Assess and Mitigate Security Vulnerabilities

The more complex a system, the less assurance it provides. More complexity means more areas for vulnerabilities exist and more areas must be secured against threats. More vulnerabilities and more threats mean that the subsequent security provided by the system is less trustworthy.

**Hardware**

Any computing professional is familiar with the concept of hardware. As in the construction industry, hardware is the physical “stuff” that makes up a computer. The term hardware encompasses any tangible part of a computer that you can actually reach out and touch, from the keyboard and monitor to its CPU(s), storage media, and memory chips. Take careful note that although the physical portion of a storage device (such as a hard disk or fl ash memory) may be considered hardware, the contents of those devices—the collections of 0s and 1s that make up the software and data stored within them—may not. After all, you can’t reach inside the computer and pull out a handful of bits and bytes!

**Processor**

**The central processing unit (CPU), generally called the processor , is the computer’s nerve center**—it is the chip (or chips in a multiprocessor system) that governs all major operations and either directly performs or coordinates the complex symphony of calculations that allows a computer to perform its intended tasks. Surprisingly, **the CPU is capable of performing only a limited set of computational and logical operations**, despite the complexity of the tasks it allows the computer to perform. **It is the responsibility of the operating system and compilers to translate high-level programming languages used to design software into simple assembly language instructions that a CPU understands. This limited range of functionality is intentional**—it allows a CPU to perform computational and logical operations at blazing speeds.

**Execution Types**

As computer processing power increased, users demanded more advanced features to enable these systems to process information at greater rates and to manage multiple functions simultaneously. Computer engineers devised several methods to meet these demands:

**Multitasking** In computing, **multitasking means handling two or more tasks simultaneously**. In reality, most systems do not truly multitask; they rely on the operating system to simulate multitasking by carefully structuring the sequence of commands sent to the CPU for execution. After all, when your processor is humming along at multiple gigahertz, it’s hard to tell that it’s switching between tasks rather than working on two tasks at once. However, you can assume that a multitasking system is able to juggle more than one task or process at any given time.

**Multiprocessing In a multiprocessing environment, a multiprocessor computing system (that is, one with more than one CPU) harnesses the power of more than one processor to complete the execution of a single application**. For example, a **database server might run on a system that contains four, six, or more processors. If the database application receives a number of separate queries simultaneously, it might send each query to a separate processor for execution**

**Two types of multiprocessing** are most common in modern systems with multiple CPUs. The scenario just described, **where a single computer contains multiple processors that are treated equally and controlled by a single operating system, is called symmetric multiprocessing (SMP**) **. In SMP, processors share not only a common operating system but also a common data bus and memory resources. In this type of arrangement, systems may use a large number of processors. Fortunately, this type of computing** power is more than sufficient to drive most systems.

**Some computationally intensive operations, such as those that support the research of scientists and mathematicians, require more processing power than a single operating system can deliver. Such operations may be best served by a technology known as massively parallel processing (MPP**) . **MPP systems house hundreds or even thousands of processors, each of which has its own operating system and memory/bus resources**. When the software that coordinates the entire system’s activities and schedules them for processing encounters a computationally intensive task, **it assigns responsibility for the task to a single processor. This processor in turn breaks the task up into manageable parts and distributes them to other processors for execution. Those processors return their results to the coordinating processor where they are assembled and returned to the requesting application**. **MPP systems are extremely powerful** (not to mention extremely expensive!) and are used in **a great deal of computing or computational-based research.**

Both types of multiprocessing provide unique advantages and are suitable for different types of situations. **SMP systems are adept at processing simple operations at extremely high rates, whereas MPP systems are uniquely suited for processing very large, complex, computationally intensive** tasks that lend themselves to decomposition and distribution into a number of subordinate parts.

**Next-Generation Multiprocessing Until the release of dual-core and quad-core processors, the only way to create a multiprocessing system was to place two or more CPUs onto the motherboard. However, today we have several options of multicore CPUs so that with a single CPU chip on the motherboard, there are two or four (or more!) execution paths. This truly allows single CPU multiprocessing because it enables two (or more) calculations to occur simultaneously**.

**Multiprogramming** is similar to multitasking**. It involves the pseudo simultaneous execution of two tasks on a single processor coordinated by the operating system as a way to increase operational efficiency**. For the most part, multiprogramming is a way to batch or serialize multiple processes so that when one process stops to wait on a peripheral, its state is saved and the next process in line begins to process. The first program does not return to processing until all other processes in the batch have had their chance to execute and they in turn stop for a peripheral. For any single program, this methodology causes significant delays in completing a task. However, across all processes in the batch, the total time to complete all tasks is reduced.

**Multiprogramming is considered a relatively obsolete technology and is rarely found in use today except in legacy systems. There are two main differences between multiprogramming and multitasking:**

**Multiprogramming usually takes place on large-scale systems, such as mainframes, whereas multitasking takes place on PC operating systems, such as Windows and Linux.**

**■ Multitasking is normally coordinated by the operating system, whereas multiprogramming requires specially written software that coordinates its own activities and execution through the operating system**

**Multithreading permits multiple concurrent tasks to be performed within a single process. Unlike multitasking, where multiple tasks occupy multiple processes, multithreading permits multiple tasks to operate within a single process**. A thread is a self-contained sequence of instructions that can execute in parallel with other threads that are part of the same parent process. Multithreading is often used in applications where frequent context switching between multiple active processes consumes excessive overhead and reduces efficiency. In multithreading, switching between threads incurs far less overhead and is therefore more efficient. In modern Windows implementations, **for example, the overhead involved in switching from one thread to another within a single process is on the order of 40 to 50 instructions, with no substantial memory transfers needed. By contrast, switching from one process to another involves 1,000 instructions or more and requires substantial memory transfers as well.**

A good example of multithreading occurs when multiple documents are opened at the same time in a word processing program. In that situation, you do not actually run multiple instances of the word processor—this would place far too great a demand on the system. Instead, each document is treated as a single thread within a single word processor process, and the software chooses which thread it works on at any given moment. **Symmetric multiprocessing systems use threading at the operating system level**. As in the word processing example just described, the operating system also contains a number of threads that control the tasks assigned to it. In a single-processor system, the OS sends one thread at a time to the processor for execution. **SMP systems send one thread to each available processor for simultaneous execution**.

**Processing Types**

Many high-security systems control the processing of information assigned to various security levels, such as the classification levels of unclassified, sensitive, confidential, secret, and top secret that the US government assigns to information related to national defense. Computers must be designed so that they do not—ideally, so that they cannot—inadvertently disclose information to unauthorized recipients**. Computer architects and security policy administrators have addressed this problem at the processor level in two different ways. One is through a policy mechanism, whereas the other is through a hardware solution**. The following list explores each of those options

**Single State Single-state systems require the use of policy mechanisms to manage information at different levels. In this type of arrangement, security administrators approve a processor and system to handle only one security level at a time. For example, a system might be labeled to handle only secret information. All users of that system must then be approved to handle information at the secret level**. This shifts the burden of protecting the information being processed on a system away from the hardware and operating system and onto the administrators who control access to the system

**Multistate systems** are capable of **implementing a much higher level of security. These systems are certified to handle multiple security levels simultaneously by using specialized security mechanisms such as those described in the next section, “Protection Mechanisms**.” **These mechanisms are designed to prevent information from crossing between security levels. One user might be using a multistate system to process secret information, while another user is processing top-secret information at the same time. Technical mechanisms prevent information from crossing between the two users and thereby crossing between security levels.**

In actual practice, multistate systems are relatively uncommon owing to the expense of implementing the necessary technical mechanisms. This expense is sometimes justified; however, when you’re dealing with a very expensive resource, such as a massively parallel system, the cost of obtaining multiple systems far exceeds the cost of implementing the additional security controls necessary to enable multistate operation on a single such system.

**Protection Mechanisms**

If a computer isn’t running, it’s an inert lump of plastic, silicon, and metal doing nothing. When a computer is running, it operates a runtime environment that represents the combination of the operating system and whatever applications may be active. When running, the computer also has the capability to access fi les and other data as the user’s security permissions allow. **Within that runtime environment, it’s necessary to integrate security information and controls to protect the integrity of the operating system itself, to manage which users are allowed to access specific data items, to authorize or deny operations requested against such data, and so forth. The ways in which running computers implement and handle security at runtime may be broadly described as a collection of protection mechanisms. What follows are descriptions of various protection mechanisms such as protection rings, operational states, and security modes.**

**Protection Rings** The ring protection scheme is an oldie but a goodie. It dates all the way back to work on the Multics operating system. This experimental operating system was designed and built between 1963 and 1969 through the collaboration of Bell Labs, MIT, and General Electric. It saw commercial use in implementations from Honeywell. Multics has left two enduring legacies in the computing world. First, it inspired the creation of a simpler, less intricate operating system called Unix (a play on the word multics ), and second, it introduced the idea of protection rings to OS design.

From a security standpoint, protection rings organize code and components in an operating system (as well as applications, utilities, or other code that runs under the operating system’s control) into concentric rings, **The deeper inside the circle you go, the higher the privilege level associated with the code that occupies a specific ring**. **Though the original Multics implementation allowed up to seven rings (numbered 0 through 6), most modern operating systems use a four-ring model (numbered 0 through 3).**

As the innermost ring, 0 has the highest level of privilege and can basically access any resource, fi le, or memory location. The part of an operating system that always remains resident in memory (so that it can run on demand at any time) is called the kernel . It occupies ring 0 and can preempt code running at any other ring. The remaining parts of the operating system—**those that come and go as various tasks are requested, operations performed, processes switched, and so forth—occupy ring 1. Ring 2 is also somewhat privileged in that it’s where I/O drivers and system utilities reside; these are able to access peripheral devices, special fi les, and so forth that applications and other programs cannot themselves access directly. Those applications and programs occupy the outermost ring, ring 3.**

**Ring 0: OS Kernel/Memory (Resident Components)**

**Ring 1: Other OS Components**

**Ring 2: Drivers, Protocols, etc.**

**Ring 3: User-Level Programs and Applications**

**Rings 0–2 run in supervisory or privileged mode.**

**Ring 3 runs in user mode.**

**The essence of the ring model lies in priority, privilege, and memory segmentation**. Any process that wants to execute must get in line (a pending process queue). **The process associated with the lowest ring number always runs before processes associated with higher-numbered rings**. Processes in lower-numbered rings can access more resources and interact with the operating system more directly than those in higher-numbered rings. **Those processes that run in higher-numbered rings must generally ask a handler or a driver in a lower-numbered ring for services they need; this is sometimes called a mediated-access model** . In its strictest implementation, each ring has its own associated memory segment. Thus, any request from a process in a higher-numbered ring for an address in a lower numbered ring must call on a helper process in the ring associated with that address. **In practice, many modern operating systems break memory into only two segments: one for system-level access (rings 0 through 2), often called kernel mode or privileged mode , and one for user-level programs and applications (ring 3), often called user mode** .

From a security standpoint, the ring model enables an operating system to protect and insulate itself from users and applications. It also permits the enforcement of strict boundaries between highly privileged operating system components (such as the kernel) and less privileged parts of the operating system (such as other parts of the operating system, plus drivers and utilities). Within this model, direct access to specific resources is possible only within certain rings; likewise, certain operations (such as process switching, termination, and scheduling) are allowed only within certain rings.

The ring that a process occupies determines its access level to system resources (and determines what kinds of resources it must request from processes in lower-numbered, more privileged rings). Processes may access objects directly only if they reside within their own ring or within some ring outside its current boundaries (in numerical terms, for example, this means a process at ring 1 can access its own resources directly, plus any associated with rings 2 and 3, but it can’t access any resources associated only with ring 0). The mechanism whereby mediated access occurs—that is, the driver or handler request mentioned previously—is usually known as a system call and usually involves invocation of a specific system or programming interface designed to pass the request to an inner ring for service. Before any such request can be honored, however, the called ring must check to make sure that the calling process has the right credentials and authorization to access the data and to perform the operation(s) involved in satisfying the request.

**Process States Also known as operating states , process states are various forms of execution in which a process may run. Where the operating system is concerned, it can be in one of two modes at any given moment: operating in a privileged, all-access mode known as supervisor state or operating in what’s called the problem state associated with user mode, where privileges are low and all access requests must be checked against credentials for authorization before they are granted or deni**ed. The latter is called the problem state not because problems are guaranteed to occur but because the unprivileged nature of user access means that problems can occur and the system must take appropriate measures to protect security, integrity, and confidentiality.

Processes line up for execution in an operating system in a processing queue, where they will be scheduled to run as a processor becomes available. Because many operating systems allow processes to consume processor time only in fixed increments or chunks, when a new process is **created, it enters the processing queue for the first time; should a process consume its entire chunk of processing time (called a time slice** ) without completing, it returns to the processing queue for another time slice the next time its turn comes around. Also, the process scheduler usually selects the highest-priority process for execution, so reaching the front of the line doesn’t always guarantee access to the CPU (because a process may be preempted at the last instant by another process with higher priority).

**Ready** In the ready state, a process is ready to resume or begin processing as soon as it is scheduled for execution. If the CPU is available when the process reaches this state, it will transition directly into the running state; otherwise, it sits in the ready state until its turn comes up. This means the process has all the memory and other resources it needs to begin executing immediately.

**Waiting**  can also be understood as “waiting for a resource”—that is, the process is ready for continued execution but is waiting for a device or access request (an interrupt of some kind) to be serviced before it can continue processing (for example, a database application that asks to read records from a fi le must wait for that fi le to be located and opened and for the right set of records to be found). Some references label this state as a blocked state because the process could be said to be blocked from further execution until an external event occurs.

**Running** The running process executes on the CPU and keeps going until it finishes, its time slice expires, or it is blocked for some reason (usually because it has generated an interrupt for access to a device or the network and is waiting for that interrupt to be serviced). If the time slice ends and the process isn’t completed, it returns to the ready state (and queue); if the process blocks while waiting for a resource to become available, it goes into the waiting state (and queue).

The running state is also often called the problem state . However, don’t associate the word problem with an error. Instead, think of the problem state as you would think of a math problem being solved to obtain the answer. But keep in mind that it is called the problem state because it is possible for problems or errors to occur, just as you could do a math problem incorrectly. The problem state is separated from the supervisory state so that any errors that might occur do not easily affect the stability of the overall system; they affect only the process that experienced the error.

**Supervisory** The supervisory state is used when the process must perform an action that requires privileges that are greater than the problem state’s set of privileges, including modifying system configuration, installing device drivers, or modifying security settings. Basically, any function not occurring in the user mode (ring 3) or problem state takes place in the supervisory mode.

**Stopped** When a process finishes or must be terminated (because an error occurs, a required resource is not available, or a resource request can’t be met), it goes into a stopped state. At this point, the operating system can recover all memory and other resources allocated to the process and reuse them for other processes as needed.

how these various states relate to one another. New processes always transition into the ready state. From there, ready processes always transition into the running state. While running, a process can transition into the stopped state if it completes or is terminated, return to the ready state for another time slice, or transition to the waiting state until its pending resource request is met. When the operating system decides which process to run next, it checks the waiting queue and the ready queue and takes the highest-priority job that’s ready to run (so that only waiting jobs whose pending requests have been serviced, or are ready to service, are eligible in this consideration**). A special part of the kernel, called the program executive or the process scheduler, is always around (waiting in memory) so that when a process state transition must occur, it can step in and handle the mechanics involved**.

**Security Modes** The only new term in this context is need to know, which refers to an access authorization scheme in which a subject’s right to access an object takes into consideration not just a privilege level but also the relevance of the data involved in the role the subject plays (or the job they perform). This indicates that the subject requires access to the object to perform their job properly or to fill some specific role. Those with no need to know may not access the object, no matter what level of privilege they hold. Three specific elements must exist before the security modes themselves can be deployed:

**■ A hierarchical MAC environment**

**■ Total physical control over which subjects can access the computer console**

**■ Total physical control over which subjects can enter into the same room as the computer console**

**Dedicated Mode** Dedicated mode systems are essentially equivalent to the single-state system described in the section “Processing Types” earlier in this chapter. Three requirements exist for users of dedicated systems:

**■ Each user must have a security clearance that permits access to all information processed by the system.**

**■ Each user must have access approval for all information processed by the system.**

**■ Each user must have a valid need to know for all information processed by the system.**

**System High Mode** System high mode systems have slightly different requirements that must be met by users:

**■ Each user must have a valid security clearance that permits access to all information processed by the system.**

**■ Each user must have access approval for all information processed by the system.**

**■ Each user must have a valid need to know for some information processed by the system but not necessarily all information processed by the system**

**Compartmented mode** Compartmented mode systems weaken these requirements one step further:

■ **Each user must have a valid security clearance that permits access to all information processed by the system.**

**■ Each user must have access approval for any information they will have access to on the system.**

**■ Each user must have a valid need to know for all information they will have access to on the system.**

Notice that the major difference between compartmented mode systems and system high mode systems is that users of a compartmented mode system do not necessarily have access approval for all the information on the system. However, as with system high and dedicated systems, all users of the system must still have appropriate security clearances. In a special implementation of this mode called compartmented mode workstations (CMWs), users with the necessary clearances can process multiple compartments of data at the same time.

**CMWs require that two forms of security labels be placed on objects: sensitivity levels and information label**s. Sensitivity levels describe the levels at which objects must be protected. These are common among all four of the modes. Information labels prevent data over classification and associate additional information with the objects, which assists in proper and accurate data labeling not related to access control.

**Multilevel Mode** The government’s definition of multilevel mode systems pretty much parallels the technical definition given in the previous section. However, for consistency, we’ll express it in terms of clearance, access approval, and need to know:

■ **Some users do not have a valid security clearance for all information processed by the system. Thus, access is controlled by whether the subject’s clearance level dominates the object’s sensitivity label.**

**■ Each user must have access approval for all information they will have access to on the system.**

**■ Each user must have a valid need to know for all information they will have access to on the system.**

**Operating Modes**

**The processor itself supports two modes of operation: user mode and privileged mode**

**User Mode** User mode is the basic mode used by the CPU when executing user applications. In this mode, the CPU allows the execution of only a portion of its full instruction set. This is designed to protect users from accidentally damaging the system through the execution of poorly designed code or the unintentional misuse of that code. It also protects the system and its data from a malicious user who might try to execute instructions designed to circumvent the security measures put in place by the operating system or who might mistakenly perform actions that could result in unauthorized access or damage to the system or valuable information assets.

Often processes within user mode are executed within a controlled environment called a virtual machine (VM) or a virtual subsystem machine. A virtual machine is a simulated environment created by the OS to provide a safe and efficient place for programs to execute. Each VM is isolated from all other VMs, and each VM has its own assigned memory address space that can be used by the hosted application. It is the responsibility of the elements in privileged mode (aka kernel mode) to create and support the VMs and prevent the processes in one VM from interfering with the processes in other VMs.

**Privileged Mode** CPUs also support privileged mode, which is designed to give the operating system access to the full range of instructions supported by the CPU. This mode goes by a number of names, and the exact terminology varies according to the CPU manufacturer. Some of the more common monikers are included in the following list:

■ Privileged mode

■ Supervisory mode

■ System mode

■ Kernel mode

No matter which term you use, the basic concept remains the same—this mode grants a wide range of permissions to the process executing on the CPU. For this reason, well designed operating systems do not let any user applications execute in privileged mode. Only those processes that are components of the operating system itself are allowed to execute in this mode, for both security and system integrity purposes.

**Memory**

The second major hardware component of a system is *memory* , the storage bank for information that the computer needs to keep readily available. There are many different kinds of memory, each suitable for different purposes

**Read-Only Memory**

*Read-only memory* (ROM) works like the name implies—it’s memory the PC can read but can’t change (no writing allowed). The contents of a standard ROM chip are burned in at the factory, and the end user simply cannot alter it. **ROM chips often contain “bootstrap” information that computers use to start up prior to loading an operating system from disk. This includes the familiar power-on self-test (POST**) series of diagnostics that run each time you boot a PC.

**ROM’s primary advantage is that it can’t be modified. There is no chance that user or administrator error will accidentally wipe out or modify the contents of such a chip. This attribute makes ROM extremely desirable for orchestrating a computer’s innermost workings**

**There is a type of ROM that may be altered by administrators to some extent. It is known as programmable read-only memory (PROM), and its several subtypes**

**Programmable Read-Only Memory (PROM)** A basic programmable read-only memory (PROM) chip is similar to a ROM chip in functionality, but with one exception. During the manufacturing process, a PROM chip’s contents aren’t “burned in. **Once data is written to a PROM chip, no further changes are possible.**

PROM chips provide software developers with an opportunity to store information permanently on a high-speed, customized memory chip. PROMs are commonly used for hardware applications where some custom functionality is necessary but seldom changes once programmed.

**Erasable Programmable Read-Only Memory (EPROM)** Combine the relatively high cost of PROM chips and software developers’ inevitable desires to tinker with their code once it’s written and you have the rationale that led to the development of erasable PROM (EPROM). These chips have a small window that, when illuminated with a **special ultraviolet light, causes** the contents of the chip to be erased. After this process is complete, end users can burn new information into the EPROM as if it had never been programmed before.

**Electronically Erasable Programmable Read-Only Memory (EEPROM) Although it’s better than no erase function at all, EPROM erasure is pretty cumbersome. It requires the physical removal of the chip from the computer and exposure to a special kind of ultraviolet light. A more flexible, friendly alternative is electronically erasable PROM (EEPROM**), which uses electric voltages delivered to the pins of the chip to force erasure. EEPROM chips can be erased without removing them from the computer, which makes them much more attractive than standard PROM or EPROM chips

**Flash Memory** Flash memory is a derivative concept from EEPROM. It is a nonvolatile form of storage media that can be electronically erased and rewritten**. The primary difference between EEPROM and fl ash memory is that EEPROM must be fully erased to be rewritten whereas flash memory can be erased and written in blocks or pages. The most common type of flash memory is NAND flash. It is widely used in memory cards, thumb drives, mobile devices, and SSD (**solid-state drives).

**Random Access Memory**

Random access memory (RAM) is readable and writable memory that contains information a computer uses during processing. RAM retains its contents only when power is continuously supplied to it. Unlike with ROM, when a computer is powered off, all data stored in RAM disappears. For this reason, RAM is useful only for temporary storage. Critical data

should never be stored solely in RAM; a backup copy should always be kept on another storage device to prevent its disappearance in the event of a sudden loss of electrical power. The following are types of RAM

**Real Memory** Real memory (also known as *main memory* or *primary memory* ) is typically the largest RAM storage resource available to a computer. It is normally composed of a number of dynamic RAM chips and, therefore, must be refreshed by the CPU on a periodic basis

**Cache RAM** Computer systems contain a number of caches that improve performance by taking data from slower devices and temporarily storing it in faster devices when repeated use is likely; this is cache RAM. The processor normally contains an onboard cache of extremely fast memory used to hold data on which it will operate. This on-chip, or level 1, cache is often backed up by a static RAM cache on a separate chip, called a *level 2 cache* , which holds data from the computer’s main bank of real memory. Likewise, real memory often contains a cache of information stored on magnetic media or SSD. This chain continues down through the memory/ storage hierarchy to enable computers to improve performance by keeping data that’s likely to be used next closer at hand (be it for CPU instructions, data fetches, file access, or what have you).

**Dynamic RAM is cheaper than static RAM because capacitors are cheaper than flip-flops. However, static RAM runs much faster than dynamic RAM. This creates a trade-off for system designers, who combine static and dynamic RAM modules to strike the right balance of cost versus performance.**

**Registers**

The CPU also includes a limited amount of onboard memory, known as *registers* , that provide it with directly accessible memory locations that the brain of the CPU, the arithmetic logical unit (ALU), uses when performing calculations or processing instructions. In fact, any data that the ALU is to manipulate must be loaded into a register unless it is directly supplied as part of the instruction. The main advantage of this type of memory is that it is part of the ALU itself and, therefore, operates in lockstep with the CPU at typical CPU speeds.

**Memory Addressing**

When using memory resources, the processor must have some means of referring to various locations in memory. The solution to this problem is known as *addressing*, and there are several different addressing schemes used in various circumstances. The following are five of the more common addressing schemes

**Register Addressing As you learned in the previous section, registers are small memory locations directly in the CPU. When the CPU needs information from one of its registers to complete an operation, it uses a register address (for example, “register 1”) to access its contents**

**Immediate Addressing Immediate addressing is not a memory addressing scheme per se but rather a way of referring to data that is supplied to the CPU as part of an instruction.** For example, the CPU might process the command “Add 2 to the value in register 1.” This command uses two addressing schemes. The first is immediate addressing—the CPU is being told to add the value 2 and does not need to retrieve that value from a memory location—**it’s supplied as part of the command**. The second is register addressing; it’s instructed to retrieve the value from register 1.

**Direct Addressing** In direct addressing, **the CPU is provided with an actual address of the memory location to access. The address must be located on the same memory page as the instruction being executed**. Direct addressing **is more flexible than immediate** addressing since the contents of the memory location can be changed more **readily than** reprogramming the immediate addressing’s hard-coded data.

**Indirect Addressing** Indirect addressing uses a scheme similar to direct addressing. However, the memory address supplied to the CPU as part of the instruction doesn’t contain the actual value that the CPU is to use as an operand. Instead**, the memory address contains another memory address (perhaps located on a different page). The CPU reads the indirect address to learn the address where the desired data resides and then retrieves the actual operand from that address**.

**Base+Offset Addressing** Base+offset addressing uses a **value stored in one of the CPU’s registers as the base location from which to begin counting. The CPU then adds the offset supplied with the instruction to that base address and retrieves the operand from that computed memory location**.

**Base+Offset Addressing** Base+offset addressing uses a value stored in one of the CPU’s registers as the base location from which to begin counting. The CPU then adds the offset supplied with the instruction to that base address and retrieves the operand from that computed memory location. secondary memory is much more inexpensive

than primary memory and can be used to store massive amounts of information. In this context, hard disks, floppy drives, and optical media such as CDs and DVDs can all function as secondary memory.

Virtual memory is a special type of secondary memory that the operating system manages to make look and act just like real memory. The most common type of virtual memory is the page file that most operating systems manage as part of their memory management functions. This specially formatted fi le contains data previously stored in memory but not recently used. When the operating system needs to access addresses stored in the page file, it checks to **see whether the page is memory-resident (in which case it can access it immediately) or whether it has been swapped to disk, in which case it reads the data from disk back into real memory (this process is called *paging* ).**

Using virtual memory is **an inexpensive wa**y to make a computer operate as if it had more real memory than is physically installed. Its major drawback is that the paging operations that occur when data is exchanged between primary and secondary memory **are relatively slow (**memory functions in nanoseconds, disk systems in microseconds; usually this means three orders of magnitude difference!) and **consume significant computer overhead, slowing down the entire system.**

**Memory Security Issues**

Memory stores and processes your data—some of which may be extremely sensitive. It’s essential that you understand the various types of memory and know how they store and retain data. Any memory devices that may retain sensitive data should be purged before they are allowed to leave your organization for any reason. This is especially true for secondary memory and ROM/PROM/EPROM/EEPROM devices designed to retain data even after the power is turned off. However, memory data retention issues are not limited to those types of memory designed to retain data. Remember that static and dynamic RAM chips store data through the use of capacitors and flip-flops (see the sidebar “Dynamic vs. Static RAM”). It is technically possible that those electrical components could retain some of their charge for a limited period of time after power is turned off. A technically sophisticated individual could theoretically take electrical measurements of those components and retrieve portions of the data stored on such devices. However, this requires a good deal of technical expertise and is not a likely threat unless you have adversaries with mind-bogglingly deep pockets

**There is also an attack that freezes memory chips to delay the decay of resident data when the system is turned off or the RAM is pulled out of the motherboard “COLD\_BOOT\_ATTACK”**

**Storage**

**Primary vs. Secondary**

The concepts of primary and secondary storage can be somewhat confusing, especially when compared to primary and secondary memory. There’s an easy way to keep it straight—they’re the same thing! *Primary memory* , also known as *primary storage* , is the RAM that a computer uses to keep necessary information readily available to the CPU while the computer is running. *Secondary memory* (or *secondary storage*) includes all the familiar long-term storage devices that you use every day. Secondary storage consists of magnetic and optical media such as hard drives, solid-state drives (SSDs), floppy disks, magnetic tapes, compact discs (CDs), digital video disks (DVDs), flash memory cards, and the like.

**Volatile vs. Nonvolatile**

The volatility of a storage device is simply a measure of how likely it is to lose its data when power is turned off. Devices designed to retain their data (such as magnetic media) are classified as *nonvolatile* , whereas devices such as static or dynamic RAM modules, which are designed to lose their data, are classified as *volatile*

**Random vs. Sequential**

Storage devices may be accessed in one of two fashions. ***Random access storage* devices allow an operating system to read (and sometimes write) immediately from any point within the device by using some type of addressing system. Almost all primary storage devices are random access devices**. You can use a memory address to access information stored at any point within a RAM chip without reading the data that is physically stored before it. Most secondary storage devices are also random access. For example, hard drives use a movable head system that allows you to move directly to any point on the disk without spinning past all the data stored on previous tracks; likewise, CD and DVD devices use an optical scanner that can position itself anywhere on the platter surface.

*Sequential storage* devices, on the other hand, do not provide this flexibility. They require that you read (or speed past) all the data physically stored prior to the desired location. **A common example of a sequential storage device is a magnetic tape drive**. To provide access to data stored in the middle of a tape, the tape drive must physically scan through the entire tape (even if it’s not necessarily processing the data that it passes in fast-forward mode) until it reaches the desired point. Obviously, sequential storage devices operate much slower than random access storage devices. However, **here again you’re faced with a cost/benefit decision. Many sequential storage devices can hold massive amounts of data on relatively inexpensive media . This property makes tape drives uniquely suited for backup tasks associated with a disaster recovery/business continuity plan. In a backup situation, you often have extremely large** amounts of data that need to be stored, and you infrequently need to access that stored information. The situation just begs for a sequential storage device.

**Storage Media Security**

**Data may remain on secondary storage devices even after it has been erased. This condition is known as *data remanence.* you must use a specialized utility designed to destroy all traces of data on the device or damage or destroy it beyond possible repair (commonly called *sanitizing* ).**

**SSDs present a unique problem in relation to sanitization. SSD wear leveling means that there are often blocks of data that are not marked as “live” but that hold a copy of the data when it was copied off to lower wear leveled blocks. This means that a traditional zero wipe is ineffective as a data security measure for SSDs.**

it is important to use full disk encryption to reduce the risk of an unauthorized entity gaining access to your data. It is good security practice to encrypt SSDs prior to storing any data on them due to their wear leveling technology. This will minimize the chance of any plaintext data residing in dormant blocks. Fortunately, many HDD and SSD devices offer on-device native encryption.

**Input and Output Devices- Monitor/printer/keyboard/mic/Modem- TEMPEST**

**Input/Output Structures**

**Memory-Mapped I/O For many kinds of devices, memory-mapped I/O is a technique used to manage input/output. That is, a p**art of the address space that the CPU manages functions to provide access to some kind of device through a series of mapped memory addresses or locations. Thus, by reading mapped memory locations, you’re actually reading the input from the corresponding device (which is automatically copied to those memory locations at the system level when the device signals that input is available). Likewise, by writing to those mapped memory locations, you’re actually sending output to that device (automatically handled by copying from those memory locations to the device at the system level when the CPU signals that the output is available).

**From a configuration standpoint, it’s important to make sure that only one device maps into a specific memory address range and that the address range is used for no other purpose than to handle device I/O. From a security standpoint, access to mapped memory locations should be mediated by the operating system and subject to proper authorization and access controls.**

**Interrupt (IRQ)** Interrupt (IRQ) is an abbreviation **for *interrupt request***, a technique for assigning specific signal lines to specific devices through a special interrupt controller. When a device wants to supply input to the CPU, it sends a signal on its assigned IRQ (which usually falls in a **range of 0 to 16 on older PCs** **for two cascaded 8-line interrupt controllers and 0 to 23 on newer ones with three cascaded 8-line interrupt** controllers . From a configuration standpoint, finding unused IRQ numbers that will work with legacy devices can be a sometimes trying exercise. **From a security standpoint, only the operating system should be able to mediate access to IRQs at a sufficiently high level of privilege to prevent tampering or accidental misconfiguration.**

**Direct Memory Access (DMA)** Direct Memory Access (DMA) works as a channel with two signal lines, where one line is a DMA request (DMQ) line and the other is a DMA acknowledgment (DACK) line. Devices that can exchange data directly with real memory (RAM) without requiring assistance from the CPU use DMA to manage such access. Using its DRQ line, a device signals the CPU that it wants to make direct access (which may be read or write or some combination of the two) to another device, usually real memory. The CPU authorizes access and then allows the access to proceed independently while blocking other access to the memory locations involved. When the access is complete, the device uses the DACK line to signal that the CPU may once again permit access to previously blocked memory locations. This is faster than requiring the CPU to mediate such access and permits the CPU to move on to other tasks while the memory access is underway. DMA is used most commonly to permit disk drives, optical drives, display cards, and multimedia

cards to manage large-scale data transfers to and from real memory**. From a configuration standpoint, it’s important to manage DMA addresses to keep device addresses unique and to make sure such addresses are used only for DMA signaling. From a security standpoint, only the operating system should be able to mediate DMA assignment and the use of DMA to access I/O devices.**

**Firmware**

*Firmware* (also known as *microcode* in some circles) is a term used to describe software that is stored in a ROM chip. This type of software is changed infrequently (actually, never, if it’s stored on a true ROM chip as opposed to an EPROM/EEPROM) and often drives the basic operation of a computing device. **There are two types of firmware: BIOS on a motherboard and general internal and external device firmware**

**BIOS**

The Basic Input/Output System (BIOS) contains the operating system–independent primitive instructions that a computer needs to start up and load the operating system from disk. The BIOS is contained in a firmware device that is accessed immediately by the computer at boot time. **In most computers, the BIOS is stored on an EEPROM chip to facilitate version updates.** **The process of updating the BIOS is known as “flashing the BIOS**.” There have been a few examples of **malicious code embedding itself into BIOS/firmware. There is also an attack known as *phlashing,* in which a malicious variation of official BIOS or firmware is installed that introduces remote control or other malicious features into a device.**

Since 2011, most system manufacturers have replaced the traditional BIOS system on their motherboards with UEFI **(unified extensible firmware interface).** UEFI is a more advanced interface between hardware and the operating system, which maintains support for legacy BIOS services.

**Applets**

applets are actually self-contained miniature programs that execute independently of the server that sent them applets introduce a number of security concerns. They allow a remote system to send code to the local system for execution. Security administrators must take steps to ensure that code sent to systems on their network is safe and properly screened for malicious activity. Also, unless the code is analyzed line by line, the end user can never be certain that the applet doesn’t contain a Trojan horse component. For example, the mortgage calculator might indeed transmit sensitive financial information to the web server without the end user’s knowledge or consent

**Two common applet types are Java applets and ActiveX controls:**

**Jave applets** Java overcomes this limitation by inserting the Java Virtual Machine (JVM) into the picture. Each system that runs Java code downloads the version of the JVM supported by its operating system. The JVM then takes the Java code and translates it into a format executable by that specific system. The great benefit of this arrangement is that code can be shared between operating systems without modification. Java applets are simply short Java programs transmitted over the Internet to perform operations on a remote system.

The sandbox isolates Java code objects from the rest of the operating system and enforces strict rules about the resources those objects can access. For example, the sandbox would prohibit a Java applet from retrieving information from areas of memory not specifically allocated to it, preventing the applet from stealing that information. Unfortunately, while sandboxing reduces the forms of malicious events that can be launched via Java, there are still plenty of other vulnerabilities that have been widely exploited.

**Activex** First, ActiveX controls use proprietary Microsoft technology and, therefore, can execute only on systems running Microsoft browsers. Second, ActiveX controls are not subject to the sandbox restrictions placed on Java applets. They have full access to the Windows operating environment and can perform a number of privileged actions. Therefore, you must take special precautions when deciding which ActiveX controls to download and execute. Some security administrators have taken the somewhat harsh position of prohibiting the download of any ActiveX content from all but a select handful of trusted sites

**ARP cache poisoning is caused by an attack responding to ARP broadcast queries in order to send back falsified replies. If the false reply is received by the client before the valid reply, then the false reply is used to populate the ARP cache and the valid reply is discarded as being outside of an open query. The dynamic content of ARP cache, whether poisoned or legitimate, will remain in cache until a timeout occurs**

**A second form of ARP cache poisoning is to create static ARP entries. This is done via the ARP command and must be done locally. But this is easily accomplished through a script that gets executed on the client either through a Trojan horse, buffer overflow, or social engineering attack.**

**ARP cache poisoning or just ARP poisoning is one means of setting up a man-in-the-middle attack**

**There are many means of performing DNS cache poisoning, including HOSTS poisoning, authorized DNS server attacks, caching DNS server attacks, DNS lookup address changing, and DNS query spoofing**

**HOSTS poisoning -** The HOSTS fi le is the static fi le found on TCP/IP supporting system that contains hardcoded references for domain names and their associated IP addresses

**Authorized DNS server attacks** aim at altering the **primary record of a FQDN** on its original host system. The authoritative DNS server hosts the zone fi le or domain database. If this original dataset is altered, then eventually those changes will propagate across the entire Internet. However, an attack on an authoritative DNS server typically gets noticed very quickly,

**caching DNS server attack** is any **DNS system deployed to cache DNS** information from other DNS servers. Most companies and ISPs provide a caching DNS server for their users. The content hosted on a caching DNS server is not being watched by the worldwide security community

**DNS poisoning** focuses on sending an alternate IP address to the client to be used as the DNS server the client uses for resolving queries.

A fifth example of **DNS poisoning is that of DNS query spoofing. This attack occurs when the hacker is able to eavesdrop on a client’s query to a DNS server.**

Database Security

**Aggregation -** Aggregation attacks are used to collect numerous **low level security items or low-value items and combine them to create something of a higher security level or value.**

**Inference attacks** involve **combining several pieces of nonsensitive information to gain access to information that should be classified at a higher level**

**Many organizations use large databases, known as *data warehouse.***

**A *data dictionary* is commonly used for storing critical information about data, including usage, type, sources, relationships, and formats. DBMS software reads the data dictionary to determine access rights for users attempting to access data**.

**Data mining techniques allow analysts to comb through data warehouses and look for potential correlated information.**

**The activity of data mining produces metadata. Metadata is data about data or information about data. It can also be a superset, a subset, or a representation of a larger dataset. Metadata can be the important, significant, relevant, abnormal, or aberrant elements from a dataset. metadata is stored in a more secure container known as the data mart.**

**Cloud computing**

**Platform-as-a-Service** Platform-as-a-Service (PaaS) is the concept of providing a computing platform and software solution stack as a virtual or cloud-based service. Essentially, this type of cloud solution provides all the aspects of a platform (that is, the operating system and complete solution package). The primary attraction of PaaS is the avoidance of having to purchase and maintain high-end hardware and software locally.

**Software-as-a-Service** Software-as-a-Service (SaaS) is a derivative of PaaS. SaaS provides on-demand online access to specific software applications or suites without the need for local installation. In many cases, there are few local hardware and OS limitations. SaaS can be implemented as a subscription service (for example, Microsoft Offi ce 365), a pay-as-yougo service, or a free service (for example, Google Docs).

**Infrastructure-as-a-**Service Infrastructure-as-a-Service (IaaS) takes the PaaS model yet another step forward and provides not just on-demand operating solutions but complete outsourcing options. This can include utility or metered computing services, administrative task automation, dynamic scaling, virtualization services, policy implementation and management services, and managed/fi ltered Internet connectivity. Ultimately, IaaS allows an enterprise to scale up new software or data-based services/solutions through cloud systems quickly and without having to install massive hardware locally

**Grid computing**

The biggest security concern with grid computing is that the content of each work packet is potentially exposed to the world. Many grid computing projects are open to the world, so there is no restriction on who can run the local processing application and participate in the grid’s project. This also means that grid members could keep copies of each work packet and examine the contents. Thus, grid projects will not likely be able to maintain secrecy and are not appropriate for private, confidential, or proprietary data.

**Peerto peer -**Security concerns with P2P solutions include a perceived inducement to pirate copyrighted materials, the ability to eavesdrop on distributed content, a lack of central control/oversight/ management/filtering, and the potential for services to consume all available bandwidth

Industrial Control Systems

An industrial control system (ICS) is a form of computer-management device that controls industrial processes and machines. ICSs are used across a wide range of industries, including manufacturing, fabrication, electricity generation and distribution, water distribution, sewage processing, and oil refining. There are several forms of ICS, including distributed control systems (DCSs), programmable logic controllers (PLCs), and supervisory control and data acquisition (SCADA).

DCS units are typically found in industrial process plans where the need to gather data and implement control over a large-scale environment from a single location is essential. An important aspect of DCS is that the controlling elements are distributed across the monitored environment, **such as a manufacturing floor or a production line, and the centralized monitoring location sends commands out of** those localized controllers while gathering status and performance data. A DCS might be analog or digital in nature, depending on the task being performed or the device being controlled. For example, a **liquid flow value DCS would be an analog system whereas an electric voltage regulator DCS would likely be a digital system.**

PLC units are effectively single**-purpose or focused-purpose digital computers**. They are typically deployed for the management and **automation of various industrial electromechanical operations**, such as controlling systems on an assembly line or a large-scale digital light display (such as a giant display system in a stadium or on a Las Vegas Strip marquee).

A SCADA system can operate as a **stand-alone device, be networked together** with other SCADA systems, or be networked with traditional IT systems. Most SCADA systems are designed **with minimal human interfaces.** Often, they use mechanical buttons and **knobs r simple LCD screen interfaces (similar to what you might have on a business printer or a GPS navigation device). However, networked SCADA** devices may have more **complex remote-control software interfaces**.

**Stuxnet delivered the first-ever rootkit to a SCADA system located in a nuclear facility. Many SCADA vendors have started implementing security improvements into their solutions in order to prevent or at least reduce future compromises**

**Mobile functions**

**Full Device Encryption**

**Remote Wiping**

**Lockout**

**Screen Locks**

**GPS**

**Application Control**

**Storage Segmentation**

**Asset Tracking** Asset tracking is the management process used to maintain

**Inventory Control** The term *inventory control* may describe hardware asset tracking (

**Mobile Device Management**

The goals of MDM are to improve security, provide monitoring, enable remote management, and support troubleshooting

**Device Access Control**

**Removable Storage**

**Disabling Unused Features**

**Application Security**

**Key Management**

**Credential Management**

**Authentication**

**Geotagging -** Mobile devices with GPS support enable the embedding of geographical location in the form of latitude and longitude as well as date/time information on photos taken with these devices.

**Encryption**

**Application Whitelisting**

**BYOD Concerns**

**Data Ownership, Support Ownership, Patch Management , Antivirus Management , Forensics , Privacy , On-boarding/Off-boarding , Adherence to Corporate Policies , User Acceptance**

**Architecture/Infrastructure Considerations , Legal Concerns , Acceptable Use Policy , On-board Camera/Video**

The IoT is the collection of devices that can communicate over the Internet with one another or with a control console in order to affect and monitor the real world. IoT devices might be labeled as smart devices or smart-home equipment. Many of the ideas of industrial environmental control found in office buildings are finding their way into more consumer-available solutions for small offices or personal homes. IoT is not limited to static location equipment but can also be used in association with land, air, or water vehicles or on mobile devices.

**Layering -** By *layering* processes, you implement a structure similar to the ring model used for operating

Modes

**Abstraction**

Abstraction is one of the fundamental principles behind the field known as object-oriented programming . It is the “black-box” doctrine that says t g hat users of an object (or operating system component) don’t necessarily need to know the details of how the object works; they need to know just the proper syntax for using the object and the type of data that will be returned as a result.

**Data Hiding**

*Data hiding* is an important characteristic in multilevel secure systems

**Process Isolation**

*Process isolation* requires that the operating system provide separate memory spaces for each process’s instructions and data. It also requires that the operating system enforce those boundaries, preventing one process from reading or writing data that belongs to another process.

**Hardware Segmentation**

**Separation of Privilege**

The principle of *separation of privilege* builds on the principle of least privilege. It requires the use of granular access permissions; that is, different permissions for each type of privileged operation. This allows designers to assign some processes rights to perform certain supervisory functions without granting them unrestricted access to the system

**Covert Channels**

A *covert channel* is a method that is used to pass information over a path that is not normally used for communication

There are two basic types of covert channels:

**Covert Timing Channel** A covert timing channel conveys information by altering the performance of a system component or modifying a resource’s timing in a predictable manner. Using a covert timing channel is generally a method to secretly transfer data and is very difficult to detect

**Covert Storage Channel** A covert storage channel conveys information by writing data to a common storage area where another process can read it. When assessing the security of software, be diligent for any process that writes to any area of memory that another process can read.

**Maintenance Hooks and Privileged Programs**

Maintenance hooks are entry points into a system that are known only by the developer of the system. Such entry points are also called back doors . Although the existence of maintenance hooks is a clear violation of security policy, they still pop up in many systems. The original purpose of back doors was to provide guaranteed access to the system for maintenance reasons or if regular access was inadvertently disabled. The problem is that this type of access bypasses all security controls and provides free access to anyone who knows that the back doors exist. It is imperative that you explicitly prohibit such entry points and monitor your audit logs to uncover any activity that may indicate unauthorized administrator access.

Another common system vulnerability is the practice of executing a program whose security level is elevated during execution. Such programs must be carefully written and tested so they do not allow any exit and/or entry points that would leave a subject with a higher security rating. Ensure that all programs that operate at a high security level are accessible only to appropriate users and that they are hardened against misuse..

**Incremental Attacks**

Some forms of attack occur in slow, gradual increments rather than through obvious or recognizable attempts to compromise system security or integrity. Two such forms of attack are data diddling and the salami attack.

*Data diddling* occurs when an attacker gains access to a system and makes small, random, or incremental changes to data during storage, processing, input, output, or transaction rather than obviously altering fi le contents or damaging or deleting entire fi les. Such changes can be difficult to detect unless fi les and data are protected by encryption or unless

some kind of integrity check.

**Data diddling is often considered an attack performed more often by insiders rather than outsiders (in other words, external intruders). It should be obvious that since data diddling is an attack that alters data, it is considered an active attack.**

**The *salami attack* is more mythical by all published reports. The name of the attack refers to a systematic whittling at assets in accounts or other records with financial value, where very small amounts are deducted from balances regularly and routinely.**

The *time of check* (TOC) is the time at which the subject checks on the status of the object. There may be several decisions to make before returning to the object to access it. When the decision is made to access the object, the procedure accesses it at the *time of use* (TOU). The difference between the TOC and the TOU is sometimes large enough for an attacker to replace the original object with another object that suits their own needs. *Time-of-check-to-time-of-use* (TOCTTOU) attacks are often called *race conditions* because the attacker is racing with the legitimate process to replace the object before it is used..