sVirt: Hardening Linux Virtualization with Mandatory Access Control

James Morris
Red Hat Security Engineering

Linux.conf.au 2009
Hobart, Australia

Goal:

Improve security for Linux virtualization

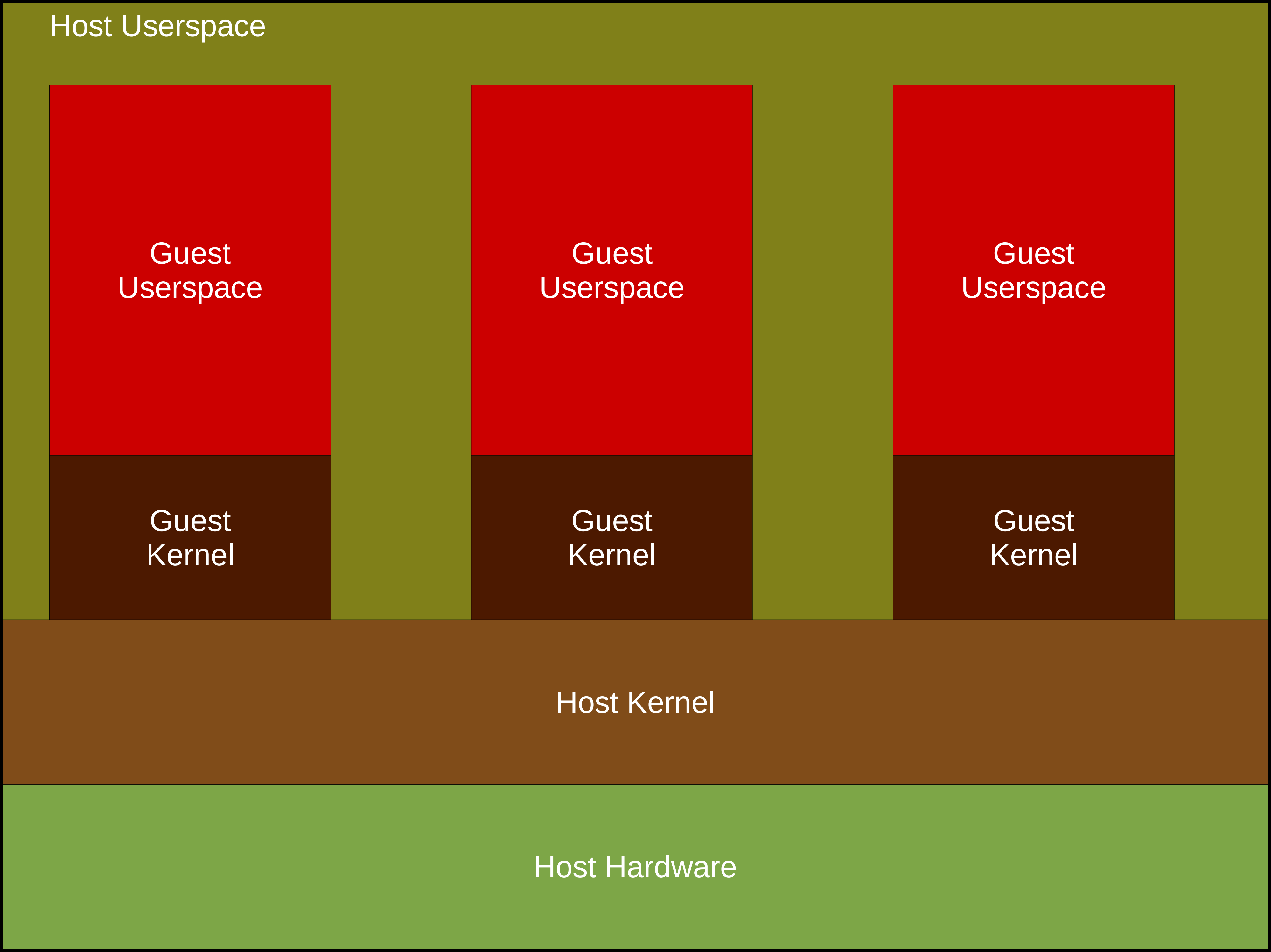
Linux Virtualization:

Where the “hypervisor” is a normal Linux process

KVM

Lguest

UML


Host Userspace
Guest Userspace
Guest Userspace
Guest Userspace
Guest Kernel
Guest Kernel
Guest Kernel
Host Kernel
Host Hardware

Utilize existing process-based security mechanisms

DAC is not enough:

Subjects can modify own security policy

Mandatory Access Control (MAC):

Subjects cannot bypass security policy

Virtualization Threat Model

(work in progress)

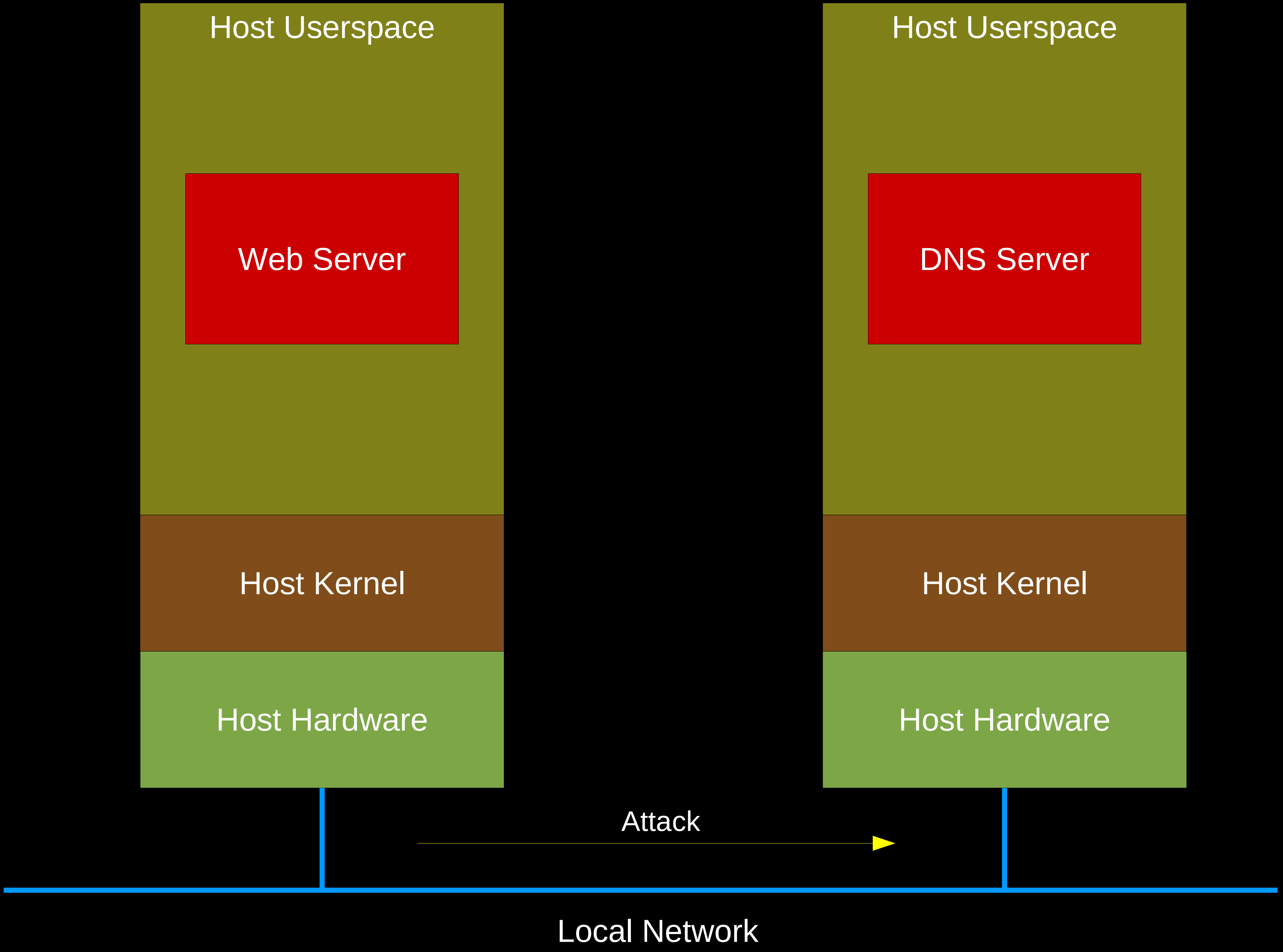
Virtualization introduces new security risks

Flawed hypervisor:

Malicious guest breaks out, attacks other guests or host

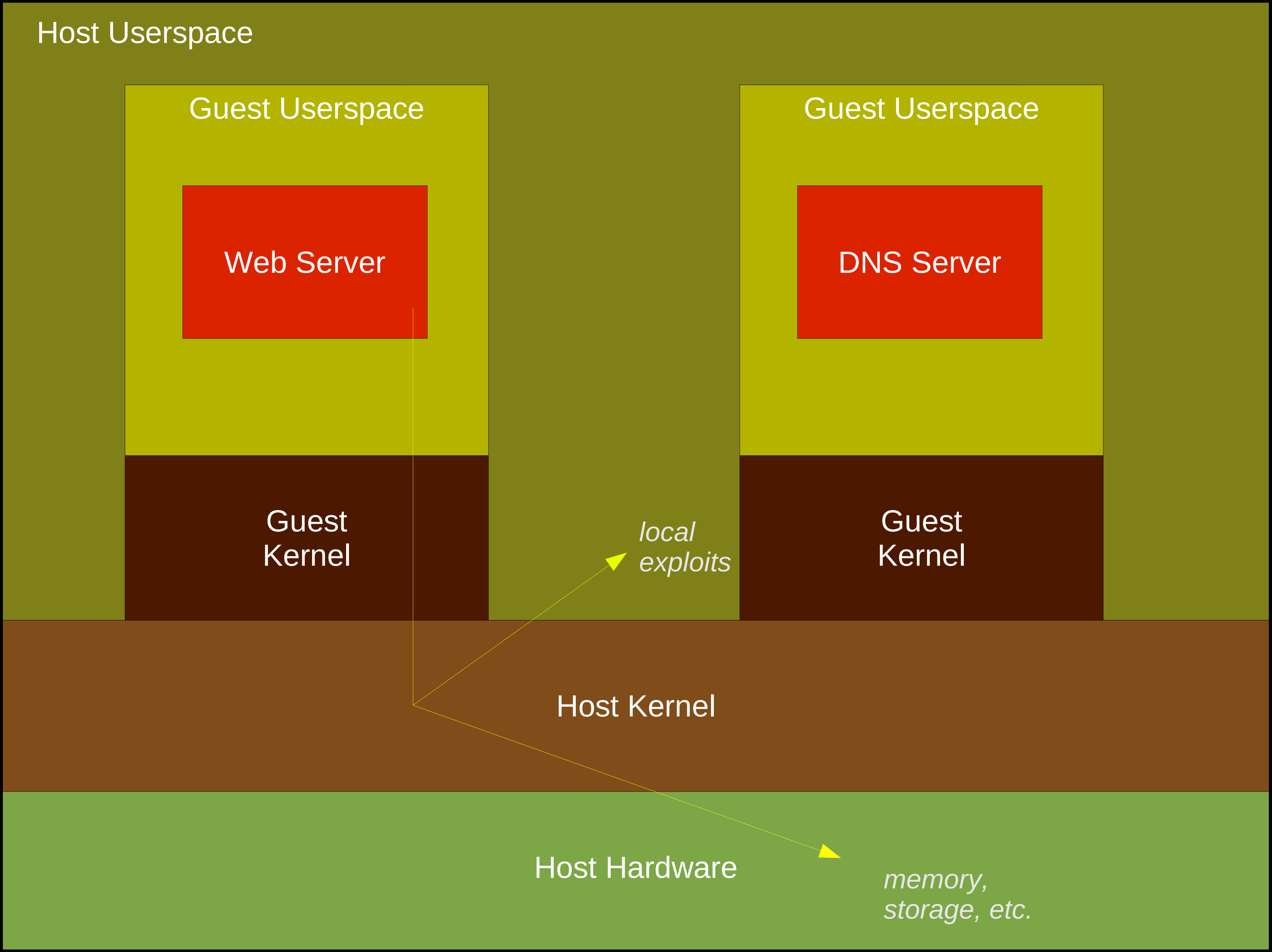
Before virtualization:

Systems were physically separated, damage limited to network attacks


Host Userspace
Web Server
Host Kernel
Host Hardware
Host Userspace
DNS Server
Host Kernel
Host Hardware
Attack
Local Network

After virtualization:

Guest systems running on same server, possibly as same UID


Host Userspace
Guest Userspace
Web Server
Guest Kernel
local exploits
Guest Userspace
DNS Server
Guest Kernel
Host Kernel
Host Hardware
memory, storage, etc.

Malicious or compromised guests can now attack other guests via local mechanisms

Hypervisor vulnerabilities:

Not theoretical

Evolving field

Potentially huge payoffs

sVirt in a nutshell:

Isolate guests using MAC security policy

Contain hypervisor breaches

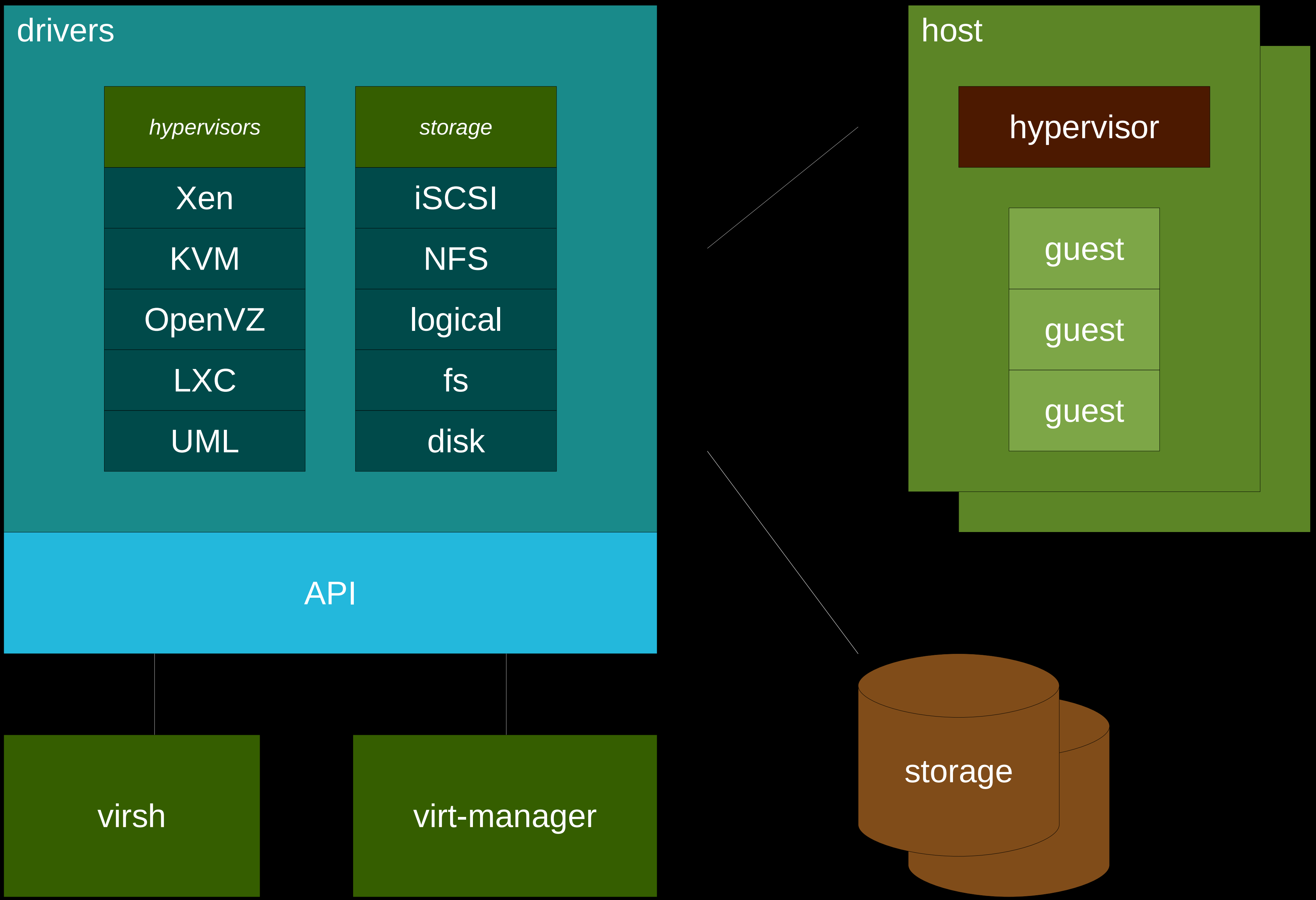
libvirt:

Virtualization API by Daniel Veillard

Abstraction layer for managing different virt schemes

Xen, KVM, LXC, OpenVZ

Simplified libvirt architecture



sVirt design:

Pluggable security framework for libvirt

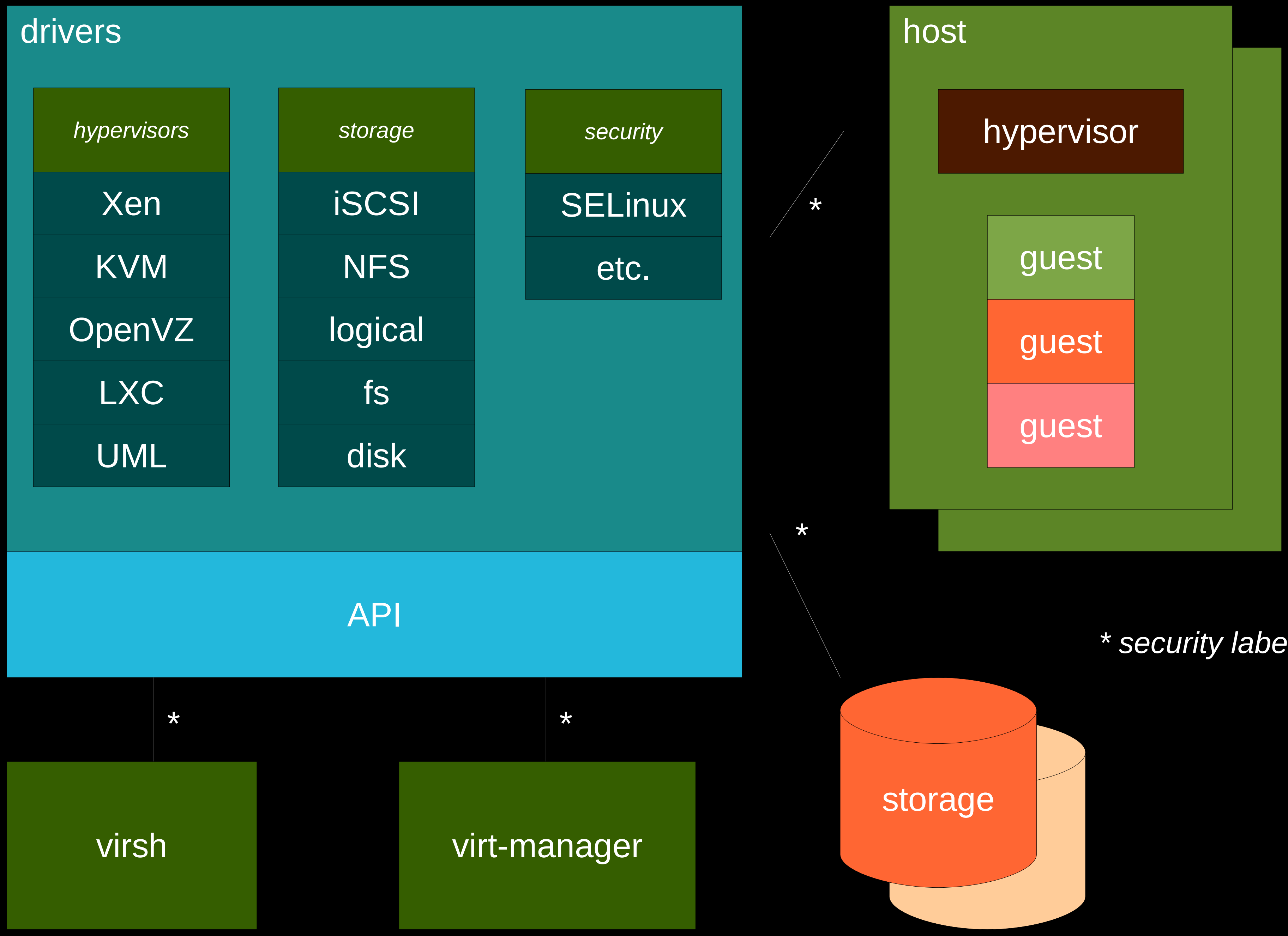
Supports MAC security schemes (SELinux, SMACK)

sVirt design:

Security “driver” manages MAC labeling of guests and resources

MAC policy enforced by host kernel

Simplified libvirt architecture w/ SVirt



sVirt design:

Reuse of proven code and security models

Coherent and complete system policy

Reduced complexity and cost

sVirt design:

Must be usable and useful with demonstrable value

sVirt v1.0:

Provide simple isolation of guests

Zero configuration

Debuggable

SELinux Policy:

Guests and resources uniquely labeled

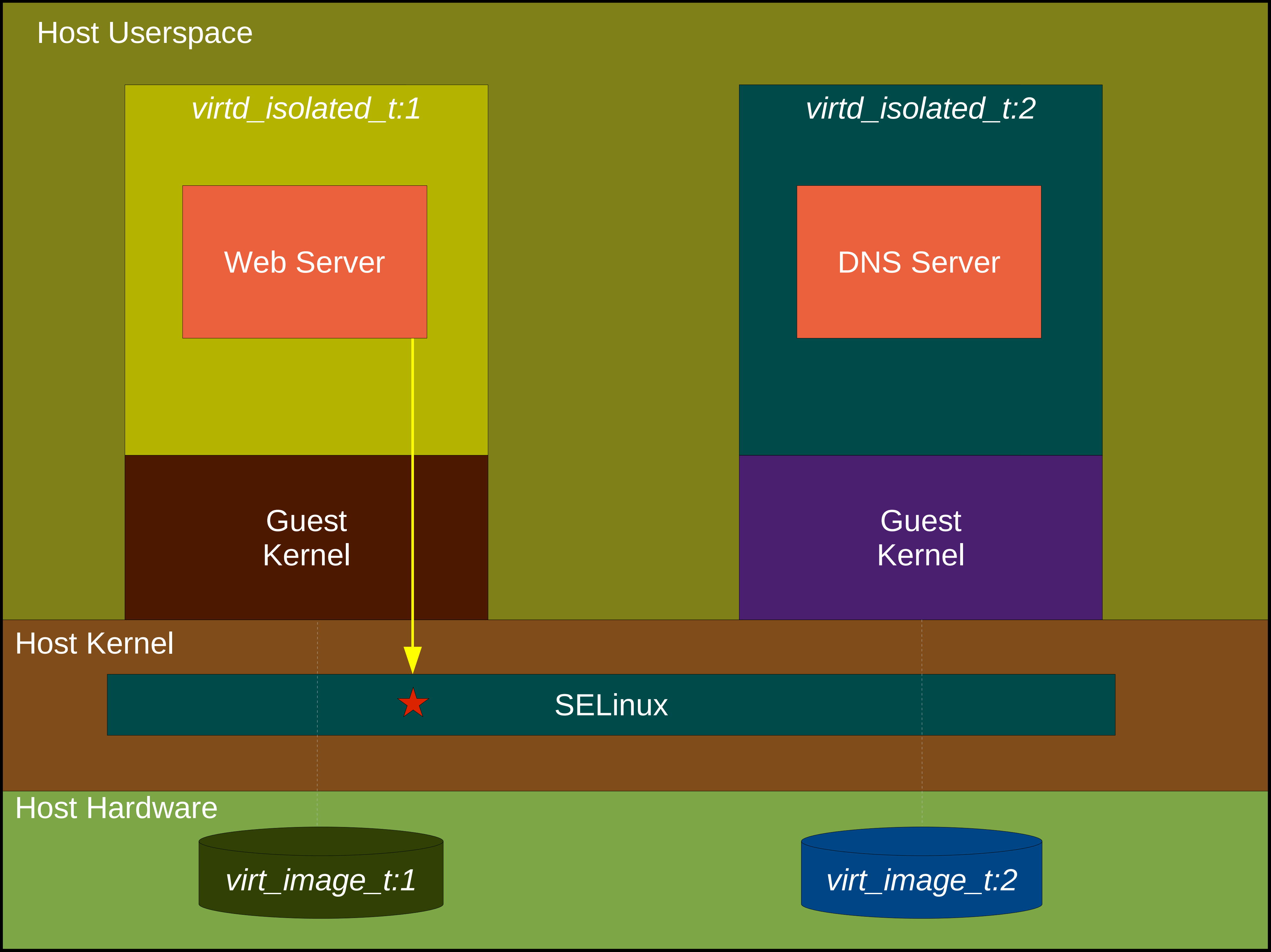
virtd_isolated_t:<UUID>

SELinux Policy:

Coarse rules for all isolated guests applied to *virtd_isolated_t*

SELinux Policy:

For simple isolation: all accesses between different UUIDs are denied


Host Userspace
virtd_isolated_t:1
Web Server
Guest
Kernel
virtd_isolated_t:2
DNS Server
Guest
Kernel
Host Kernel
SELinux
Host Hardware
virt_image_t:1
virt_image_t:2

Future enhancements:

Different types of isolated guests

virtd_isolated_webserver_t

Future enhancements:

Virtual network security

Controlled flow between guests

Distributed guest security

Multilevel security

Related work:

Labeled NFS

Labeled Networking

XACE

Similar work:

XSM (port of Flask to Xen)

Several proprietary schemes

Current status:

Low-level libvirt integration done

Can launch labeled guest

Basic label support in virsh

sVirt project page:

http://selinuxproject.org/page/SVirt

Questions...