Xoar is a modified version of Xen that is designed to boost system security [90]. The security model

of Xoar assumes that the system is professionally managed and that privileged access to the system

is granted only to system administrators. The model also assumes that the administrators have neither

financial incentives nor the desire to violate the trust of the user. The security threats come from a guest

VM that could attempt to violate the data integrity or the confidentiality of another guest VM on the

same platform or exploit the code of the guest. Another source of threats are bugs in the initialization

code of the management virtual machine.

Xoar is based on microkernel20 design principles. Xoar modularity makes exposure to risk explicit

and allows guests to configure access to services based on their needs. Modularity allows the designers

of Xoar to reduce the size of the system’s permanent footprint and increase the level of security of

critical components. The ability to record a secure audit log is another critical function of a hypervisor

facilitated by a modular design. The design goals of Xoar are:

• Maintain the functionality provided by Xen.

• Ensure transparency with existing management and VM interfaces.

• Maintain tight control of privileges; each component should only have the privileges required by its

function.

• Minimize the interfaces of all components to reduce the possibility that a component can be used

by an attacker.

• Eliminate sharing and make sharing explicit whenever it cannot be eliminated to allow meaningful

logging and auditing.

• Reduce the opportunity of an attack targeting a system component by limiting the time window

when the component runs.

These design principles aim to break the monolithic TCB design of a Xen-based system. Inevitably,

this strategy has an impact on performance, but the implementation attempted to keep the modularization

overhead to a minimum.

A close analysis shows that booting the system is a complex activity, but the fairly large modules

used during booting are no longer needed once the system is up and running. In Section 5.8 we saw that

XenStore is a critical system component because it maintains the state of the system; thus, it is a prime

candidate for hardening. The ToolStack is only used for management functions and can only be loaded

upon request.

The Xoar system has four types of components: permanent, self-destructing, restarted upon request,

and restarted on timer (see Figure 9.4):

1. Permanent components. XenStore-State maintains all information regarding the state of the system.

2. Components used to boot the system. These components self-destruct before any user VMis started.

Two components discover the hardware configuration of the server, including the PCI drivers, and

then boot the system:

• PCIBack. Virtualizes access to PCI bus configuration.

• Bootstrapper. Coordinates booting of the system.

3. Components restarted on each request:

• XenStore-Logic.

• Toolstack. Handles VM management requests, e.g., it requests the Builder to create a new guest

VM in response to a user request.

• Builder. Initiates user VMs.

4. Components restarted on a timer. Two components export physical storage device drivers and the

physical network driver to a guest VM:

• Blk-Back. Exports physical storage device drivers using udev21 rules.

• NetBack. Exports the physical network driver.

Another component, QEMU, is responsible for device emulation. Bootstrapper, PCIBack, and Builder

are the most privileged components, but the first two are destroyed once Xoar is initialized, and the Builder is very small; it consists of only 13,000 lines of code. XenStore is broken into two components:

XenStore-Logic and XenStore-State. Access control checks are done by a small monitor module in

XenStore-State. Guest virtual machines share only the Builder, XenStore-Logic, and XenStore-State

(see Figure 9.5).

Users of Xoar are able to only share service VMs with guest VMs that they control. To do so, they

specify a tag on all the devices of their hosted VMs. Auditing is more secure; whenever aVMis created,

deleted, stopped, or restarted by Xoar, the action is recorded in an append-only database on a different

server accessible via a secure channel.

Rebooting provides the means to ensure that a virtual machine is in a known-good state. To reduce

the overhead and the increased start-up time demanded by a reboot, Xoar uses snapshots instead of

rebooting. The service VM snapshots itself when it is ready to service a request; similarly, snapshots of

all components are taken immediately after their initialization and before they start interacting with other

services or guest VMs. Snapshots are implemented using a copy-on-write mechanism22 to preserve any

page about to be modified.