Self-Defending Clouds: Myth and Realities*

Marc Lacoste[1], Aurélien Wailly[1], and Hervé Debar[2]

[1] Orange Labs, Security and Trusted Transactions Dept.
38-40, rue du Général Leclerc, 92794 Issy-Les-Moulineaux, France
{marc.lacoste,aurelien.wailly}@orange.com
[2] Télécom SudParis, CNRS Samovar UMR 5157
9, rue Charles Fourier, 91011 Evry, France
herve.debar@telecom-sudparis.eu

Abstract. Security is a growing concern as it remains the last barrier to widespread adoption of cloud environments. However, is today's cloud security Lucy in the Sky with Diamonds? Expected to be strong, flexible, efficient, and simple? But surprisingly, being neither? A new approach, making clouds *self-defending*, has been heralded as a possible element of answer to the cloud protection challenge. This paper presents an overview of today's state and advances in the field of cloud infrastructure self-defense. Four key self-protection principles are identified for IaaS self-protection to be effective. For each layer, mechanisms actually deployed to deliver security are analyzed to see how well they fulfill those principles. The main remaining research challenges are also discussed to yield truly mature self-defending clouds.

Keywords: Cloud Security Supervision; Cloud Security Management; Self-Defending Clouds; Cloud Threats; Autonomic Security; IaaS Infrastructures.

1 What's Wrong with Today's Cloud Security?

Security is undoubtedly the # 1 barrier for cloud technology adoption, as its very nature raises multiple concerns regarding protection of computing, networking, and storage resources. What about it?

Ideally, cloud security should be *strong*, *flexible*, *efficient*, and *simple* (Figure 1):

- *Strong security* is required to face new threats introduced by virtualization, or by resource sharing, as clouds are by essence multi-tenant environments. The crux is to achieve strict isolation between Virtual Machines (VMs), which may fail if the virtualization layer is compromised. To guarantee perimetric security, dissolved organization boundaries also make it more difficult to define the "inside" and the "outside" of an infrastructure.
- *Flexible security* is needed to respond to the multiplicity of threats and to their evolution. This means security resource provisioning should adapt to changes in the cloud environment to provide an optimal level of protection. Flexibility is also a must considering the diversity of stakeholders using cloud services, each with their security objectives, processes, and technical components.

* This work was supported by the OpenCloudWare project, funded by the French Fonds national pour la Société Numérique (FSN).

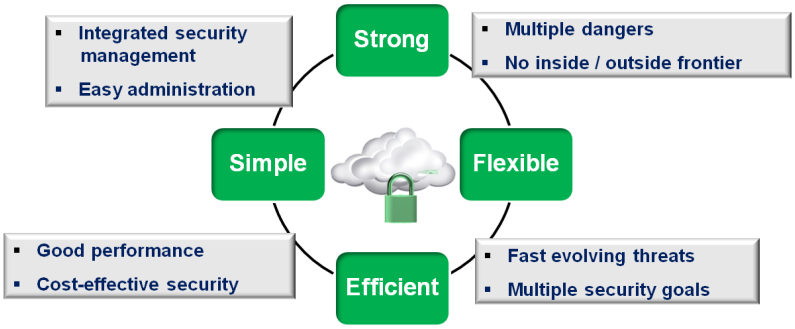

Fig. 1. The Myth – Cloud Security Expectations.

- *Efficient security* is desirable to guarantee security SLAs, e.g., to keep end-to-end incident response times short. Managing vulnerabilities, detecting intrusions, and activating defenses should be performed smoothly, transparently, and in a way which is both highly scalable and non-intrusive regarding performance.
- Above all, *simple security* is essential to keep infrastructure administration within manageable bounds. This requires to abstract away heterogeneity of security building blocks protecting systems, networks, and data to provide an integrated view of cloud defense.

But unfortunately, the reality of today's cloud security is a bit different. Protection appears neither as *so strong*, *so flexible*, *so efficient*, nor *so simple* (Figure 2):

- Cloud security is not really as *strong* as expected: vulnerabilities are hard to detect, and counter-measures hard to place. If in theory, mainstream hypervisors have a low surface of attack, in practice, new classes of threats such as malicious device drivers or rootkits in the virtualization layer call for higher levels of assurance.
- Similarly, security is neither very *flexible* nor *efficient*: many security configurations are still static, policies being often hardcoded within the protection mechanisms. Dynamic, reactive protection strategies are thus hard to set up, as policies must be configured and updated manually – a costly and error-prone process. A generic security supervision architecture is also lacking for the cloud.
- Finally, cloud security is definitely *not simple*: available security components are highly fragmented. How to orchestrate them to guarantee end-to-end security is still undefined, with as main barriers heterogeneity, scalability and interoperability. The problem also turns into a maintenance nightmare for a nebula of different administrators who are not even aware of the existence of one another. Traditional approaches to protection are thus clearly not enough!

Self-protection has recently raised growing interest as possible element of answer to the cloud computing infrastructure protection challenge. Research initiated by IBM on *autonomic security architectures* [7] has the ambition to build infrastructures where security is self-managed: the idea is that security parameters are autonomously negotiated with the environment to match the ambient estimated risks to provide an optimal level of protection. For cloud computing, the result is a *self-defending cloud* infrastructure.

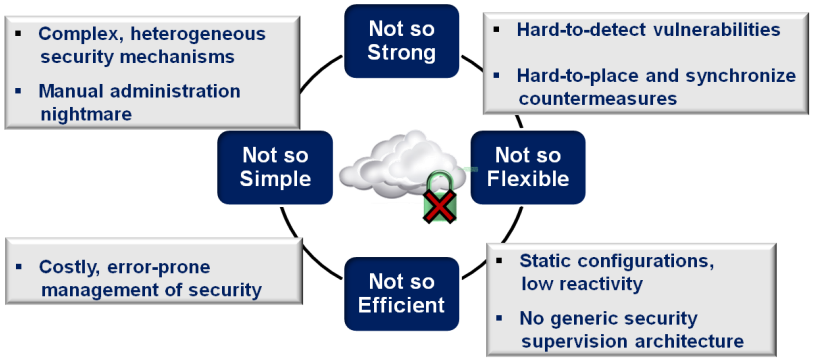

Fig. 2. Reality – Current Cloud Security.

Faced with multiple threats and heterogeneous defense mechanisms, the autonomic approach appears tempting by proposing simpler, stronger, and more efficient cloud security management. Quite a few projects [1, 9, 19, 23] have already investigated the design, implementation, and deployment of self-protection architectures and mechanisms regarding different aspects of infrastructure security. Prospects look rather good, as fully automated security supervision of cloud and inter-cloud infrastructures could be at hand! At the same time, policy-driven security automation still remains at a very early stage for the cloud. First attempts seem to fall at the last hurdle as they overlook several key cloud features.

The objective of this paper is to clarify where we currently stand regarding self-defense of cloud infrastructures. After providing some background on self-protection and its benefits, we identify four *fundamental principles* that a IaaS infrastructure should satisfy for self-protection to be effective: (1) flexible security policies; (2) cross-layered defense; (3) multiple loops; and (4) open security architecture.

We analyze how well current cloud security mechanisms fulfill those principles for each infrastructure layer. We also identify the main *remaining challenges* to solve to yield truly mature self-defending cloud infrastructures.

The rest of this paper is organized as follows. After a survey of threats to sensitive assets in a IaaS infrastructure (Section 2), we recall the approach of cloud self-defense and its benefits (Section 3). We then present the self-protection principles and their coverage by existing cloud security mechanisms (Section 4). Finally, we discuss remaining research challenges and perspectives (Section 5).

2 Threats in a IaaS Infrastructure

A typical IaaS infrastructure is generally organized in three main layers (Figure 3):

– The *physical layer* consists of computing (CPU, memory), networking, and storage resources spread over the cloud infrastructure. Those hardware resources are virtualized and shared among virtual machines.

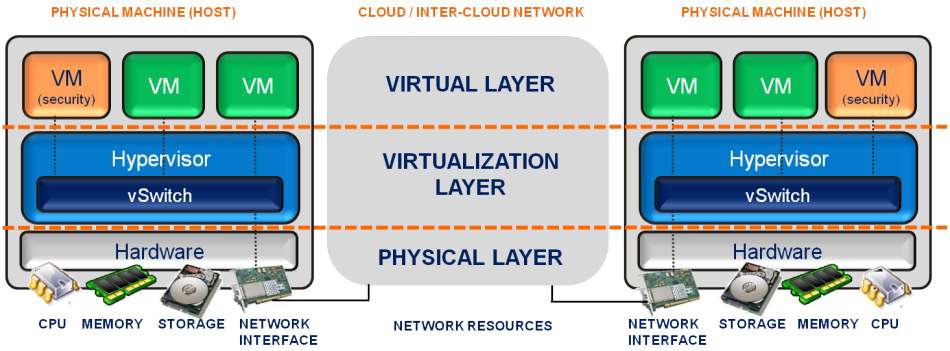

Fig. 3. Elements of a IaaS Infrastructure.

- The *virtualization layer* enables to deploy and run concurrently multiple instances of a software stack (applications, middleware, and OS) on a same physical host in the form of VMs. This task is performed by a specific component, the *hypervisor*, which manages the allocation of physical machine resources among VM instances. It notably guarantees strict isolation between VMs. The hypervisor also enables communications between VMs, either locally on the same host, or through the cloud network through a low-level software bus, often called *virtual switch*.
- The *virtual layer* is composed of the VMs running over the cloud infrastructure. From a security viewpoint, two broad types of VMs may be distinguished: (1) *user VMs* are the customer VMs to be protected. They typically contain customer code and data which may be security-sensitive; (2) *security VMs* are security services (firewalls, intrusion detection systems, anti-malware tools...) running as virtual machines in order to protect the user VMs. This class of VMs is often referred to as *virtual security appliances*.

Some of the main threats against a IaaS infrastructure are described next.

VM-to-VM Threats. A malicious VM may attempt to fool the IaaS VM placement strategy to run on the same physical machine as the attack target (another VM). It may then take advantage of a flaw in hypervisor isolation to launch a side-channel attack to steal or corrupt information from the target VM.

Hypervisor Subversion. More potent attacks attempt to take control of the hypervisor itself from a malicious VM (*hyperjacking*). Such attacks undermine the commonly established idea that commodity hypervisors have a relatively low surface of attack. A VM is able to "escape" from hypervisor isolation enforcement to take full control of the virtualization layer. Possible attack vectors include misconfigurations, or malicious or poorly confined device drivers in the hypervisor. Possible subsequent steps include compromising hypervisor integrity, installing rootkits, or launching an attack against another VM, resulting in breaches in confidentiality, integrity, or availability (e.g., Denial of Service). In the past few years, such isolation breakout attacks have been published for nearly all main hypervisors.

Network Threats. Traditional network security threats such as *traffic snooping* (i.e., intercepting network traffic), *address spoofing* (i.e., forging VM MAC or IP addresses), or *VLAN hopping* (i.e., breaking out of traffic segregation) are also possible, either between physical hosts, or between VMs on the same host.

Availability Threats. This is also a major issue due to resource sharing. Faulty or malicious VM behaviour may lead to resource starvation and interruption of service. For instance, the cloud platform may be brought to a halt, hosts running out of memory due to greedy VM behaviors. Such events may greatly alter the cloud provider image. Crimeware-as-a-Service scenarios may be even worse: the large amount of cloud resources are then deliberately used by hackers to launch massive attacks against juicy targets, for instance using botnets or other malwares.

Information Security Threats. This class of threat such as violations of confidentiality or integrity should also be taken into account for stored data. Few techniques are available to address those threats specifically, apart from traditional cryptographic counter-measures (encryption, signature, authentication).

3 Self-Defending Clouds

3.1 Approach: Automation of Security Management

To address most of the previous threats, it is necessary to detect and prevent intrusions in a way which is as automated as possible. Indeed, cloud infrastructures provide dynamic and flexible services by federating highly heterogeneous resources, hiding underlying complexity. By nature, such infrastructures involve a significant number of physical resources, further growth requiring new management capacities. Unfortunately, current administration solutions, involving manual intervention, do not scale to such complexity. Moreover, the absence of central knowledge may result in unintended conflicts, wasting administrator time. *Automated management of cloud security* is thus decisive for infrastructure stability, protection, as well as for cost-effectiveness.

This is precisely the aim of the approach put forward by IBM regarding *autonomic management of security*, where a system becomes self-protected to automate response to incidents, and react rapidly to detected attacks [7]. Autonomic security applies the idea of flexibility to the security space itself. It goes a step further than simple adaptation by automating the entire process of reconfiguration, thus making the security mechanisms self-responsive, almost running without any user intervention[3]. From a design perspective, this introduces a control structure which oversees the different aspects of the reconfiguration activity – sense, analyze and respond. A feedback loop is set up to select the adequate security policies matching the ambient estimated risk, and achieve an optimal level of protection of system resources.

[3] The system is running "almost" without user intervention, as the user is still part of the loop: users notably specify the security adaptation strategy defining the response to changes in the security context. The user can also be notified, or prompted to take a decision, in case unexpected events occur that the system cannot manage autonomously. Thus, security management is viewed as a closed loop, but in which the user has still an active role to play.

This approach is related to *Intrusion Detection and Prevention Systems (IDPS)*, whose goal is also to monitor a system, analyzing its behavior to detect attacks, and possibly react to them [10]. Self-protection has been viewed as a promising next step of such well-established techniques to fight intrusions [11]. In addition to yielding stronger security management of infrastructures, automated capabilities of detection of and reaction to attacks also bring clear benefits such as lighter administration, lower incident response times, or reduced error-rates.

3.2 The Autonomic Security Loop

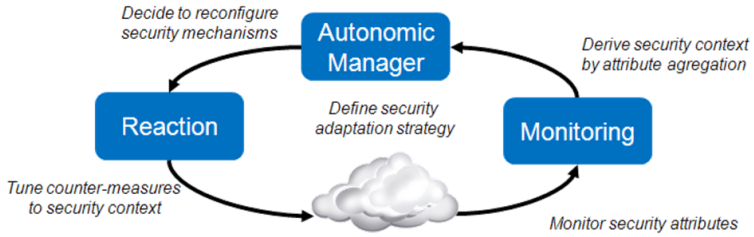

Fig. 4. A Typical Self-Protection Loop.

Operation of a self-protecting system includes the following phases (Figure 4):

- *Detection*: information regarding monitored system activity is collected through sensors, and correlated to derive an overall picture of the risk level. Alerts are raised whenever abnormal activity or known attack patterns are detected.
- *Decision*: alerts are analyzed and checked against the system security policy. In case of policy violation, launching of counter-measures in the infrastructure may be triggered during this phase.
- *Reaction*: counter-measures selected during the decision phase are activated to mitigate the detected attack.

More precisely, an autonomic security loop can be instantiated using a small number of components:

- *Security Context Provider*: this component provides a high-level generic description of the current context. Low-level input data are gathered from different sources (e.g., system/network monitoring components) and aggregated into a higher-level generic representation of the current context – e.g., through a context management infrastructure. The Security Context Provider also provides a description of the ambient security context. A set of security-related attributes (e.g., security levels) are extracted from the generic information provided by the context management infrastructure – e.g., through an inference engine.
- *Decision-Making Component*: based on the security context, this component decides whether or not to reconfigure the security infrastructure, for instance to relax the strength of authentication, or change cryptographic key lengths.

– *Adaptable Security Mechanisms*: The decision is then transmitted to the security mechanism which should be adapted. Reconfiguration is performed by changing the needed component. Security mechanisms should be flexible enough to be reconfigured, by tuning security configuration parameters, or by replacing them with other components offering similar security services. Specific support mechanisms are needed to guarantee that the reconfiguration process is safe and secure.

The autonomic loop is then set up, with monitoring, decision, and action steps. At this point, the system is able to negotiate the security parameters autonomously with its environment, fulfilling the vision of a *self-protecting system*.

3.3 Benefits

The autonomic paradigm applied to manage the security of IaaS infrastructures yields *self-defending clouds*. This approach enables to lighten administrator workload, and offers increased reactivity in an environment where threats are constantly evolving. This results in stronger, more flexible, more efficient, and simpler cloud security.

With such an approach, a cloud infrastructure provider can leverage a self-defense framework to secure virtual machines simply, quickly, dynamically, and automatically while benefiting from increased cost effectiveness, thus making it a key element of a differentiation strategy. Key benefits are lighter administration of the infrastructure, increased reactivity and agility, and lower operating costs. The cloud provider also benefits from an automated solution to set up a "graduated response" against cloud threats while ensuring effective isolation of virtualized resources. This solution should therefore be directly relevant to achieve strong and flexible security in the corresponding products, offering a clear competitive advantage. Finally, an autonomic security framework could be an enabler for providing integrated security supervision of cloud and multi-cloud infrastructures.

Risk-aware, dynamic VM quarantine is a typical application of the self-defense approach. It may be realized as follows. In each monitored VM, a lightweight agent (similar to an anti-virus) gathers and analyzes all files loaded into memory before they are accessed. A VM in charge of system supervision aggregates and orchestrates all decisions concerning the security of the architecture. Decisions are transmitted to different security elements of the infrastructure to confine infected VMs. Once the threat has been eradicated by the agent, the quarantine may be released. The user recovers his VMs free of viruses immediately, and without loss of data.

4 Reality: Mechanisms

4.1 Design Principles

How far are we today from building and operating such systems? Cloud environments come with a number of distinguishing features that should be considered from a security architecture standpoint. The three main dimensions to keep in mind are:

- *Multi-layering*: a cloud infrastructure is built from many independent software layers with their specific security mechanisms, while attacks may target several layers.
- *Multi-laterality*: a cloud involves multiple organizations with their own security objectives, calling for flexible policies and monitoring granularities for security management.
- *Openness*: clouds are increasingly evolving towards interoperability with other clouds or third-party IT systems, making a closed-world vision of security not adequate.

Those dimensions in turn allow deriving a small set of principles to guide the design of new, self-defending cloud architectures. In any case, it seems reasonable to assume that those design principles may be satified by any such existing systems. Those principles are stated and discussed below.

Principle 1 (Policy-Based Self-Protection) *The architecture should be a refinement of a well-defined security adaptation model based on policies.*

This approach has well-known benefits to increase self-management adaptability and extensibility [29]. Therefore, a richer choice of security strategies can be supported in each phase of the control loop [18]. For instance, a wider range of detection and reaction policies can be defined and enforced.

Principle 2 (Cross-Layer Defense) *Detection and reaction should not be performed within a single software layer (VM, VMM, physical), but may also span several layers.*

This means that events detected in one layer may trigger reactions in other layers. The cross-layer approach by coordinating security mechanisms who are not aware of one another, improves security by helping to capture the overall extent of an attack, often not limited to a single layer, and to respond to it with greater accuracy.

Principle 3 (Multiple Loops) *Several control loops of variable level of granularity should be defined and coordinated.*

A single loop has insufficient flexibility for supervision perimeter and does not allow trade-offs for response optimality: either local and fast, but of variable accuracy, or broader and more relevant due to more knowledge available, but slower [20]. Trade-offs with other concerns than security are also possible.

Principle 4 (Open Architecture) *Multiple detection and reaction strategies and mechanisms should be easily integrated in the architecture, to defend the system against both known and unknown threats.*

The architecture should allow to integrate simply heterogeneous off-the-shelf security components, to support the widest possible array possible of defense mechanisms.

To build a self-protecting cloud, three broad classes of solutions are available:

1. Solutions to *protect the VM*;
2. Architectures to *secure the hypervisor*;
3. And *generic tools for detection and reaction*, independent from those two layers.

We give an overview below of each family of techniques, and how well they satisfy design principles 1–4.

4.2 Protecting VMs

Virtual machine introspection [21] sparked a whole stream of research to use the capabilities of the hypervisor to supervise VM behaviors, such as detecting integrity violations. Different alternatives have been proposed to place the monitoring component: *embedded in the VM*, *in the VMM*, or *in an "out-of-VM" appliance*.

"In-VM" placement benefits from proximity with the monitored resources to increase detection accuracy. However, placing an agent in the VM to verify its code may compromise stealth and transparency: programs co-located within the VM may detect the security component, and possibly alter its behavior.

Appliances greatly contribute to improving security: as the protection mechanism does not share the same execution context as the protection target, the former is less likely to be vulnerable if the latter is under attack. Peformance is generally enhanced for similar reasons. For instance, the well-known "anti-virus storm" phenomenon (which may induce VM resource starvation in case of extensive scans for malicious code) may be avoided if the detection mechanism is not located in the VM to protect.

Alternatively, *hypervisor* mechanisms offer transparent and privileged access to resources of the VM to monitor. For instance, a number of malware detection tools intercept system calls, and compare monitoring information gathered from different layers [16, 17]. Unfortunately, such efforts are mostly focused on detection with almost no (or very simple) remediation policies (e.g., restart, kill a VM) [16]. Some systems [13] based on a trusted VMM allow verification of flexible integrity policies. Overall, the corresponding architectures generally prove difficult to be compatible with legacy anti-malware software. A number of reaction mechanisms have also been proposed, mainly in terms of firewalls [25] or self-recovery mechanisms after an intrusion [14]. But few of them have self-protection loops or flexible security policies.

4.3 Hypervisor Defense

One layer below, a variety of techniques were proposed to protect the VMM from subversion, with special attention to buggy or malicious device drivers which enable kernel exploitation due to poor confinement. *Trusted computing architectures* [2, 26] provide strong VMM code integrity and authenticity guarantees using hardware mechanisms. VMM capabilities may be proven to third parties using attestation protocols, therefore allowing to build trust. Those techniques are limited to detection in the form of integrity measurement, without possibilities of remediation in case of violation. For instance, [6] describes a solution controlling integrity of several components of a IaaS infrastructure, based on the TPM (Trusted Platform Module) [5] and on a secure hypervisor [25].

Sandboxing techniques [12] were also heavily explored to confine malicious code by controlling communications between driver and device, kernel, or user space. Many techniques are based on isolating software faults or intercepting system calls. This class of mechanisms is expected to provide strong security if driver code is confined strictly. Unfortunately, the code of the reference monitor enforcing access control remains difficult to protect without any additional hardware mechanism.

Virtualizing some components of the VMM [30] also strengthens isolation, without requiring modifications to existing code. Partial VMM virtualization increases architecture modularity, going in the same direction as new evolutions regarding "disaggregated" VMM security architectures (coming from micro-kernels and component-based architectures) [8, 28]: simpler VMM integrity verification is achieved through a thin core hypervisor providing strong isolation by design, which can be simply formally proven and audited. Unfortunately, those new architectures are not yet compatible with mainstream hypervisors. However, this could change in the near future [22].

A number of *language-based techniques* [34] have also been proposed to protect VMM control flow by detecting safety violations through type systems. Unfortunately, this is usually limited to programming errors, malicious code being hardly considered.

Overall, solutions remain limited either to pure integrity detection [2, 34] or simple containment [12, 30], proposing no actions to sanitize the kernel. Security policies are also often not well separated from the interception mechanisms themselves, making them hard-to-manage. Security mechanisms usually require extensive code rewriting, making them hard to apply to legacy hypervisors. VMM self-protection is thus still a widely unchartered area.

4.4 Detecting Intrusions and Malware

Finally, an extensive number of generic techniques have been explored to detect and prevent intrusions (IDPS) and fight against malware [3, 33], leveraging virtualization to mitigate both known and unknown attacks [24]. Those systems are usually based on a single control loop, with a few attempts at cross-layering to detect elusive malwares. Their architecture is usually more open to allow selection and composition of several detection algorithms to improve accuracy. However, in most cases, they have been little applied to the cloud.

4.5 Towards a Big Picture

Table 1 gives a coverage estimation of principles 1–4 for the classes of mechanisms described previously. One tends to see that, although the use of policy-based design can still be improved, it is already present in tools to protect VMs and to fight against intrusions. Multi-layer defense remains however for the moment limited to the VM protection arena. Existing mechanisms still offer little flexibility in terms of security supervision granularity or openness – even though IDPS systems have made considerable progress in such directions in recent years. Unsurprisingly, hypervisor defense appears as the area where almost everything remains to be invented to realize self-protection in the virtualization layer, despite the wide range of techniques already explored.

Class of Mechanisms	Principle Satisfied?			
	Principle 1	**Principle 2**	**Principle 3**	**Principle 4**
Protection of VMs	A few	Yes	No	No
Hypervisor Defense	No	A few	No	No
IDPS / Anti-Malware	Yes	No	A few	Yes

Table 1. Principle Coverage by Some Classes of Security Mechanisms.

5 Open Issues and Perspectives

We conclude by discussing a number of research issues that still need to be explored (and solutions found!) to reach truly mature self-defending clouds. The remaining challenges are analyzed according to each step of the autonomic security loop: detection, reaction, and decision-making.

5.1 Detection

Here, "the" hard issue is the *placement of security mechanisms* in the infrastructure. This problem may be addressed *horizontally*, i.e., where in the security architecture (network or host) are protection mechanisms most relevant? It may also be considered *vertically*, i.e., in which infrastructure layers are they most effective?

- *Horizontal placement* has been traditionally addressed by detecting intrusions, either from network and host perspectives with: *Network Intrusion Detection Systems (NIDS)* to analyze network traffic; and *Host-Based Intrusion Detection Systems (HIDS)* to collect, correlate, and analyze system information on each host.
  NIDS sensors are spread throughout the network to get a distributed picture of the attack perimeter. This approach is fine for network threats, but could overlook malicious traffic between VMs on the same physical host.
  An HIDS provides precise audit data, which facilitates knowledge of the attack context. But, monitoring and monitored system often reside on the same host, which has performance and security impacts.
  Hybrid monitoring architectures orchestrating detection elements in the network and on hosts should bring many benefits, but are still lacking today. The situation is similar for emerging multi-cloud environments for which there is currently no integrated detection architecture.
- *Vertical placement* is an issue that was magnified with virtualization, a choice becoming possible between *virtual* (e.g., self-defending VMs), *virtualization* (e.g., VM introspection-based architectures), and *physical layers* to set the detection components. This choice depends on the stakeholders operating the layers (e.g., Cloud provider or customer). It may also vary according to: (1) their objectives regarding cloud infrastructure security management; (2) the level of openness provided to other parties for layer resource management; (3) the Cloud model. As for the horizontal case, hybrid architectures federating detection components in the different layers are promising, but have up-to-now little been explored.

To reconcile both dimensions, a *multi-layered and multi-lateral detection architecture* is clearly needed. A first idea could be to build the architecture around the well-known abstraction of *security domain* for defining monitoring scopes, that can then be flexibly coordinated in the overall infrastructure. We are currently working on such an architecture for the inter-cloud setting to enable 360°, both vertical and horizontal, self-protection [32].

Another tough question is *whether the monitoring system should be co-located (in the same layer, machine, network, ...) with the protection target?* A number of "out-of-VM" monitoring appliances have already been proposed to introduce a horizontal separation [27]. Similarly, nested virtualization [4, 36] is increasingly being explored to add one or more privileged layer of protection below the hypervisor for strong sandboxing of vulnerable upper IaaS layers. More generally, this approach can be used to realize "islands" in a IaaS infrastructure. The virtualization layer is then composed of two separate sub-layers:

- The lower layer is under the control of the cloud provider, and contains a mainstream VMM;
- The upper layer is under the control of groups of users, which may install their own VMM, without changing cloud provider.

This type of design allows to introduce more modularity in the IaaS infrastructure of a provider, often deemed too monolithic [35].

5.2 Reaction

The *placement issue* of mechanisms in the infrastructure is totally symmetric to the detection case. One may also wonder *whether reaction mechanisms are truly adaptable to a changing context*, as for the most part, configurations of defense equipments remain largely static. The emerging area of *cloud networking* aims to explore a few of such adaptability issues, e.g., real-time reconfiguration of virtual networks, or on-demand chaining of network security services. Particularly important is the influence of VM migration (across data centers, across clouds). How to migrate the security state (e.g., the IP address space) consistently, securely, and efficiently is still an unsolved issue.

Overall, *hypervisor protection* is still little addressed, notably protection of device drivers which are generally their weakest point, allowing VM isolation bypass. A rich litterature in the system community is available but, unfortunately, not yet applied. Hypervisor self-defense could be a big step towards automated hardening of a layer which remains highly vulnerable. A first design in that direction may be found in [31].

Along other lines, the evolution of regulations in the cloud area induce new compliance and auditability requirements. Thus, all techniques tending to *improve the overall level of assurance* of the cloud infrastructure (such as control flow integrity mechanisms, attestation protocols, proofs of compliance, and more generally all techniques inspired from trusted computing) should be mushrooming in the near future.

5.3 Decision-Making

The main challenge for decision-making in the cloud and inter-cloud setting is how to peform multi-lateral security policy management – starting first by *how to formalize security policies*. Despite many advances, current security policies are not sufficiently flexible for such environments, which require policies spanning organizational boundaries, geographical borders, and applicable in multiple contexts, among multiple actors. Frameworks are needed for flexible policy aggregation and deployment within clouds (for detection and reaction), as well as for policy negotiation between clouds. Despite some first frameworks being available [15], interoperability between different security domains, multiple conflicting stakeholder responsibilities, and multiple jurisdictions remain major barriers. Semantic Web technologies may facilitate such interoperability by progressing towards commonly-agreed formats for security negotiation, but much work remains to be done in that area.

Among a few other hard (but essential) unsolved problems: How to authenticate communications between detection and decision, or between decision and reaction? When coordinating multiple self-protection loops, how to guarantee the stability of the result? And, how to learn from past attacks to improve security and build defenses against future threats?

5.4 Perspectives

Overall, despite many elementary mechanisms being there and great foreseen benefits, there are still quite a few challenges remaining to be solved to reach a fully automated security supervision architecture – still lacking today both for the cloud and inter-cloud settings. Thus the road may still be long towards fully-fledged cloud self-defense. Nonetheless, not all roadblocks have the same maturity level: while self-defending VMs are already running, self-defending hypervisors are still far away down the road. In the meantime, a few self-defending mechanisms may be already running in your cloud infrastructure today...

References

1. Y. Al-Nashif, A. Kumar, S. Hariri, G. Qu, Y. Luo, and F. Szidarovsky. Multi-Level Intrusion Detection System. In *International Conference on Autonomic Computing (ICAC)*, 2008.
2. A. M. Azab, P. Ning, Z. Wang, X. Jiang, X. Zhang, and N. C. Skalsky. HyperSentry: Enabling Stealthy In-Context Measurement of Hypervisor Integrity. In *ACM Conference on Computer and Communications Security (CCS)*, 2010.
3. A. Baliga, L. Iftode, and X. Chen. Automated Containment of Rootkits Attacks. *Computers & Security*, 27:323–334, 2008.
4. M. Ben-Yehuda, M. D. Day, Z. Dubitzky, M. Factor, N. Har'El, A. Gordon, A. Liguori, O. Wasserman, and B.-A. Yassour. The Turtles Project: Design and Implementation of Nested Virtualization. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2010.
5. S. Berger, R. Cáceres, K. A. Goldman, R. Perez, R. Sailer, and L. van Doorn. vTPM: Virtualizing the Trusted Platform Module. In *USENIX Security Symposium*, 2006.
6. S. Berger, R. Cáceres, D. Pendarakis, R. Sailer, E. Valdez, R. Perez, W. Schildhauer, and D. Srinivasan. TVDc: Managing Security in the Trusted Virtual Datacenter. *SIGOPS Oper. Syst. Rev.*, 42:40–47, January 2008.

7. D. Chess, C. Palmer, and S. White. Security in an Autonomic Computing Environment. *IBM Systems Journal*, 42(1):107–118, 2003.
8. P. Colp, M. Nanavati, J. Zhu, W. Aiello, G. Coker, T. Deegan, P. Loscocco, and A. Warfield. Breaking Up is Hard to Do: Security and Functionality in a Commodity Hypervisor. In *ACM Symposium on Operating Systems Principles (SOSP)*, 2011.
9. N. De Palma, D. Hagimont, F. Boyer, and L. Broto. Self-Protection in a Clustered Distributed System. *Parallel and Distributed Systems, IEEE Transactions on*, 23(2):330 –336, 2012.
10. H. Debar. *Intrusion Detection Systems – Introduction to Intrusion Detection and Analysis*. Security And Privacy In Advanced Networking Technologies (Nato Science Series / Computer and Systems Sciences). IOS Press, 2004.
11. D. Frincke, A. Wespi, and D. Zamboni. From Intrusion Detection to Self-Protection. *Comput. Netw.*, 51:1233–1238, April 2007.
12. V. Ganapathy, M. J. Renzelmann, A. Balakrishnan, M. M. Swift, and S. Jha. The Design and Implementation of Microdrivers. In *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2008.
13. T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: A Virtual Machine-Based Platform for Trusted Computing. In *ACM Symposium on Operating Systems Principles (SOSP)*, 2003.
14. A. Goel, K. Po, K. Farhadi, Z. Li, and E. de Lara. The Taser Intrusion Recovery System. In *ACM Symposium on Operating Systems Principles (SOSP)*, 2005.
15. D. A. Haidar, N. Cuppens, F. Cuppens, and H. Debar. XeNA: An Access Negotiation Framework Using XACML. *Annales des Télécommunications*, 64(1–2):155–169, 2009.
16. A. Ibrahim, J. Hamlyn-Harris, J. Grundy, and M. Almorsy. CloudSec: A Security Monitoring Appliance for Virtual Machines in the IaaS Cloud Model. In *International Conference on Network and Systems (NSS)*, 2011.
17. X. Jiang, X. Wang, and D. Xu. Stealthy Malware Detection through VMM-Based "Out-of-the-Box" Semantic View Reconstruction. *ACM Trans. Inf. Syst. Secur.*, 13:1–28, 2010.
18. J. Kephart and W. Walsh. An Artificial Intelligence Perspective on Autonomic Computing Policies. In *Fifth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY)*, 2004.
19. R. Koller, R. Rangaswami, J. Marrero, I. Hernandez, G. Smith, M. Barsilai, S. Necula, S. M. Sadjadi, T. Li, and K. Merrill. Anatomy of a Real-Time Intrusion Prevention System. In *International Conference on Autonomic Computing (ICAC)*, 2008.
20. O. Mola and M. Bauer. Towards Cloud Management by Autonomic Manager Collaboration. *Journal of Communications, Network and System Sciences*, 4(12):790–802, 2011.
21. K. Nance, M. Bishop, and B. Hay. Virtual Machine Introspection: Observation or Interference? *IEEE Security and Privacy*, 6:32–37, September 2008.
22. A. Nguyen, H. Raj, S. Rayanchu, S. Saroiu, and A. Wolman. Delusional Boot: Securing Hypervisors without Massive Re-Engineering. In *7th ACM European Conference on Computer Systems (EUROSYS)*, 2012.
23. ObjectSecurity. OpenPMF White Paper, 2011. `http://www.openpmf.org/`.
24. A. Patel, Q. Qassim, and C. Wills. A Survey of Intrusion Detection and Prevention Systems. *Information Management & Computer Security*, 18(4):277–290, 2010.
25. R. Sailer, T. Jaeger, E. Valdez, R. Caceres, R. Perez, S. Berger, J. L. Griffin, and L. v. Doorn. Building a MAC-Based Security Architecture for the Xen Open-Source Hypervisor. In *Annual Computer Security Applications Conference (ACSAC)*, 2005.
26. R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn. Design and Implementation of a TCG-Based Integrity Measurement Architecture. In *USENIX Security Symposium*, 2004.
27. D. Srinivasan, Z. Wang, X. Jiang, and D. Xu. Process Out-Grafting: An Efficient "Out-of-VM" Approach for Fine-Grained Process Execution Monitoring. In *ACM Conference on Computer and Communications Security (CCS)*, 2011.

28. U. Steinberg and B. Kauer. NOVA: A Microhypervisor-Based Secure Virtualization Architecture. In *European conference on Computer systems (EuroSys)*, 2010.
29. J. Strassner. *Policy-Based Network Management: Solutions for the Next Generation*. Morgan Kaufman, 2003.
30. L. Tan, E. Chan, R. Farivar, N. Mallick, J. Carlyle, F. David, and R. Campbell. iKernel: Isolating Buggy and Malicious Device Drivers Using Hardware Virtualization Support. In *International Symposium on Dependable, Autonomic and Secure Computing (DASC)*, 2007.
31. A. Wailly, M. Lacoste, and H. Debar. KungFuVisor: Enabling Hypervisor Self-Defense. In *EUROSYS Doctoral Workshop (EURODW)*, 2012.
32. A. Wailly, M. Lacoste, and H. Debar. VESPA: Multi-Layered Self-Protection for Cloud Resources. In *International Conference on Autonomic Computing (ICAC)*, 2012.
33. Y.-M. Wang, D. Beck, B. Vo, and C. Verbowski. Detecting Stealth Software with Strider GhostBuster. In *Conference on Dependable Systems and Networks (DSN)*, 2005.
34. Z. Wang and X. Jiang. HyperSafe: A Lightweight Approach to Provide Lifetime Hypervisor Control-Flow Integrity. In *IEEE Symposium on Security and Privacy*, 2010.
35. D. Williams, H. Jamjoom, and H. Weatherspoon. The Xen-Blanket: Virtualize Once, Run Everywhere. In *7th ACM European Conference on Computer Systems (EUROSYS)*, 2012.
36. F. Zhang, J. Chen, H. Chen, and B. Zang. CloudVisor: Retrofitting Protection of Virtual Machines in Multi-Tenant Cloud with Nested Virtualization. In *ACM Symposium on Operating Systems Principles (SOSP)*, 2011.