisors over Single Platform for Enhancing Cloud Computing Security. In *ACM International Conference on Computing Frontiers (CF)*.

Udo Steinberg and Bernhard Kauer. 2010. NOVA: A Microhypervisor Based Secure Virtualization Architecture. In *ACM European Conference on Computer Systems (EUROSYS)*.

Jakub Szefer, Eric Keller, Ruby B. Lee, and Jennifer Rexford. 2011. Eliminating the Hypervisor Attack Surface for a More Secure Cloud. In *ACM Conference on Computer and Communications Security (CCS)*.

Jakub Szefer and Ruby Lee. 2012. Architectural Support for Hypervisor-Secure Virtualization. In *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*.

Aurélien Wailly, Marc Lacoste, and Hervé Debar. 2012. KungFuVisor: Enabling Hypervisor Self-Defense. In *EUROSYS Doctoral Workshop (EURODW)*.

Zhi Wang and Xuxian Jiang. 2010. HyperSafe: A Lightweight Approach to Provide Lifetime Hypervisor Control-Flow Integrity. In *IEEE Symposium on Security and Privacy*.

Dan Williams, Hani Jamjoom, and Hakim Weatherspoon. 2012. The Xen-Blanket: Virtualize Once, Run Everywhere. In *ACM European Conference on Computer Systems (EUROSYS)*.

Dan Williams, Hani Jamjoom, and Hakim Weatherspoon. 2013. Plug into the Supercloud. *IEEE Internet Computing, Special Issue on Virtualization* 17, 2 (2013).

Fengzhe Zhang, Jin Chen, Haibo Chen, and Binyu Zang. 2011. CloudVisor: Retrofitting Protection of Virtual Machines in Multi-Tenant Cloud with Nested Virtualization. In *ACM Symposium on Operating Systems Principles (SOSP)*.