Security Implications of Virtualization: A Literature Study

André van Cleeff, Wolter Pieters, Roel Wieringa
Information Systems Group
University of Twente
P.O. Box 217, 7500 AE Enschede, The Netherlands
{a.vancleeff,w.pieters,r.j.wieringa}@utwente.nl

Abstract— Data centers accumulate corporate and personal data at a rapid pace. Driven by economy of scale and the high bandwidth of today’s network connections, more and more businesses and individuals store their data remotely. Server virtualization is an important technology to facilitate this process, allowing dedicated hardware to be turned into resources that can be used on demand. However this technology is still under development and therefore, in spite of its increasingly important role, the overall security impact of virtualization is not yet completely known.

To remedy this situation, we have performed a systematic literature review on virtualization, and decomposed the virtualization technology into distinct features, which are dependent on each other, but also have individual positive and negative effects on security.

Our study shows that, given adequate management, the core virtualization technology has a clear positive effect on availability, but that the effect on confidentiality and integrity is less positive. Virtualized systems tend to lose the properties of location-boundness, uniqueness and monotonicity. In order to ensure corporate and private data security, we propose to either remove or tightly manage non-essential features such as introspection, rollback and transfer.

I. INTRODUCTION

Server virtualization, running many virtual servers on one physical machine, has become widespread in recent years. Benefits include reduced power consumption, lower hardware costs, as well as easier management. As for security, claimed improvements are the increased availability of applications and the isolation of processes. However, virtualization gives much more possibilities. With introspection, the inside of a virtual machine can be examined, intervention gives the ability to modify a running virtual machine. Furthermore, physical servers equipped with virtualization software can now be linked to each other, creating a whole new virtualization infrastructure, making it possible to move virtual machines from one physical server to another, while they are running. Virtualization also has security drawbacks, such as exploitable weaknesses in virtualization software, the existence of covert channels and the possibility of new types of malware. However, apart from these distinct threats, not so much is known about the overall security effect of virtualization. This is a serious issue, because virtualization is an important technology for data centers, Web 2.0 applications and new forms of on-demand or “cloud” computing. In order to understand the security impact of those technologies, it is necessary to understand the security impact of the virtualization technology as a whole. Under which circumstances does virtualization improve security, and under which does it pose a threat?

A first step towards answering these questions is to distinguish between different features of virtualization and show their interactions. In this paper we present a model consisting of five groups of features: (1) virtualization capable hardware, (2) virtual machines, (3) management of virtual machines, (4) management of physical servers running virtualization software, and (5) emergent behavior. Based on a systematic literature review, we aggregate the literature on the impact of virtualization for each feature with respect to different security properties and show how these features fit together. We conclude with an overview of how the security benefits can be maximized, from both a technical and a management point of view.

II. BACKGROUND ON VIRTUALIZATION

A. Types of virtualization

Generally, virtualization is a software layer that implements a (hardware) architecture. This layer provides a consistent interface that can be used to decouple software systems from the hardware on which they are running, making them more portable and providing easier management. Different types of components can be virtualized. For example, with resource virtualization, multiple harddisks are combined to one virtual disk, whereas with machine virtualization, the instruction set of a CPU architecture is emulated on a real physical machine. Likewise, network virtualization can use a physical switch to create virtual network compartments. With respect to virtual machines (VMs), the abstraction layer is called virtual machine monitor (VMM). The VMM controls the VMs running on top of it. There are different types of VMMs and a taxonomy can be found in Smith and Nair. [1] In their terms, the Java Virtual Machine is of type process virtual machine, because it allows individual processes to run. In contrast, XenServer is a system virtual machine because entire operating systems (guests) can be run on top of it. System virtual machines can be further split into hosted VMMs (that run on top of another host operation system) and classic VMMs (that run on the bare hardware). Often, system VMMs also involve network virtualization, as the network connections between virtual machines can be configured in the VMM. Classic
virtualization is often called server virtualization. VMMs can also be joined together, resulting in a virtualized infrastructure. In this infrastructure, capabilities such as load balancing and the transfer of virtual machines between different physical servers are managed from a central location. We call this infrastructure and its management the “VMMM”, an acronym for virtual machine monitors’ management.

In this paper, we focus on system virtual machines of the classic type, and for the remainder of this paper we will simply use the terms terms VM, VMM, VMMM, and virtualization when discussing system virtualization technologies.

B. Similarities with other technologies

VMMs resemble computer clusters, computing grids and mainframes. These systems are also comprised of multiple disks and processors, forming a logical whole. The security characteristics of virtualization are therefore also relevant for these types of computing. A notable difference is that a cluster is more often designed as a complete physical unit, whereas VMMs are created ad hoc, out of existing machines that are spread geographically. Utility or cloud computing can use any of these technologies, treating servers simply as resources, that can be used for data storage and processing, regardless of their physical location. For example, Amazon’s Elastic Compute Cloud (EC2) is known to be using the Xen VMM.

C. Usage of virtualization

To evaluate the effects of virtualization, we distinguish between different types of usage. In the literature, several types of usage are found:

1) Software testing: a testbed is created with virtual machines.
2) Software evaluation: untrusted software is evaluated in a virtual machine. The VM thus functions as a “sandbox” from which the software cannot escape.
3) Running production applications: a business’s applications are placed inside VMs.
4) Desktop virtualization: rather than giving employees physical PCs, enterprises can provide them with a personal VM running on a central server.
5) Running an intrusion detection system (IDS): here, the VM also serves as a sandbox, often called “honeypot”. Attackers breaking into the VM will not be able to go any further, and their behavior can be monitored.
6) Running cross-platform applications: an application developed for a specific OS is placed inside a VM, such that it can be run on a different OS (for example running a Windows application on a Mac).
7) Debug and replay: VMMs such as ReVirt can replay and log actions of virtual machines. When a VM is infected with a virus, its actions can be studied, simply by replaying its execution.
8) Software distribution: software can be installed on one virtual machine, which can be distributed as virtual appliance, requiring few configuration changes.

In the remainder of the paper, we concentrate on the security effects for running production applications, because here security concerns are the greatest.

III. RESEARCH APPROACH

A. Research design

The literature study is based on the method described by Webster and Watson. Here, literature on a certain topic is retrieved from well-known sources such as leading journals. After this iteration, additional literature is found by tracing back the cited papers and forward towards conferences papers that cite the journal papers. Rather than discussing each article or author separately, the findings are presented concept-centric, meaning all literature on a certain concept is discussed in one section. For this study, we began with a literature search on Scopus, yielding a total of 151 papers of which 46 were relevant. Included journals were IEEE’s Computer as well as Security & Privacy. Another notable source was the ACM VMSec’08 workshop on virtual machine security. Literature from other sources was also included, such as datasheets from virtualization product vendors such as VMware.

The results are presented centered around specific features of virtualization, linking to earlier research on feature interaction in the telecommunications domain. There, a basic phone system is extended with different features such as call forwarding and on-hold. When used together, these features can cause interactions with either desirable or undesirable consequences.

Decomposing virtualization has several benefits: firstly, the literature consists mainly of distinct security claims, arguing how a specific feature affects security in a certain context. These claims can be aggregated together for each feature. By grouping claims together for each feature, and analyzing them, we improve the understanding of the feature’s effect and can identity the more fundamental effects. Secondly, regarding those claims that are not attributed to a specific feature, but rather virtualization as a whole, we can attempt to trace their effects back to specific features. Thirdly, the feature interaction model allows us to detect previously unknown interactions between features and their security consequences.

Obviously, a complex technology such as virtualization can be split up in many ways, and each choice is to some extent arbitrary. For this model, we used the following guidelines:

1) For something to be considered a feature, it should represent a distinct piece of functionality, characteristic or architecture, having a unique impact on security, if only on availability.
2) Existing decompositions found in the literature should be used as much as possible: care was taken to limit the introduction of new terminology. In some cases we introduced features that are not previously covered in the

1 VMware’s ESX is a hybrid form because it is based on a Linux kernel, but was examined in the study.
2 An early application of virtualization was IBM’s System/360 mainframe.
literature. For example, we could not find any discussion on the power features of virtualization (functions to switch a VM on or off). These have a distinct effect on availability.

3) Features that are present in specific implementations can be excluded: we are interested in common virtualization features, not in the complete feature list of one particular vendor.

For each feature, we present the security effects, together with several key references. In case effects were listed in more than one paper, we only reference to a key publication. When no direct security effects were found, such as in the case of the power functions, we added analytical claims based on the literature studied. Obviously, these claims should be the focus of further research.

B. Conceptual model

Virtualization technology consists of features, which are divided into five groups:

1) features of virtualization capable hardware
2) features of VMs
3) features of individual VMMs
4) features of VMMMs
5) features arising from unintended interaction between features

Figure 1 shows a graphical representation of the groups. The hardware (1) enables virtualization, several VMs (2) run on top of a group of VMMMs (3) and the VMMs are managed by a VMMM(4), leading to emergent features (5).

The threat model used in this paper is essentially a white box, no component of the virtualization technology is trusted. As such, we depart from earlier threat models created by Vaarela [8] and VMware [2] that assume trusted components and are therefore unsuitable for the study of the inner workings of virtualization technology. Since we are considering securing production applications, the ultimate security objective is to protect the application running inside a VM. In our model, threats to this application can originate from five different components: (i) hardware, (ii) other VMs, (iii) VMMMs, (iv) VMMMs and (v) network.

Hardware threats are not discussed in this paper, because these are mostly generic threats such as theft that are not specifically relevant virtualization. More technical information on virtualization hardware is available from Perez et al. [9].

Combined, this leads to the following model of threats depicted in figure 2. Note that these attacks can be combined, resulting in multi-step attacks. For example the attack Network ▶ VMM is a stepping stone for a more complicated attack plan, such as Network ▶ VMM ▶ VM.

<table>
<thead>
<tr>
<th>Threat source</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network ▶ VMMM</td>
<td>An outsider attacks the VMMM</td>
</tr>
<tr>
<td>Network ▶ VM</td>
<td>An outsider attacks the VMM</td>
</tr>
<tr>
<td>Network ▶ VM</td>
<td>An outsider attacks the VM</td>
</tr>
<tr>
<td>VMMM ▶ VM</td>
<td>A VMMM attacks a VMM</td>
</tr>
<tr>
<td>VMM ▶ VM</td>
<td>A VMM attacks a VM</td>
</tr>
<tr>
<td>VM ▶ VM</td>
<td>A VM attacks another VM</td>
</tr>
</tbody>
</table>

Fig. 2. Virtualization threats

IV. SECURITY IMPACT PER FEATURE GROUP

We now list what is currently known about the security impact of all the different features per group. Security claims in the literature are split over three categories: (i) analytical claims, based on logical arguments, (ii) empirical claims, based on experiments and (iii) claims based on mathematical models. The first category was dominant. Only one paper fitted into the third category, calculating the reliability of different virtualization architectures. [10]

A. Features of hardware

We list two important hardware features below. For more detailed information on other virtualization hardware issues, we refer to Uhlig et al. [11] and Van Doorn [12].

1) Trap program execution: The essential hardware feature enabling virtualization (present in all modern x86 hardware) is the ability to trap the execution of a running process and hand over control of the CPU to the VMM. [13] This allows the VMM to intervene in the execution of the process. In such a way, the VMM can perform two critical tasks: (i) emulate certain hardware and (ii) isolate the virtual machine from other running processes.

2) Trusted Platform Module: An optional hardware feature is the Trusted Platform Module (TPM) chip, which can be used to verify that the proper VMM is indeed running, as opposed to an insecure version installed by an attacker. [9]

B. Features of VMs

1) Store VM as image: VMs consist essentially of files, on which the machine’s own data is stored, as well as some metadata (for example, the amount of memory used, or the number of CPUs). Storing this data as files allows easy copying of the data by a VMM, with the drawback that they can be inspected (breaching confidentiality) or modified (breaching integrity).

An overview of common VM data elements was created by the Distributed Management Task Force (DMTF), an IT industry consortium. [14]
2) Modified VM software: For ease of use, the VMM interface should be as transparent as possible: ideally a server’s software should not need to be changed if the server is virtualized. However, the OS running inside the VM can be modified to provide better performance (paravirtualization [3]). VMs have the ability to contact the VMM in order to execute security checks, using so-called hooks. [15]

C. Features of individual VMMs

1) Small footprint: Software layers provide new opportunities for attack and for defense. Generally, the amount of exploitable vulnerabilities is proportional to the amount of code. [16] To get an indication of a VMM’s security, it can be compared to the previous lowest layer, the operating system (OS). From this viewpoint, the VMM provides the opportunity for improved security because it is notably smaller. For example, VMWare’s ESX fits into 32 Megabytes [17], whereas a complete operating system such as Microsoft’s Windows Vista can be several Gigabytes big. Thus, the chance of an attack of a VM on an underlying VMM is successful is probably lower than the chance that an attack of an application on an OS is successful.

2) Hierarchical control: The VMM layer is designed to control the VMs that run on top of it. Therefore it should not be possible for code running inside the VM to “escape” to the VMM and gain control over it. However, several vulnerabilities have been detected in VMMs that might allow for such an escape, in laboratory experiments [18] as well as in production environments [19].

3) Isolation between processes: For normal operations, different VMs running on one VMM cannot affect each other, unless their interactions (network connections, file shares) are explicitly specified in the VMM. Such a VMM provides better isolation than an operating system, in which applications can normally interact: if we put each application in its own OS in a VM we reduce the chance of undesired interactions and intrusions. [20] However, covert channels are still an issue in existing VMM implementations. [16] A drawback is also that conventional security mechanisms depending on network access do not work well - a standard IDS cannot observe the traffic on one physical machine between different VMs.

4) Logging: Virtualization can help to implement secure logging: during the execution of the VM, the VMM collects data and stores it in a place outside of the VM. Therefore it cannot be altered by an exploit that is contained inside the VM. Such a feature is implemented in ReVirt. [5]

5) Load balancing: VMMs can determine (and limit) the CPU utilization and disk space that a VM uses. This prevents a VM from starving the other VMs of resources.

6) Copy and backup VMs: Making backups and copies of VMs is easier than making copies of data on physical machines. Therefore, a defective VM can be easily replaced by a working version.

7) Introspection: Since the VMM is placed at a lower level than the VM, it has the ability to “look inside the VM”, to see its data and monitor its execution. This process is called virtual machine introspection. [21] The functionality can be used to run security applications such as virus scanners, intrusion detection systems such as Livewire [22], and policy checkers. Because these are outside of the VM, the VM cannot interfere with them. Security problems can be reported to the management, who can shutdown or quarantine the VM.

8) Attestation: If an inspection of a VM is performed, the VMM can send the results (the attestation) to another party. Based on the results, the receiving party can authenticate the VM, and decide whether to trust it. For optimal confidence, this function is best used in combination with trusted hardware (as in the Terra VMM [23]).

9) Interference: In some cases intrusions cannot just be detected, but intervention may be possible: Once the VMM detects a malicious program running inside a VM, its memory can be altered, preventing its execution. [21]

10) Power functions: VMMs control the execution of a VM, as if it would have a virtual power button. Commonly, several functions are provided: start/power on, stop/power off, pause/suspend and reboot/reset. These functions are useful to increase availability, for example if a VM crashes, it can be more easily rebooted than if it were a physical machine, and it is also easier to limit resource usage by simply pausing VMs that are not needed.

11) Networking: VMMs can configure the network, so that a VM only communicates with predefined VMs, forming a private network that does not require hardware changes. The networking functions can also be used to monitor traffic, for example running an intrusion detection system.

12) Rollback: Another feature that can be provided by a VMM is to rollback actions of a VM. [24] For example Microsoft Hyper-V has a feature to create a snapshot. [25] If a problem is detected, the VM can be restored to an earlier state.

13) VM Management: In order to manage the previously described features, a user interface has to be provided to control the VMM and the VMs that it runs. The management can be done from another machine (virtual or physical). For example Xen is designed to use a specific VM (“Dom0”) for its management. [16] Obviously, this enlarges the small footprint that was supposed to be a strong asset of the VMM.

D. Features of VMMMs

1) Transfer: A crucial feature of a VMM is the ability to transfer ("migrate") running virtual machines between physical servers. If a physical machine undergoes maintenance, the virtual machine running on top of it can be temporarily moved to another physical machine. Several VMMs provide this capacity (VMware ESX [26], XenServer [27]).

A problem is that the network ID (often the MAC address) is reassigned randomly after the transfer, making it difficult to identify the VM. [24] Some VMMs can retain the MAC address during a live transfer. [20] Oberheide et al. [28] demonstrated how to exploit the transfer feature by modifying the VM in transit. Improper management can also lead to the transfer of VMs to unsecure

...
hosts, to breaches of confidentiality and to denial-of-service (DOS) attacks if too many virtual machines are migrated to one VMM host.

2) Replication: Apart from being transferred, VMs can also be replicated on different physical servers. This is useful to ward off a DOS attack, to distribute workload and to cope with hardware failures. At the downside, replication and transfer can be used to have VMs run on less secure locations, as was the case for the transfer feature. Obviously, the security of a VM is determined by all the physical hosts it has run on.

3) Load balancing: Features such as transfer and replication can be used for load balancing across different physical machines, increasing the availability of applications.

4) Patching: Before virtualization, efficient use of resources meant having multiple applications running on one operating system. Since operating systems isolate applications poorly from each other, a common approach is to put each application in a separate VM, having a dedicated operating system. This increases the number of operating systems, because deployment of VMs is easy. Therefore, the process of managing operating system patches becomes more critical.

A benefit of VMMs is that they can contain software to ease the process of patching, such as VMware’s vCenter Update Manager. This software inspects VMs to check whether they are fully patched. If patching is necessary, a snapshot can be made of the system, after which the patches are installed. If the patching fails, the VM can revert to the snapshot.

VMMs make it therefore extremely easy to rollback patches. Thus, a failed patch can be removed, but this also holds for a successful patch. Normally patching is a monotonic process: patches are applied in sequential order. This is not necessarily true when a VMM is used: a VM that was patched on a certain date might have its patch removed at a later date. At any given time, not all of the existing VMs are running, some might be dormant for days or even weeks, depending on the need. If such a VM is started, it will lack the critical patches.

When applications cannot be patched, they need to be isolated from the network. Some VMMs provide specific functionality for this case (for example ServerShield).

An additional issue is that VMMs themselves also have to be patched, and this is more critical - if a VMM is breached, the breach might be impossible to detect because the VMM is the lowest layer of the IT stack.

5) VMM management: If several VMMs are linked together, their work needs to be coordinated from a special server, such as vCenter from VMware. This gives the administrators enormous power, since they can control any VMM and any VM running on it from a single point. Another side effect is that the trusted code base is increased (similar to the management domain for a single VM).

E. Features emerging from interactions

In the past sections, we have discussed many virtualization features, some of which have a positive effect on security, whereas others have a negative impact. In this section, we aggregate these into three emerging features that were not explicitly designed, but rather evolved from the interaction between existing features. This high-level clustering helps to understand the effects.

1) Loss of uniqueness of machines and data: In a non-virtualized server environment, applications, servers and data are to a great extent unique. However, the replication and copy/backup features reduce the uniqueness of these.

2) Loss of location-boundedness of data: It is difficult to ascertain the location of a certain VM, since it can move between different physical servers, due to features such as transfer, replication and backup. In fact this also holds for the virtual location of a VM: if the networks are also virtualized, a VM can be accidentally dragged-and-dropped outside of a DMZ.

3) Loss of monotonicity of program execution: Virtualization technology causes a server’s history to stop being a straight line. Instead it becomes a graph, where branches are made on replication and copy operations, and a previous state can be reached when a restore is performed. Data cannot be deleted easily, there can be many copies and the VM can be restored to an earlier version.

V. OVERALL IMPACT OF VIRTUALIZATION

The overall effects of virtualization technology depend not just on the technology itself, but also on the environment in which it is used. Conceptually we therefore split the causes into three groups:

1) the technical capabilities of features and the dependencies between them
2) the selection of features that can be made in practice
3) the management of the selected features in practice

The first aspect is illustrated in figure 4 where a summary of all features and their effect on security is provided. For each feature, we show its known effect on all five security properties. A + sign indicates that the functionality provided by the feature has a positive effect on security (compared to a non-virtualized situation), without placing high demands on the environment regarding management, whereas a − indicates the opposite: the feature is technically vulnerable or can be easily misused. Furthermore a ± indicates that the security effect depends on the particular implementation or the circumstances, or that the literature indicates both positive and negative effects. Finally an empty cell indicates that we could not find or deduce an effect on a security property. In the threats column of figure 4, we can find which feature requires strict management because it can be easily abused.

The dependencies between features in figure 3 shed light on the second group, showing which features can be easily altered or removed. As for the precise meaning of feature dependencies, obviously we do not have the source code available for all virtualization technologies to investigate the real technical dependencies such as system calls. Therefore we take a more practical approach: Given two features A and B, A depends on B when logically it cannot function independently from B without duplicating part of B’s functionality. To
A technological weakness can be mitigated by strong procedural security, and vice versa can a secure technology be misused in practice. Taking this into account, we have attempted to present the overall security impact of virtualization, as can be expected in practice for each of the five security properties in the next sections.

A. Confidentiality

Virtualization threatens confidentiality in several ways. First, the introspection feature gives the VMM the ability to look inside the VM. This feature can be misused or attacked. The problem is aggravated by the fact that VMs can be transferred between different physical servers: after several transfers, the confidentiality of the data might depend on the security of many different physical servers, because data is retained there in some form. Obviously, replication features only make this problem worse.

B. Integrity

Together with the features of intervention and rollback comes the ability to manipulate the state of a VM, which threatens the integrity of the transactions done on a VM.

C. Availability

The availability of applications running on VMs will most likely improve: VMs can be transferred for maintenance purposes, restored when a failure occurs and replicated if the workload requires it. (Cf. Jansen et al. [10]) However it must be noted that virtualization is not without availability risks, as was demonstrated by an attack on the virtualization infrastructure of the web hosting company VAServe, where 100,000 sites were deleted [33].

D. Authenticity

Virtualization creates identification problems. If a VM is duplicated there no longer is one original machine. Also, the identity of the machine used for communications (such as the MAC address) might change during a transfer. A possible solution is to alter a VM in such a way that it depends on a special piece of hardware, possibly using physically unclonable functions, as proposed by Atallah et al. [34].

E. Non-repudiation

Since virtual machines can be duplicated, rolled back and restored, there seems to be a fundamental problem regarding non-repudiation. If evidence of transactions is stored in a VM in the form of a transaction log, this can be lost if the state of a VM is restored. If transactions are signed, the key with which this is done is also stored on the VM, and can thus be copied. Therefore, even if the digital signature appears valid, we are not entirely sure which machine actually put the signature.

VI. Conclusions

This paper presents the results of a literature study on the effects of virtualization technology. As a whole, virtualization has positive effects on availability, but at the same time it threatens confidentiality, integrity, authenticity and non-repudiation, even though many features are designed with these goals in mind.

Some core features of virtualization (isolation between processes and small footprint) have clear positive effects on most security properties, but it is clear that these effects are mitigated by newer features built on top of these. These additional features make the VMM layer bloated [13], and create an additional level of complexity, causing three problems: first, they increase the amount of code and therefore the likelihood of bugs, which is (amongst other things) what makes operating system security so hard to achieve. Secondly, they make misuse more easy by giving much more control to the administrators. Thirdly, they lead to the emergent properties of loss of uniqueness, location boundedness and monotonicity.

Especially because the technology is widely deployed, these security problems are surprising, which might underscore the need for further research, especially in the form of case studies regarding current practices in enterprises, for example examining how VM integrity is maintained in practice.

However, whatever the outcome of these case studies might be, we believe that there is a fundamental limit as to what safely can be virtualized, in order to have adequate security. If we imagine a completely virtualized system, where the audit trails of the VMMs, the identity of the administrators, and the DNS servers no longer have a fixed physical basis, there is no single system from which to build trust as all systems becomes fluid. This should be a core concern for everyone building data centers, clouds or running web applications on top of them.

A. Recommendations

In this final section, several recommendations are discussed, concerning how to develop and use virtualization technology with security in mind.

1) Virtualization technology design: We propose to limit the possibilities for introspection and intervention of VMMs, such that they cannot affect the application (threatening integrity) and cannot steal data from them (threatening confidentiality). Especially in outsourcing scenarios this is useful, because the insourcer should not have full control over the outsourcer’s data. If virtualization really is just an infrastructure layer, its functionality can simply focus on availability. In such a situation, the VM applications “float” on top of the virtualization layer, that only sustains it without interfering.

An alternative is to split features between different VMMs running on different physical machines. The majority of VMMs then focus on availability, allowing high availability of the applications, while a second category of VMMs has extensive intervention and introspection features, and contribute to increased confidentiality and integrity. The latter VMMs would be only be capable of checking the VM and conferring
the result to a VMMM, for example to run a virus scan, without being able to transfer a VM image, being limited by the physical design of the network. Such a setup would minimize the impact of security breaches.

2) Application design: Earlier we noted that applications inside VMs are typically programmed to run continuously, as they are when running on physical machines. Considering the problems of bookkeeping in a virtualized infrastructure, the question can be raised whether applications running inside VMs should be redesigned to facilitate batch processing and checkpointing. A single run of a VM (from begin to end of a job) can then be validated as a whole.

3) Virtualization deployment and management: As a whole, the virtualization technology has many features in common with operating systems, but is generally much more powerful. An individual operating system can do load balancing on one physical machine, controlling several CPUs and processes, but a VMMM exerts control over many physical servers. Operating systems can create restore points and rollback patches, but only with virtualization is a complete rollback possible, which leaves no trace of the previous state. Operating systems can also debug applications and run virus scans, but only virtualization allows for the complete inspection of virtual machines, with all its data and all its processes.

If anything, the management of this infrastructure should therefore be more strictly organized than a physical one. Changes are easier to execute and harder to keep track of. Gigabytes of VM data are flowing between physical servers and a stolen VM image can be readily used elsewhere.

In fact, the VMM infrastructure should be considered to be a complete machine in itself, similar to a mainframe, of which all the parts are connected. Such a machine should be put in one physical location, with tight security controls.

ACKNOWLEDGMENT

This research is/was supported by the research program Sentinels (www.sentinels.nl). Sentinels is being financed by Technology Foundation STW, the Netherlands Organization for Scientific Research (NWO), and the Dutch Ministry of Economic Affairs under project number TIT.7628.

REFERENCES

<table>
<thead>
<tr>
<th>Group</th>
<th>Feature</th>
<th>Benefits</th>
<th>Threats</th>
<th>Vulnerabilities</th>
<th>Attacks</th>
<th>Confidentiality</th>
<th>Integrity</th>
<th>Availability</th>
<th>Non-repudiation</th>
<th>Authenticity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware</td>
<td>trap program execution</td>
<td>facilitate</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>TPM</td>
<td>store VM as image</td>
<td>backup VM</td>
<td>VM image</td>
<td>software</td>
<td>VMMM ▶ VM</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>VM</td>
<td>modified VM software</td>
<td>security checks</td>
<td>attack VMMM</td>
<td>software</td>
<td>VMMM ▶ VMMM</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>VMM</td>
<td>small footprint</td>
<td>fewer vulnerabilities</td>
<td>VMM rootkit</td>
<td>software</td>
<td>VMMM ▶ VM</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>hierarchical control</td>
<td>control untrusted VM</td>
<td>enlarged footprint, VM escape</td>
<td>Software</td>
<td>VMMM ▶ VMMM</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>isolation between processes</td>
<td>isolate untrusted VM</td>
<td>covert channels</td>
<td>VMMM ▶ VM</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td></td>
<td>logging</td>
<td>store log securely</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>load balancing</td>
<td>prevent DOS</td>
<td>N/A</td>
<td>software</td>
<td>VMMM ▶ VM</td>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>copy and backup VMs</td>
<td>facilitate backup</td>
<td>VM branching</td>
<td>management</td>
<td>N/A</td>
<td>-</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>introspection</td>
<td>virus scan,</td>
<td>introspection misuse</td>
<td>software</td>
<td>VMMM ▶ VM</td>
<td>±</td>
<td>±</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>attestation</td>
<td>authenticate VM</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>interference</td>
<td>prevent and stop</td>
<td>intervention misuse</td>
<td>software</td>
<td>VMMM ▶ VM</td>
<td>±</td>
<td>±</td>
<td>±</td>
<td>±</td>
<td>±</td>
</tr>
<tr>
<td></td>
<td>power functions</td>
<td>recover from errors</td>
<td>sleeper-exploit</td>
<td>management</td>
<td>VMMM ▶ VM</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>networking</td>
<td>isolation</td>
<td>network traffic snopes</td>
<td>software</td>
<td>VMMM ▶ VM,</td>
<td>±</td>
<td>±</td>
<td>±</td>
<td>±</td>
<td>±</td>
</tr>
<tr>
<td></td>
<td>rollback</td>
<td>rollback illegal</td>
<td>rollback patch</td>
<td>management</td>
<td>VMMM ▶ VM</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>VM management</td>
<td>facilitate control</td>
<td>abuse</td>
<td>management</td>
<td>±</td>
<td>±</td>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VMM</td>
<td>transfer</td>
<td>migrate if error</td>
<td>DOS, in-transfer modification, transfer off-site</td>
<td>management</td>
<td>Network ▶ VM, VMMM ▶ VM</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>replication</td>
<td>anti DOS</td>
<td>clone, replicate</td>
<td>management</td>
<td>VMMM ▶ VMMM</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>load balancing</td>
<td>anti DOS</td>
<td>abuse</td>
<td>management</td>
<td>VMMM ▶ VMMM</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>patching</td>
<td>facilitate patching</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>±</td>
<td>±</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VMM management</td>
<td>facilitate control</td>
<td>abuse</td>
<td>VMMM ▶ VMMM</td>
<td>±</td>
<td>±</td>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>emergent</td>
<td>loss of uniqueness</td>
<td>N/A</td>
<td>exploits</td>
<td>management</td>
<td>N/A</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>loss of location-boundness</td>
<td>N/A</td>
<td>exploits</td>
<td>management</td>
<td>N/A</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>loss of monotonicity</td>
<td>N/A</td>
<td>exploits</td>
<td>management</td>
<td>N/A</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 4. Virtualization threats


