We often hear that virtualization enhances security because a virtual machine monitor or hypervisor is

considerably smaller than an operating system. For example, the XenVMMdiscussed in Section 5.8 has

approximately 60,000 lines of code, one to two orders of magnitude fewer than a traditional operating

system.14

Ahypervisor supports stronger isolation between theVMs running under it than the isolation between

processes supported by a traditional operating system.Yet the hypervisormust rely on amanagement OS

to create VMs and to transfer data in and out from a guest VMto storage devices and network interfaces.

A small VMM can be carefully analyzed; thus, one could conclude that the security risks in a virtual

environment are diminished.We have to be cautious with such sweeping statements. Indeed, the trusted

computer base (TCB)15 of a cloud computing environment includes not only the hypervisor but also

the management OS. The management OS supports administrative tools, live migration, device drivers,

and device emulators.

For example, the TCB of an environment based on Xen includes not only the hardware and the

hypervisor but also the management operating system running in the so-called Dom0 (see Figure 9.3).

System vulnerabilities can be introduced by both software components, Xen, and the management

operating system. An analysis of Xen vulnerabilities reports that 21 of the 23 attacks were against

service components of the control VM [90]; 11 attacks were attributed to problems in the guest OS

caused by buffer overflow16 and 8 were denial-of-service attacks.

Dom0 manages the building of all user domains (DomU), a process consisting of several steps:

1. Allocate memory in the Dom0 address space and load the kernel of the guest operating system from

secondary storage.

2. Allocate memory for the new VM and use foreign mapping17 to load the kernel to the new VM.

3. Set up the initial page tables for the new VM.

4. Release the foreign mapping on the new VM memory, set up the virtual CPU registers, and launch

the new VM.

A malicious Dom0 can play several nasty tricks at the time when it creates a DomU [215]:

• Refuse to carry out the steps necessary to start the new VM, an action that can be considered a

denial-of-service attack.

• Modify the kernel of the guest operating system in ways that will allow a third party to monitor and

control the execution of applications running under the new VM.

• Undermine the integrity of the new VM by setting the wrong page tables and/or setting up incorrect

virtual CPU registers.

• Refuse to release the foreign mapping and access the memory while the new VM is running.

Let us now turn our attention to the run-time interaction between Dom0 and a DomU. Recall that

Dom0 exposes a set of abstract devices to the guest operating systems using split drivers. The front end

of such a driver is in the DomU and its back end in Dom0, and the two communicate via a ring in shared

memory (see Section 5.8).

In the original implementation of Xen a service running in a DomU sends data to or receives data

from a client located outside the cloud using a network interface in Dom0; it transfers the data to I/O

devices using a device driver in Dom0.18 Therefore, we have to ensure that run-time communication through Dom0 is encrypted. Yet, Transport Layer Security (TLS) does not guarantee that Dom0 cannot

extract cryptographic keys from the memory of the OS and applications running in DomU.

A significant security weakness of Dom0 is that the entire state of the system is maintained by

XenStore (see Section 5.8). A malicious VM can deny access to this critical element of the system to

other VMs; it can also gain access to the memory of a DomU. This brings us to additional requirements

for confidentiality and integrity imposed on Dom0.

Dom0 should be prohibited from using foreign mapping for sharing memory with a DomU unless

a DomU initiates the procedure in response to a hypercall from Dom0. When this happens, Dom0

should be provided with an encrypted copy of the memory pages and of the virtual CPU registers. The

entire process should be closely monitored by the hypervisor, which, after the access, should check the

integrity of the affected DomU.

A virtualization architecture that guarantees confidentiality, integrity, and availability for the TCB of

a Xen-based system is presented in [215]. A secure environment when Dom0 cannot be trusted can only

be ensured if the guest application is able to store, communicate, and process data safely. Thus, the guest

software should have access to secure secondary storage on a remote storage server for keeping sensitive

data and network interfaces to communicate with the user. We also need a secure run-time system.

To implement a secure run-time system we have to intercept and control the hypercalls used for communication

between a Dom0 that cannot be trusted and a DomU wewant to protect. Hypercalls issued by

Dom0 that do not read or write to the memory of a DomU or to its virtual registers should be allowed.

Other hypercalls should be restricted either completely or during specific time windows. For example,

hypercalls used by Dom0 for debugging or for the control of the IOMMU19 should be prohibited.

We cannot restrict some of the hypercalls issued by Dom0, even though they can be harmful to the

security of a DomU. For example, foreign mapping and access to the virtual registers are needed to save

and restore the state of a DomU. We should check the integrity of a DomU after the execution of such

security-critical hypercalls.

New hypercalls are necessary to protect:

• The privacy and integrity of the virtual CPU of a VM. When Dom0 wants to save the state of the

VM, the hypercall should be intercepted and the contents of the virtual CPU registers should be

encrypted. When a DomU is restored, the virtual CPU context should be decrypted and then an

integrity check should be carried out.

• The privacy and integrity of the VM virtual memory. The page table update hypercall should be

intercepted and the page should be encrypted so that Dom0 handles only encrypted pages of the

VM. To guarantee integrity, the hypervisor should calculate a hash of all the memory pages before

they are saved by Dom0. Because a restored DomU may be allocated a different memory region, an

address translation is necessary (see [215]).

• The freshness of the virtual CPU and the memory of the VM. The solution is to add to the hash a

version number.

As expected, the increased level of security and privacy leads to increased overhead. Measurements

reported in [215] show increases by factors of 1.7 to 2.3 for the domain build time, 1.3 to 1.5 for the

domain save time, and 1.7 to 1.9 for the domain restore time.