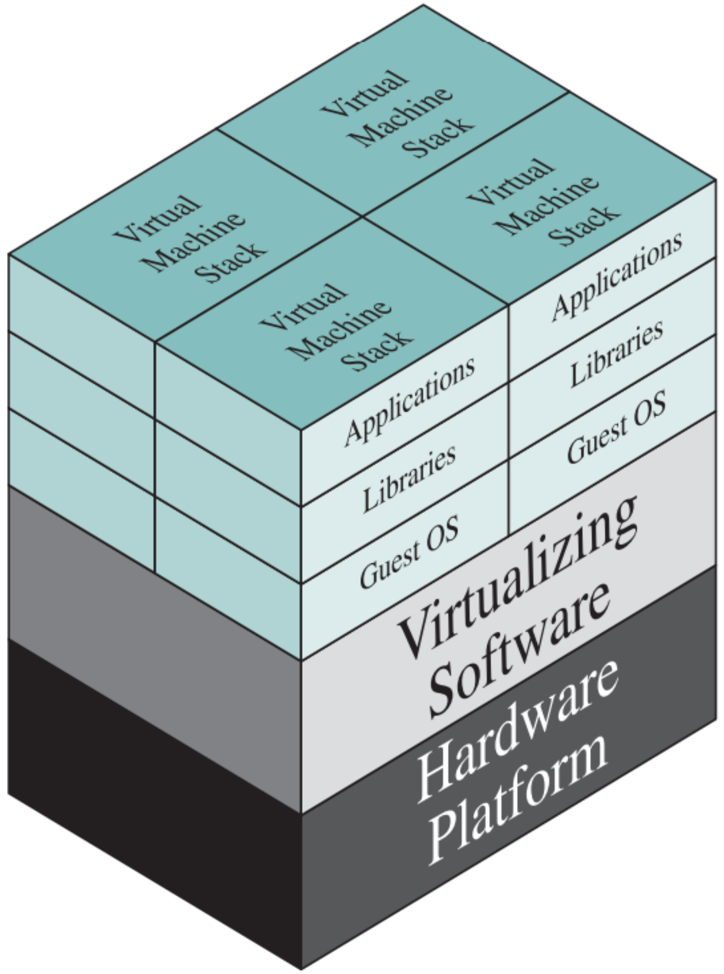
Virtualisation and Security

Virtual Machines (VM)

- Virtualization technology enables a single PC or server to simultaneously run multiple operating systems or multiple sessions of a single OS
- A machine with virtualization software can host numerous applications, including those that run on different operating systems, on a single platform
- The host operating system can support a number of virtual machines, each of which has the characteristics of a particular OS and, in some versions of virtualization, the characteristics of a particular hardware platform
- The solution that enables virtualization is a virtual machine monitor (VMM), or hypervisor
- This software sits between the hardware and the VMs acting as a resource broker

Virtual Machine Concept



Key Reasons for Using Virtualization

- We can summarize the key reasons the organizations use virtualization as follows:
- Legacy hardware
  - Applications built for legacy hardware can still be run by virtualizing the legacy hardware, enabling the retirement of the old hardware
- Rapid deployment
  - A new VM may be deployed in a matter of minutes
- Versatility
  - Hardware usage can be optimized by maximizing the number of kinds of applications that a single computer can handle
- Consolidation
  - A large-capacity or high-speed resource can be used more efficiently by sharing the resource among multiple applications simultaneously
- Aggregating
  - Virtualization makes it easy to combine multiple resources in to one virtual resource, such as in the case of storage virtualization
- Dynamics
  - Hardware resources can be easily allocated in a dynamic fashion, enhancing load balancing and fault tolerance
- Ease of management
  - Virtual machines facilitate deployment and testing of software
- Increased availability
  - Virtual machine hosts are clustered together to form pools of compute resources

Hypervisors

A Virtual Machine is a software construct that mimics the characteristics of a physical server

It is configured with some number of processors, some amount of RAM, storage resources, and connectivity through the network ports

Once the VM is created it can be powered on like a physical server, loaded with an operating system and software solutions, and utilized in the manner of a physical server

Unlike a physical server, this virtual server only sees the resources it has been configured with, not all of the resources of the physical host itself

The hypervisor facilitates the translation and I/O from the virtual machine to the physical server devices and back again to the correct virtual machine

Hypervisors

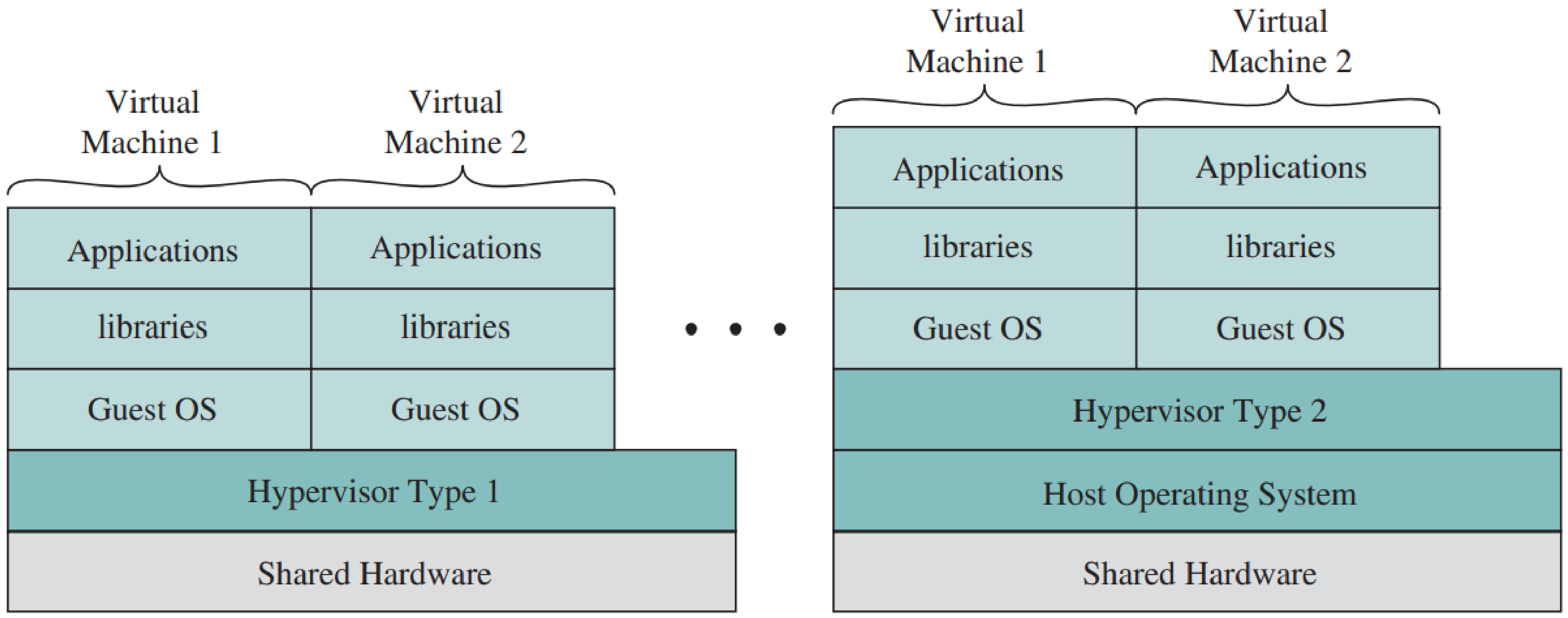
A VM instance is defined in files:

- Configuration file describes the attributes of the virtual machine
- It contains the server definition, how many virtual processors (vCPUs) are allocated to this virtual machine, how much RAM is allocated, which I/O devices the VM has access to, how many network interface cards (NICs) are in the virtual server, and more
- It also describes the storage that the VM can access
- When a virtual machine is powered on, or instantiated, additional files are created for logging, for memory paging, and other functions
- Since VMs are already files, copying them produces not only a backup of the data but also a copy of the entire server, including the operating system, applications, and the hardware configuration itself

Hypervisor Functions

- The principal functions performed by a hypervisor are:
    - Execution management of VMs
    - Devices emulation and access control
    - Execution of privileged operations by hypervisor for guest VMs
    - Management of VMs (also called VM lifecycle management
    - Administration of hypervisor platform and hypervisor software

Type 1 and Type 2 Hypervisors

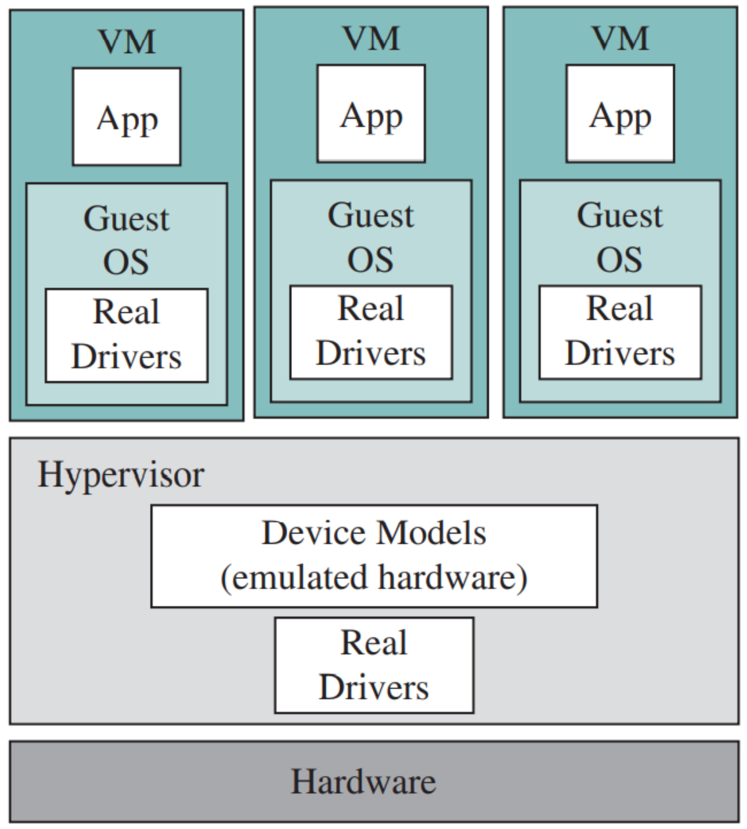


(a) Type 1 Hypervisor (b) Type 2 Hypervisor

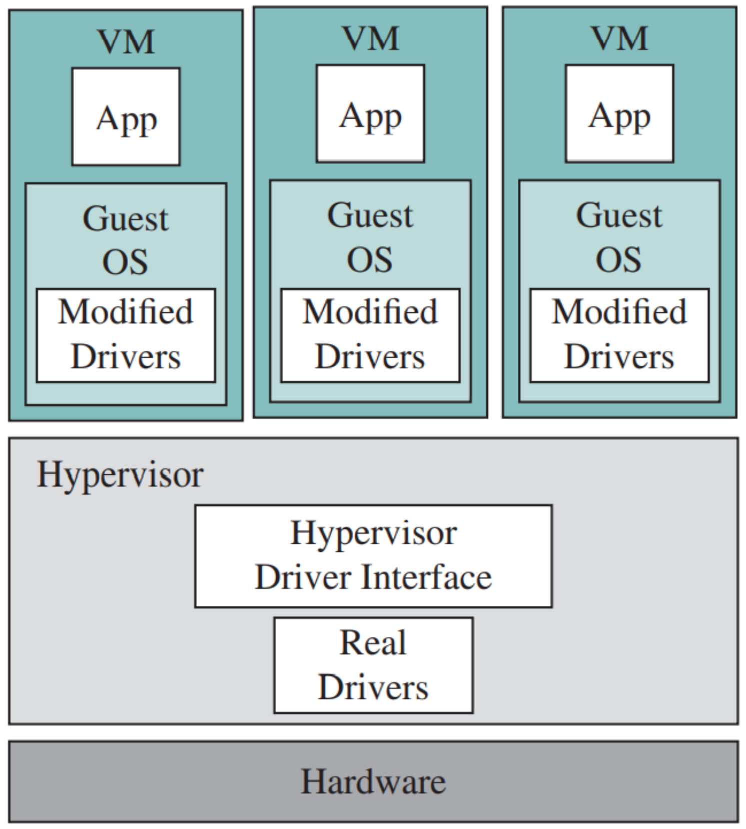
Paravirtualization

- A software assisted virtualization technique that uses specialized APIs to link virtual machines with the hypervisor to optimize their performance
- The operating system in the virtual machine, Linux or Microsoft Windows, has specialized paravirtualization support as part of the kernel, as well as specific paravirtualization drivers that allow the OS and hypervisor to work together more efficiently with the overhead of the hypervisor translations
- Support has been offered as part of many of the general Linux distributions since 2008

Paravirtualization



(a) Type 1 Hypervisor



(b) Paravirtualized Type 1 Hypervisor with Paravirtualized Guest OSs

Hardware-Assisted Virtualization

- Processor manufacturers AMD and Intel added functionality to their processors to enhance performance with hypervisors
- AMD-V and Intel's VT-x designate the hardware assisted virtualization extensions that the hypervisors can take advantage of during processing
- Intel processors offer an extra instruction set called Virtual Machine Extensions (VMX)
- By having some of these instructions as part of the processor, the hypervisors no longer need to maintain these functions as part of the processor

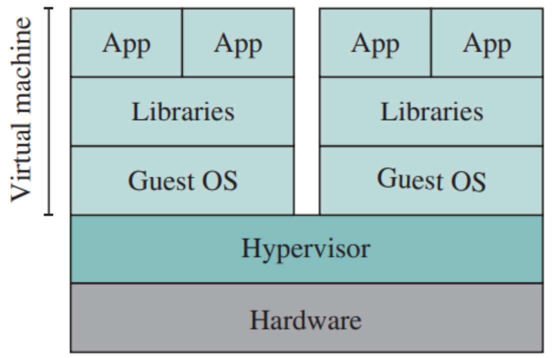
Virtual Appliance

- A virtual appliance is standalone software that can be distributed as a virtual machine image
- It consists of a packaged set of applications and guest OS
- It is independent of hypervisor or processor architecture, and can run on either a type 1 or type 2 hypervisor
- Virtual appliances are becoming a de-facto means of software distribution and have created a need for “the virtual appliance vendor”
- A recent and important development is the security virtual appliance (SVA)
  - The SVA is a security tool that performs the function of monitoring and protecting the other VMs and is run outside of those VMs in a specially security-hardened VM
  - The SVA obtains its visibility into the state of a VM as well as the network traffic between VMs, and between VMs and the hypervisor, through the virtual machine introspection API of the hypervisor
  - Advantages of SVA
    - Not vulnerable to a flaw in the Guest OS
    - Independent of the virtual network configuration and does not have to be reconfigured every time the virtual network configuration changes due to migration of VMs or change in connectivity among VMs resident on the hypervisor host

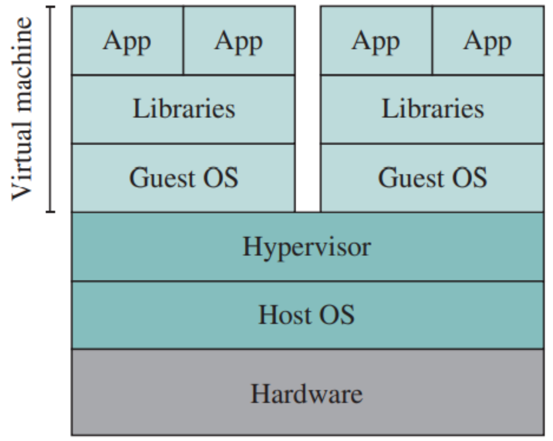
Container Virtualization

- Container virtualization is a relatively recent approach to virtualization
  - In this approach, software, known as a virtualization container, runs on top of the host OS kernel and provides an isolated execution environment for applications
  - Unlike hypervisor-based VMs, containers do not aim to emulate physical servers; instead, all containerized applications on a host share a common OS kernel
  - This eliminates the resources needed to run a separate OS for each application and can greatly reduce overhead

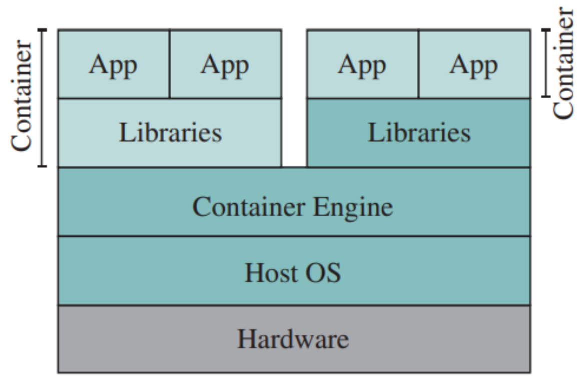
Comparison of Virtual Machines and Containers



(a) Type 1 Hypervisor

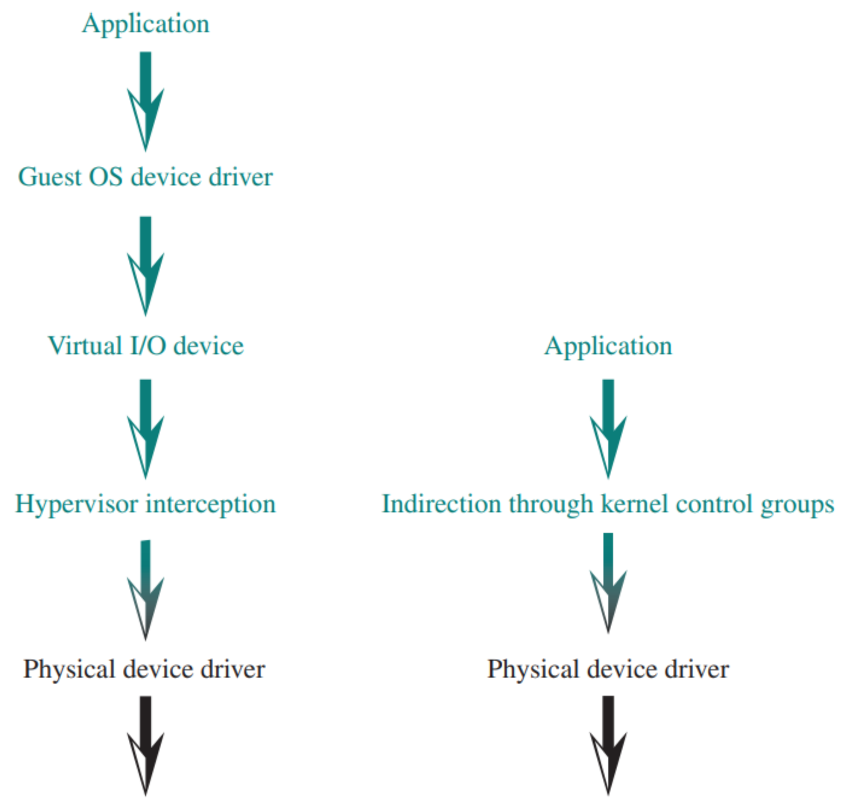


(b) Type 2 Hypervisor



(c) Container

Data Flow for I/O Operation via Hypervisor and Container



(a) Hypervisor (b) Container

Microservices

- NIST SP 800-180 (NIST Definition of Microservices, Application Containers and System Virtual Machines) defines a microservice as:

“a basic element that results from the architectural decomposition of an application’s components into loosely coupled patterns consisting of self-contained services that communicate with each other using a standard communication protocol and a set of well-defined APIs, independent of any vendor, product, or technology”

Microservices

Two key advantages of microservices are:

- Microservices implement much smaller deployable units, which then enables the user to push out updates or do features and capabilities much more quickly
  - This coincides with continuous delivery practices, where the goal is to push out small units without having to create a monolithic system
- Mocroservices also support precise scalability
  - Because a microservice is a section of a much larger application, it can easily be replicated to create multiple instances, and spread the load for just that one small piece of the application

Docker

- Provides a simpler and more standardized way to run containers
- Docker container also runs in Linux
- One of the reasons the Docker container is more popular compared to competing containers is its ability to load a container image on a host operating system in a simple and quick manner
- Docker containers are stored in the cloud as images and called upon for execution by users when needed in a simple way
- The principal components of Docker are:
    - Docker image
    - Docker client
    - Docker host
    - Docker engine
    - Docker machine
    - Docker registry
    - Docker hub

Processor Issues

- In a virtual environment there are two main strategies for providing processor resources:
  - Emulate a chip as software and provide access to that resource
    - Examples of this method are QEMU and the Android Emulator in the Android SDK
  - Provide segments of processing time on the physical processors (pCPUs) of the virtualization host to the virtual processors of the virtual machines hosted on the physical server
    - This is how most of the virtualization hypervisors offer processor resources to their guests

Processor Allocation

- When applications are migrated to virtual environments, the number of virtual processors allocated to their virtual machines needs to be determined

The number of processors a server has is one of the more important metrics when sizing a server

Moore's law provides processors that would be four times faster than those on the original physical server

There are tools available that will monitor resource (processor, memory, network, and storage I/O) usage on the physical server and then make recommendations for the optimum VM sizing

If the consolidation estimate utility cannot be run, there are a number of good practices in place

- One basic rule during VM creation is to begin with one vCPU and monitor the application's performance
- Another good practice is not to overallocate the number of vCPUs in a VM

Ring O

- Native operating systems manage hardware by acting as the intermediary between application code requests and the hardware
- One key function of the operating system is to help prevent malicious or accidental system calls from disrupting the applications or the operating system itself
- Protection rings describe level of access or privilege inside of a computer system and many operating systems and processor architectures take advantage of this security model
- The most trusted layer is often called Ring 0 (zero) and is where the operating system kernel works and can interact directly with hardware
- Hypervisors run in Ring 0 controlling hardware access for the virtual machines they host

Memory Management

- Since hypervisor manages page sharing, the virtual machine operating systems are unaware of what is happening in the physical system
- Ballooning
  - The hypervisor activates a balloon driver that (virtually) inflates and presses the guest operating system to flush pages to disk
  - Once the pages are cleared, the balloon driver deflates and the hypervisor can use the physical memory for other VMs
  - Memory overcommit
  - The capability to allocate more memory than physically exists on a host

I/O Management

- An advantage of virtualizing the workload’s I/O path enables hardware independence by abstracting vendor-specific drivers to more generalized versions that run on the hypervisor
- This abstraction enables:
  - Live migration, which is one of virtualization’s greatest availability strengths
  - The sharing of aggregate resources, such as network paths
  - The memory overcommit capability is another benefit of virtualizing the I/O of a VM
  - The trade-off for this is that the hypervisor is managing all the traffic and requires processor overhead
    - This was an issue in the early days of virtualization, but now faster multicore processors and sophisticated hypervisors have addressed this concern

Performance Technologies

I/OAT

I/O Acceleration Technology

Offered by Intel

A physical subsystem that moves memory copies via direct memory access (DMA) from the main processor to this specialized portion of the motherboard

TOE

TCP Offload Engine

Removes the TCP/IP processing from the server processor entirely to the NIC

LRO

Large Receive Offload

Aggregates incoming packets into bundles for more efficient processing

LSO

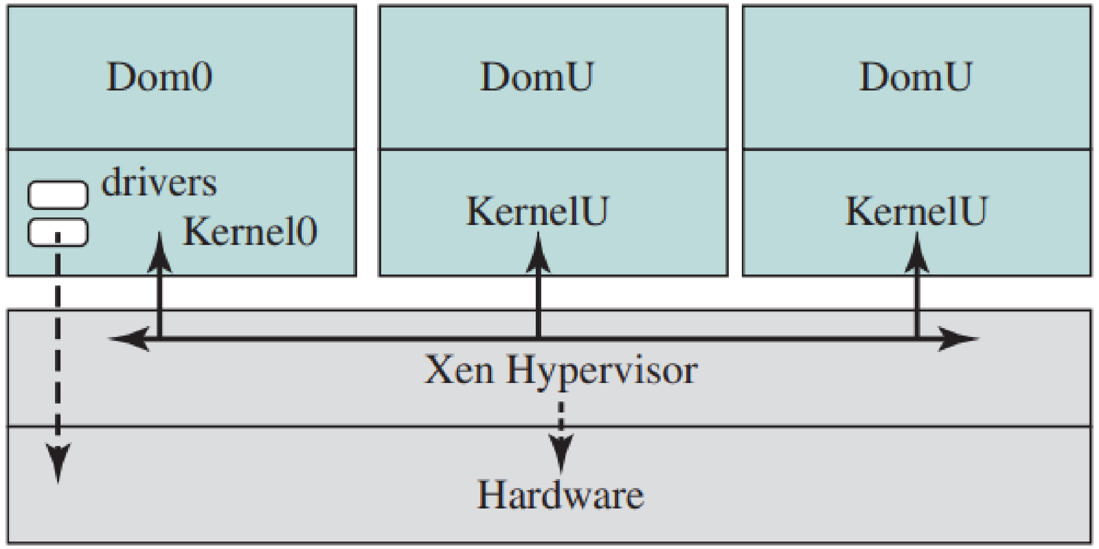
Large Segment Offload

Allows the hypervisor to aggregate multiple outgoing TCP/IP packets and has the NIC hardware segment them into separate packets

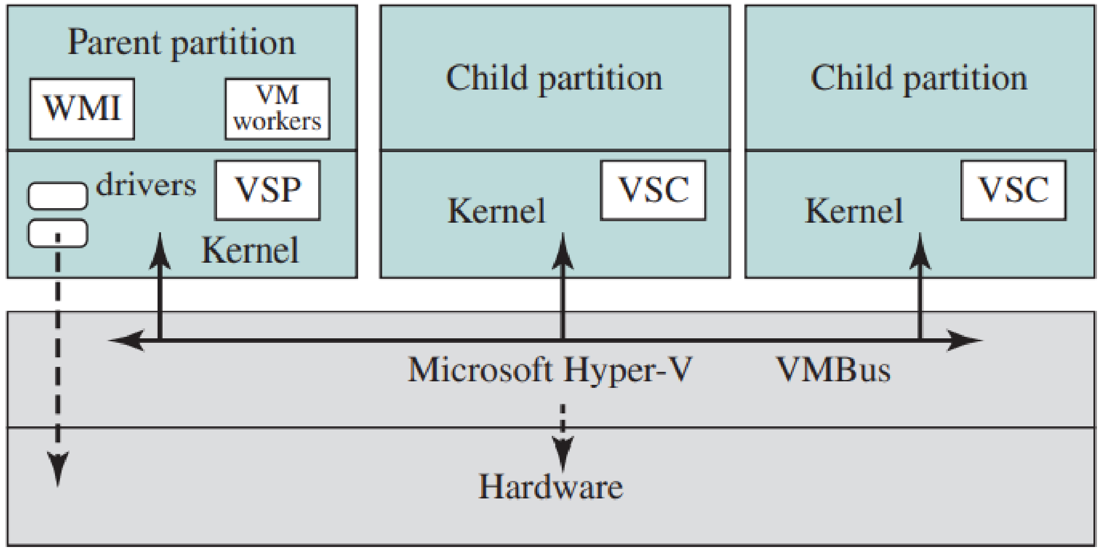
VMware ESXi

- A commercially available hypervisor from VMware that provides users a Type-1, or bare-metal, hypervisor to host virtual machines on their servers
- VMware developed their initial x86-based solutions in the late 1990s and were the first to deliver a commercial product to the marketplace
- This first-to-market timing, coupled with continuous innovations, has kept VMware firmly on top in market share

Xen & Hyper-V



Dom0
DomU
DomU
drivers
Kernel0
KernelU
KernelU
Xen Hypervisor
Hardware



Parent partition
WMI
VM workers
Child partition
Child partition
drivers
VSP
Kernel
Kernel
VSC
Kernel
VSC
Microsoft Hyper-V
VMBus
Hardware

Java VM

- The goal of a Java Virtual Machine (JVM) is to provide a runtime space for a set of Java code to run on any operating system staged on any hardware platform without needing to make code changes to accommodate the different operating systems or hardware
- The JVM can support multiple threads
- Promises “Write Once, Run Anywhere”

- The JVM is described as being an abstract computing machine consisting of:
    - An instruction set
    - A program counter register
    - A stack to hold variables and results
    - A heap for runtime data and garbage collection
    - A method area for code and constants

Linux VServer

- Linux VServer is an open-source, fast, lightweight approach to implementing virtual machines on a Linux server
- Only a single copy of the Linux kernel is involved
- VServer consists of a relatively modest modification to the kernel plus a small set of OS userland tools
- The VServer Linux kernel supports a number of separate virtual servers
- The kernel manages all system resources and tasks, including process scheduling, memory, disk space, and processor time

Architecture

- Each virtual server is isolated from the others using Linux kernel capabilities
- The isolation involves four elements:
    - chroot
        - A UNIX or Linux command to make the root directory (/) become something other than its default for the lifetime of the current process
        - This command provides file system isolation
    - chcontext
        - Linux utility that allocates a new security context and executes commands in that context
        - Each virtual server has its own execution context that provides process isolation
    - chbind
        - Executes a command and locks the resulting process and its children into using a specific IP address
        - System call provides network isolation
    - capablities
        - Refers to a partitioning of the privilege available to a root user
        - Each virtual server can be assigned a limited subset of the root user’s privileges which provides root isolation

System Access Threats

System access threats fall into two general categories:

Intruders

Malicious software

Intruders

Masquerader

An individual who is not authorized to use the computer and who penetrates a system's access controls to exploit a legitimate user's account

Misfeasor

A legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuses his or her privileges

Clandestine user

An individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection

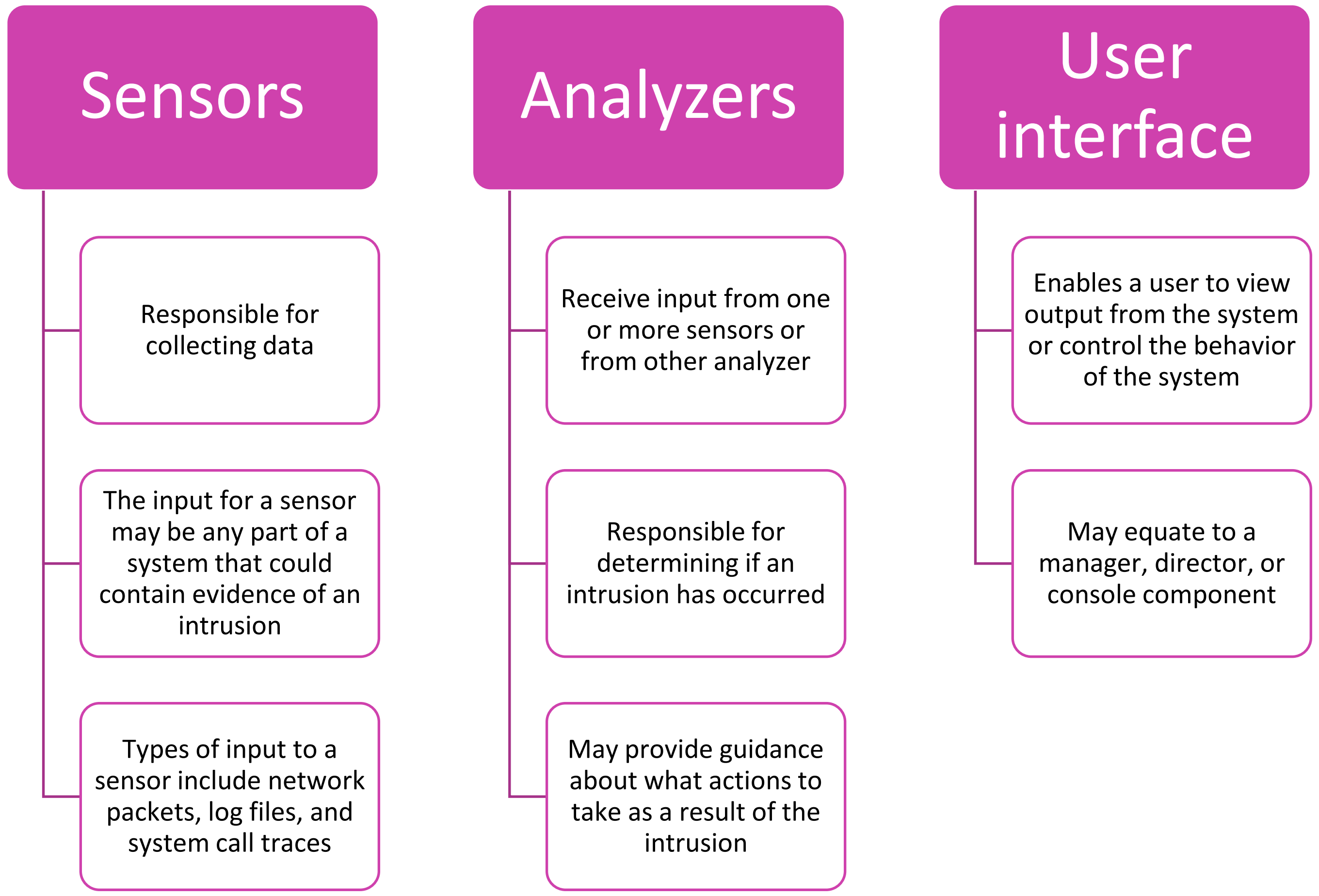
Malicious Software

- Programs that exploit vulnerabilities in computing systems
- Also referred to as malware
- Can be divided into two categories:
  - Parasitic
    - Fragments of programs that cannot exist independently of some actual application program, utility, or system program
    - Viruses, logic bombs, and backdoors are examples
  - Independent
    - Self-contained programs that can be scheduled and run by the operating system
    - Worms and bot programs are examples

Countermeasures

- RFC 4949 (Internet Security Glossary) defines intrusion detection as a security service that monitors and analyzes system events for the purpose of finding, and providing real-time or near real-time warning of, attempts to access system resources in an unauthorized manner
- Intrusion detection systems (IDSs) can be classified as:
  - Host-based IDS
    - Monitors the characteristics of a single host and the events occurring within that host for suspicious activity
  - Network-based IDS
    - Monitors network traffic for particular network segments or devices and analyzes network, transport, and application protocols to identify suspicious activity

IDS Components



Sensors

- Responsible for collecting data
- The input for a sensor may be any part of a system that could contain evidence of an intrusion
- Types of input to a sensor include network packets, log files, and system call traces

Analyzers

- Receive input from one or more sensors or from other analyzer
- Responsible for determining if an intrusion has occurred
- May provide guidance about what actions to take as a result of the intrusion

User interface

- Enables a user to view output from the system or control the behavior of the system
- May equate to a manager, director, or console component

Authentication

- In most computer security contexts, user authentication is the fundamental building block and the primary line of defense
- RFC 4949 defines user authentication as the process of verifying an identity claimed by or for a system entity
- An authentication process consists of two steps:
    - Identification step
        - Presenting an identifier to the security system
    - Verification step
        - Presenting or generating authentication information that corroborates the binding between the entity and the identifier

Means of Authentication

- Something the individual knows
  - Examples include a password, a personal identification number (PIN), or answers to a prearranged set of questions
- Something the individual possesses
  - Examples include electronic keycards, smart cards, and physical keys
  - Referred to as a token
- Something the individual is (static biometrics)
  - Examples include recognition by fingerprint, retina, and face
- Something the individual does (dynamic biometrics)
  - Examples include recognition by voice pattern, handwriting characteristics, and typing rhythm

Access Control

- Implements a security policy that specifies who or what may have access to each specific system resource and the type of access that is permitted in each instance
- Mediates between a user and system resources, such as applications, operating systems, firewalls, routers, files, and databases
- A security administrator maintains an authorization database that specifies what type of access to which resources is allowed for this user
  - The access control function consults this database to determine whether to grant access
- An auditing function monitors and keeps a record of user accesses to system resources

Firewalls

Design goals:

1. All traffic from inside to outside, and vice versa, must pass through the firewall. This is achieved by physically blocking all access to the local network except via the firewall
2. Only authorized traffic, as defined by the local security policy, will be allowed to pass. Various types of firewalls are used, which implement various types of security policies
3. The firewall itself is immune to penetration. This implies the use of a hardened system with a secured operating system.

Buffer Overflow Attacks

- Also known as a buffer overrun
- Defined in the NIST (National Institute of Standards and Technology) Glossary of Key Information Security Terms as:

“A condition at an interface under which more input can be placed into a buffer or data-holding area than the capacity allocated, overwriting other information. Attackers exploit such a condition to crash a system or to insert specially crafted code that allows them to gain control of the system”

- One of the most prevalent and dangerous types of security attacks

Exploiting Buffer Overflow

- To exploit any type of buffer overflow the attacker needs:
  - To identify a buffer overflow vulnerability in some program that can be triggered using externally sourced data under the attackers control
  - To understand how that buffer will be stored in the processes memory, and hence the potential for corrupting adjacent memory locations and potentially altering the flow of execution of the program

Defenses

- Countermeasures can be broadly classified into two categories:
  1. Compile-time defenses, which aim to harden programs to resist attacks
  2. Runtime defenses, which aim to detect and abort attacks in executing programs

Compile-time Techniques

- Choice of programming language
  - One possibility is to write the program using a modern high-level programming language that has a strong notion of variable type and what constitutes permissible operations on them
- Safe coding techniques
  - Programmers need to inspect the code and rewrite any unsafe coding constructs
- Language extensions and use of safe libraries
  - Libsafe is an example that implements the standard semantics but includes additional checks to ensure that the copy operations do not extend beyond the local variable space in the stack frame
- Stack protection mechanisms
  - Stackguard, one of the best-known protection mechanisms, is a GNU Compile Collection (GCC) compiler extension that inserts additional function entry and exit code

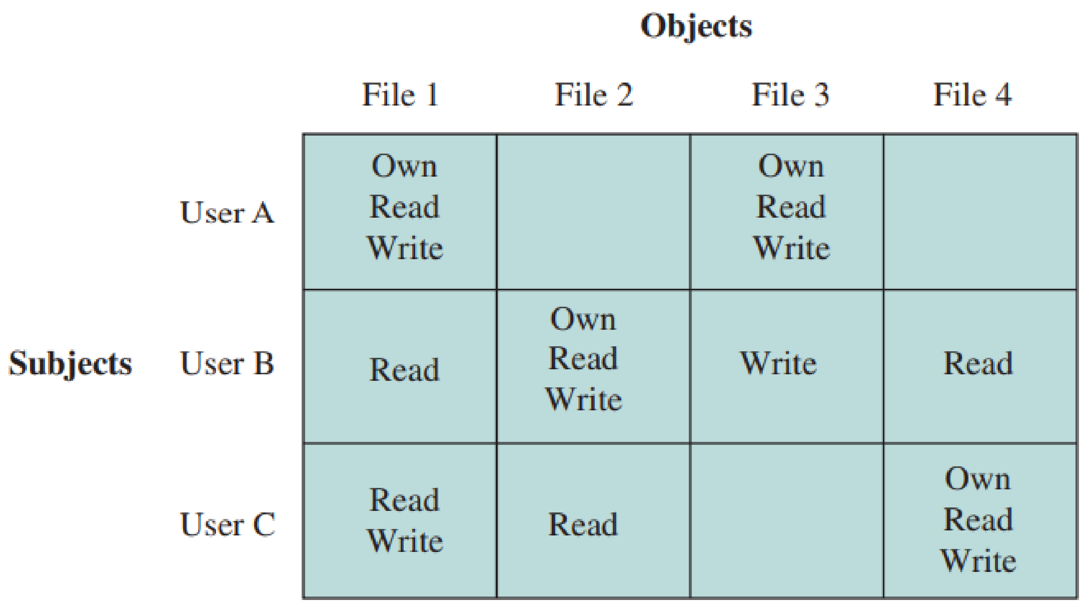
Runtime Techniques

- Executable address space protection
  - A possible defense is to block the execution of code on the stack, on the assumption that executable code should only be found elsewhere in the processes address space
- Address space randomization
  - Moving the stack memory region around by a megabyte or so has minimal impact on most programs but makes predicting the targeted buffer’s address almost impossible
  - Another technique is to use a security extension that randomizes the order of loading standard libraries by a program and their virtual memory address locations
- Guard pages
  - Caps are placed between the ranges of addresses used for each of the components of the address space
  - These gaps, or guard pages, are flagged in the MMU as illegal addresses and any attempt to access them results in the process being aborted
  - A further extension places guard pages between stack frames or between different allocations on the heap

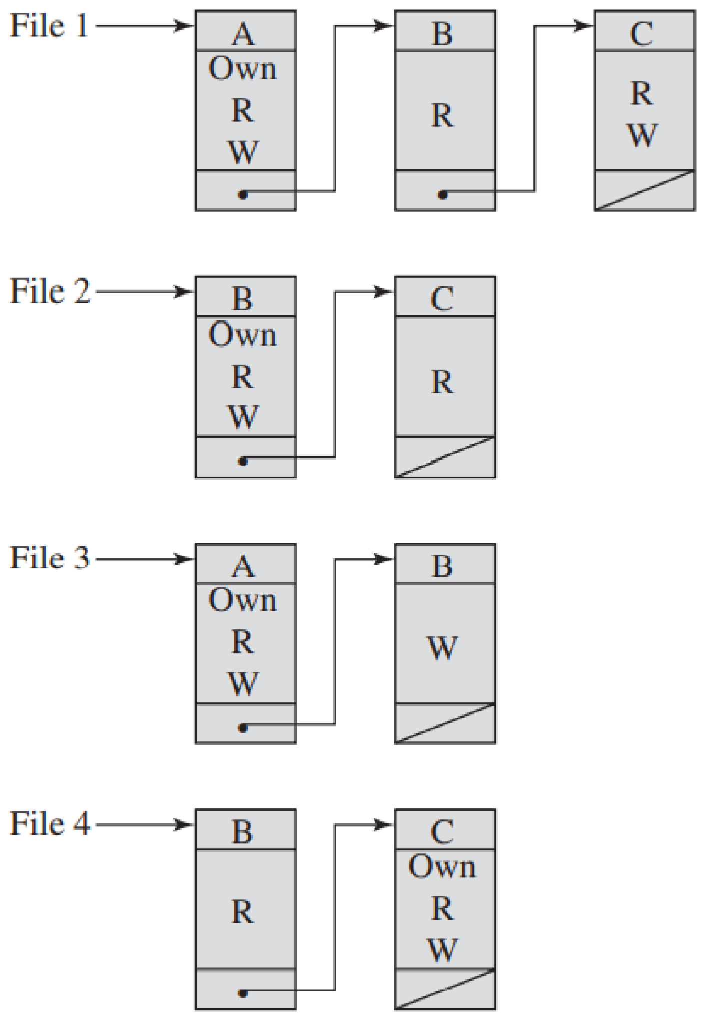
File System Access Control

- Identifies a user to the system
- Associated with each user there can be a profile that specifies permissible operations and file accesses
- The operating system can then enforce rules based on the user profile
- The database management system, however, must control access to specific records or even portions of records
  - The database management system decision for access depends not only on the user’s identity but also on the specific parts of the data being accessed and even on the information already divulged to the user

Access matrix



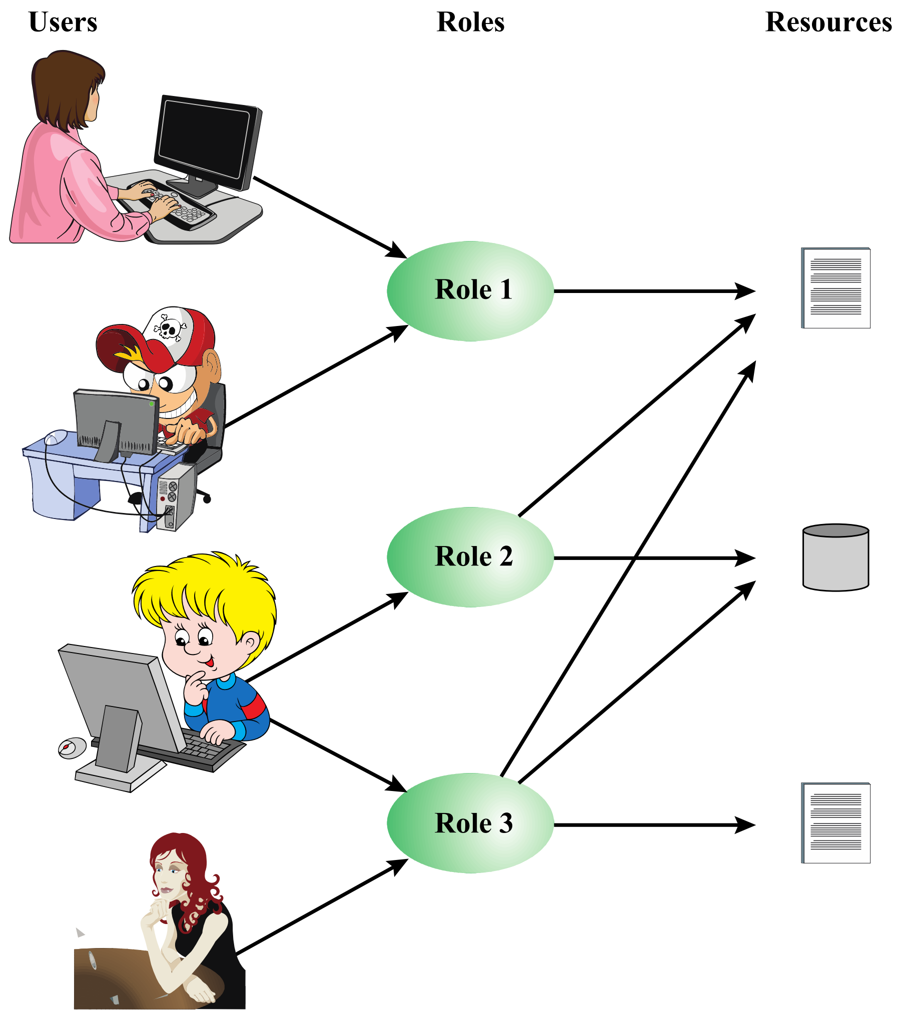
Subjects		File 1	File 2	File 3	File 4
	User A	Own Read Write		Own Read Write	
	User B	Read	Own Read Write	Write	Read
	User C	Read Write	Read		Own Read Write



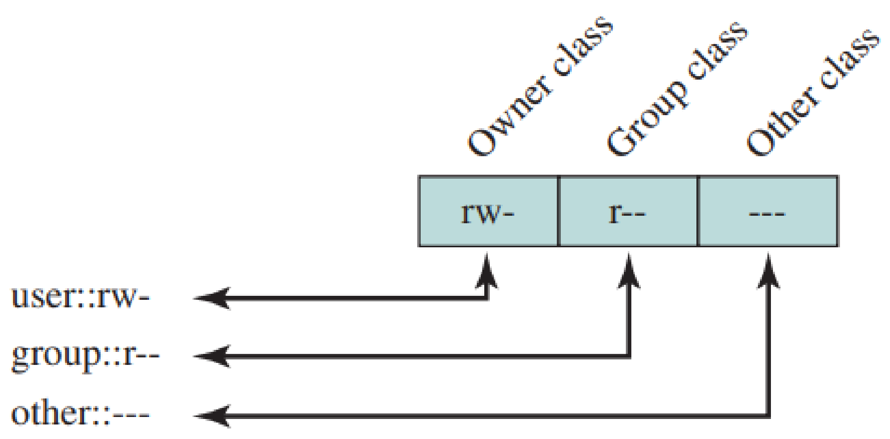
Access Control Policies

- An access control policy dictates what types of access are permitted, under what circumstances, and by whom
- Access control policies are generally grouped into the following categories:
  - Discretionary access control (DAC)
    - Controls access based on the identity of the requestor and on access rules stating what requestors are allowed to do
  - Mandatory access control (MAC)
    - Controls access based on comparing security labels with security clearances
  - Role-based access control (RBAC)
    - Controls access based on the roles that users have within the system, and on rules stating what accesses are allowed to users in given roles
  - Attribute-based access control (ABAC)
    - Controls access based on attributes of the user, the resource to be accessed, and current environmental conditions

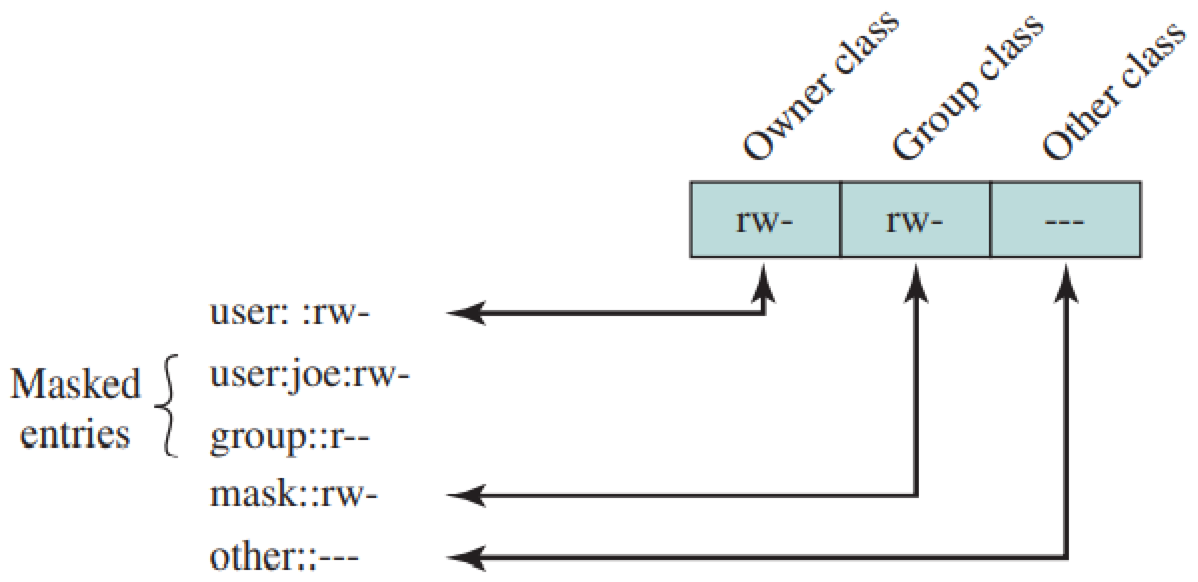
Users, Roles, and Resources



UNIX File Access Control



(a) Traditional UNIX approach (minimal access control list)



(b) Extended access control list

Operating Systems Hardening

- Basic steps to use to secure an operating system:
  - Install and patch the operating system
  - Harden and configure the operating system to adequately address the identified security needs of the system by:
    - Removing unnecessary services, applications, and protocols
    - Configuring users, groups and permissions
    - Configuring resource controls
  - Install and configure additional security controls, such as antivirus, host-based firewalls, and intrusion detection systems (IDS), if needed
  - Test the security of the basic operating system to ensure that the steps taken adequately address its security needs

Operating System Installation: Initial Setup and Patching

System security begins with the installation of the operating system

Ideally new systems should be constructed on a protected network

The initial installation should comprise the minimum necessary for the desired system, with additional software packages included only if they are required for the function of the system

The overall boot process must also be secured

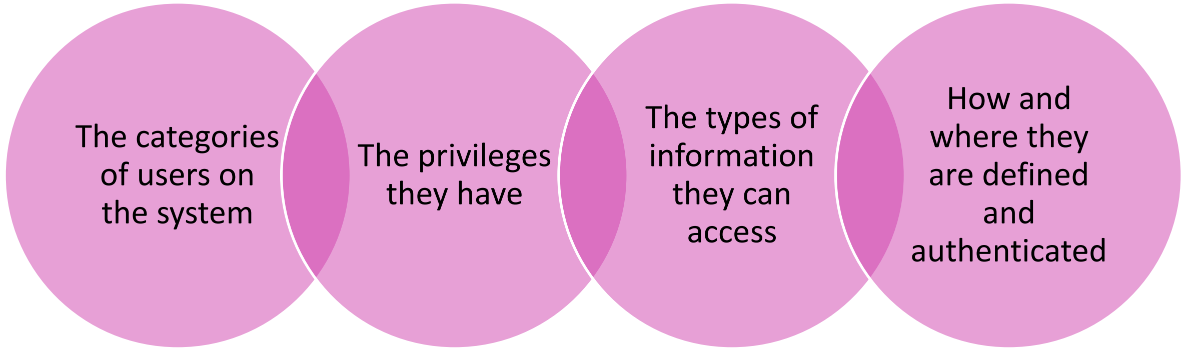
Care is also required with the selection and installation of any additional device driver code, since this executes with full kernel level privileges, but is often supplied by a third party

Remove Unnecessary Services, Applications, and Protocols

- The system planning process should identify what is actually required for a given system so that a suitable level of functionality is provided, while eliminating software that is not required to improve security
- When performing the initial installation, the supplied defaults should not be used, but rather the installation should be customized so that only the required packages are installed
- Many of the security-hardening guides provide lists of services, applications, and protocols that should not be installed if not required
- Strong preference is stated for not installing unwanted software, rather than installing and then later removing or disabling it as many uninstall scripts fail to completely remove all components of a package
  - Should an attacker succeed in gaining some access to a system, disabled software could be re-enabled and used to further compromise a system
  - It is better for security if unwanted software is not installed, and thus not available for use at all

Configure Users, Groups, and Authentication

- The system planning process should consider:


The categories of users on the system
The privileges they have
The types of information they can access
How and where they are defined and authenticated

- Restrict elevated privileges to only those users that require them
- At this stage any default accounts included as part of the system installation should be secured
- Those accounts which are not required should be either removed or at least disabled
- System accounts that manage services on the system should be set so they cannot be used for interactive logins
- Any passwords installed by default should be changed to new values with appropriate security
- Any policy that applies to authentication credentials and to password security is configured

Configure Resource Controls

- Once the users and their associated groups are defined, appropriate permissions can be set on data and resources to match the specified policy
- This may be to limit which users can execute some programs or to limit which users can read or write data in certain directory trees
- Many of the security-hardening guides provide lists of recommended changes to the default access configuration to improve security

Install Additional Security Controls

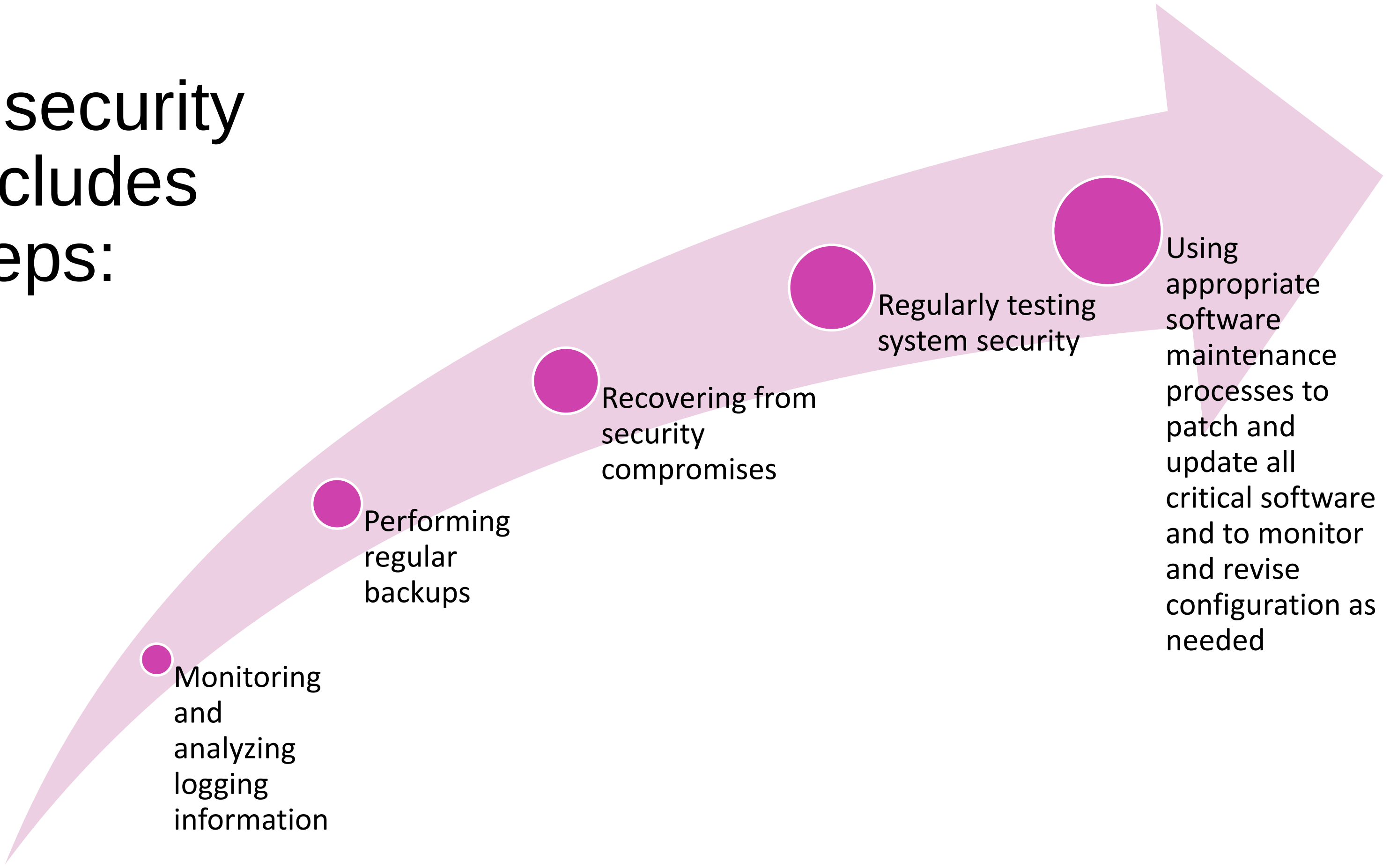
- Further security improvement may be possible by installing and configuring additional security tools such as antivirus software, host-based firewall, IDS or IPS software, or application white-listing
- Some of these may be supplied as part of the operating systems installation, but not configured and enabled by default
- Given the wide-spread prevalence of malware, appropriate antivirus is a critical security component
- IDS and IPS software may include additional mechanisms such as traffic monitoring or file integrity checking to identify and even respond to some types of attack
- White-listing applications limits the programs that can execute in the system to just those in an explicit list

Test the System Security

- The final step in the process of initially securing the base operating system is security testing
- The goal is to ensure that the previous security configuration steps are correctly implemented and to identify any possible vulnerabilities that must be corrected or managed
- Suitable checklists are included in many security-hardening guides
- There are also programs specifically designed to review a system to ensure that a system meets the basic security requirements and to scan for known vulnerabilities and poor configuration practices
- This should be done following the initial hardening of the system and then repeated periodically as part of the security maintenance process

Security Maintenance

- The process of security maintenance includes the following steps:


Monitoring and analyzing logging information
Performing regular backups
Recovering from security compromises
Regularly testing system security
Using appropriate software maintenance processes to patch and update all critical software and to monitor and revise configuration as needed

Logging

- Effective logging helps ensure that in the event of a system breach or failure, system administrators can more quickly and accurately identify what happened and more effectively focus their remediation and recovery efforts
- Logging information can be generated by the system, network, and applications
- The range of logging data acquired should be determined during the system planning stage
- Logging can generate significant volumes of information so it is important that sufficient space is allocated for them
- A suitable automatic log rotation and archive system should be configured to assist in managing the overall size of the logging information
- Some form of automated analysis is preferred as it is more likely to identify abnormal activity
  - Manual analysis of logs is tedious and is not a reliable means of detecting adverse events

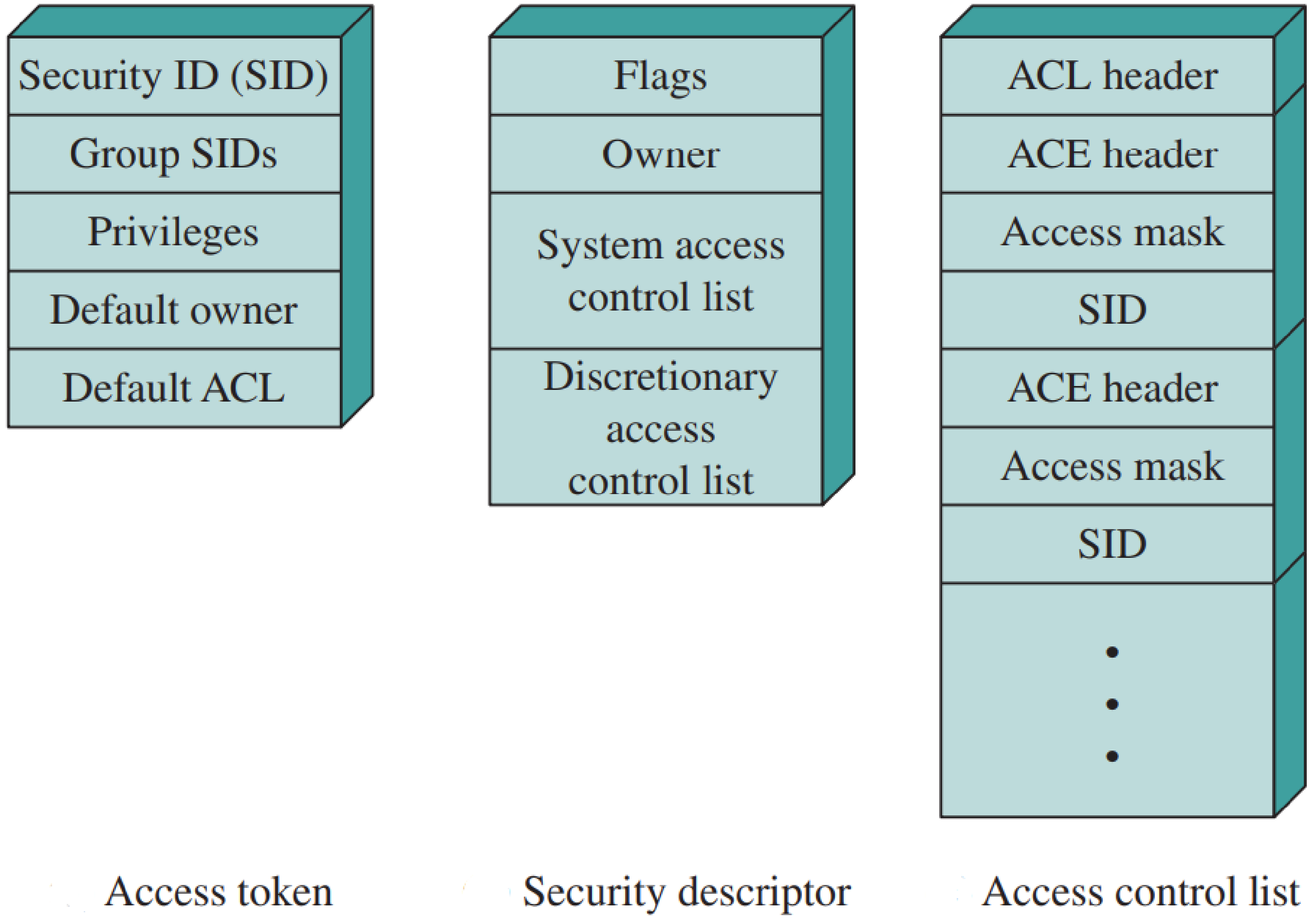
Data Backup and Archive

- Performing regular backups of data on a system is another critical control that assists with maintaining the integrity of the system and user data
- The needs and policy relating to backup and archive should be determined during the system planning stage
  - Key decisions include whether the copies should be kept online or offline and whether copies should be stored locally or transported to a remote site
- Backup
  - The process of making copies of data at regular intervals, allowing the recovery of lost or corrupted data over relatively short time periods of a few hours to some weeks
- Archive
  - The process of retaining copies of data over extended periods of time, being months or years, in order to meet legal and operational requirements to access past data

Access Control Scheme

- When a user logs on to a Windows system a name/password scheme is used to authenticate the user
- If the logon is accepted a process is created for the user and an access token is associated with that process object
  - The access token includes a security ID (SID) which is the identifier by which this user is known to the the system for purposes of security
  - The token also contains SIDs for the security groups to which the user belongs
- The access token serves two purposes:
  - It keeps all necessary security information together to speed access validation
  - It allows each process to modify its security characteristics in limited ways without affecting other processes running on behalf of the user

Windows Security Structures



Summary

- Virtual machine concepts
- Hypervisors
- Processor issues
- Memory management
- I/O management
- VMware ESXi
- Microsoft Hyper-V and Xen variants
- Java VM
- Linux VServer virtual machine architecture
- Intruders and malicious software
- Buffer overflow
- Access control
- UNIX access control
- Operating systems hardening
- Security maintenance
- Windows security

Thank you for your attention!