CS415 Module 10 Part A - Threats and the Role of the OS

Athens State University

Outline

Contents

1 Computer Security

Computer Security

- No perfect security, but can make the cost to a perpetrator sufficiently high to deter most intruders
- Security must occur at four levels to be effective:

  Physical Data centers, servers, connected devices

  Human Avoid social engineering

  Operating System Protection mechanisms, after-the-fact debugging

  Network Preventing intercepted communication, interruption, denial-of-service

  Security is as week as the weakest link in the chain but can too much security be a problem?

The CIA Triad

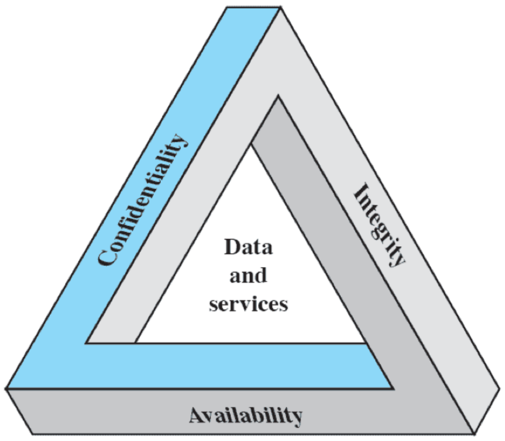


Figure 14.1 The Security Requirements Triad

- **Confidentiality**: A loss of confidentiality is the unauthorized disclosure of information
- **Integrity**: A loss of integrity is the unauthorized modification of destruction of information
- **Availability**: A loss of availability is the disruption of access to or use of information or information system

Two Additional Concepts
Authenticity

- Property of being genuine and being able to be verified and trusted
- Confidence in the validity of a transmission, a message, or message originator

Accountability

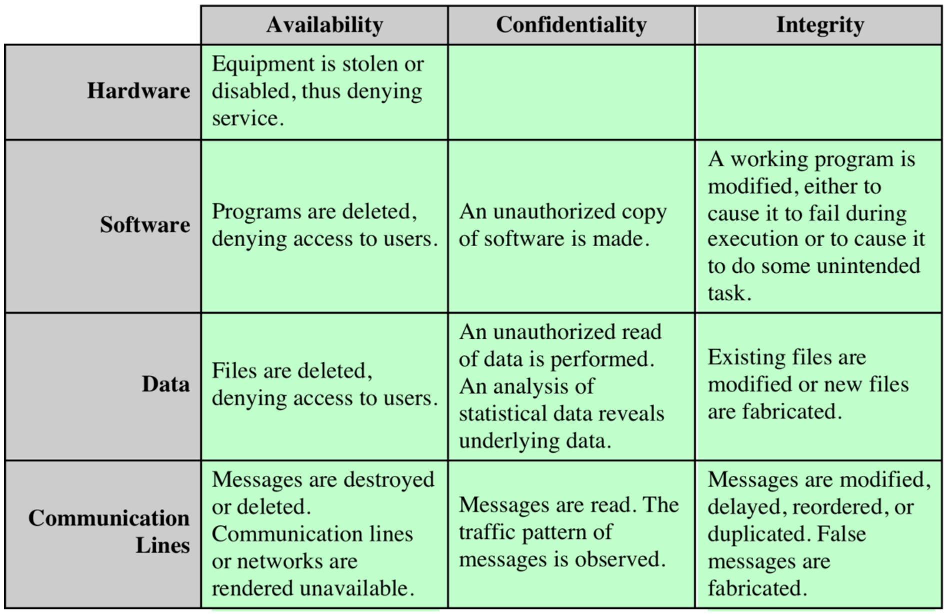
- The security goal that generates the requirement for actions of an entity to be traced uniquely to that entity
- System must keep records of their activities to permit forensic analysis

2 The Security Problem

The Security Problem

- A system is **secure** if resources used and accessed as intended under all circumstances
- **Threat**: a potential security violation
- **Attack**: an attempt to breach security
  - An attack can be accidental or malicious
  - Easier to protect against accidental than malicious misuse

Threat Examples



	Availability	Confidentiality	Integrity
Hardware	Equipment is stolen or disabled, thus denying service.		
Software	Programs are deleted, denying access to users.	An unauthorized copy of software is made.	A working program is modified, either to cause it to fail during execution or to cause it to do some unintended task.
Data	Files are deleted, denying access to users.	An unauthorized read of data is performed. An analysis of statistical data reveals underlying data.	Existing files are modified or new files are fabricated.
Communication Lines	Messages are destroyed or deleted. Communication lines or networks are rendered unavailable.	Messages are read. The traffic pattern of messages is observed.	Messages are modified, delayed, reordered, or duplicated. False messages are fabricated.

Passive Attacks

- Attempt to learn or make use of information from the system but does not affect system resources
- Goal of the attacker is to obtain information as it is transmitted
- Difficult to detect as they do not involve any alteration of data
- Emphasis is on prevention rather than detection

Network security attacks can be classified as passive attacks and active attacks. A passive attack attempts to learn or make use of information from the system but does not affect system resources. An active attack attempts to alter system resources or affect their operation. Passive attacks are in the nature of eavesdropping on, or monitoring of, transmissions. The goal of the attacker is to obtain information that is being transmitted.

Two types of passive attacks are release of message contents and traffic analysis. The concept of release of message contents , is easily understood. A telephone conversation, an electronic mail message, and a transferred file may contain sensitive or confidential information. We would like to prevent an opponent from learning the contents of these transmissions. Traffic analysis is a more subtle form of passive attack. Suppose that we had a way of masking the contents of messages or other information traffic so that opponents, even if they captured the message, could not extract the information from the message. The common technique for masking contents is encryption. If we had encryption protection in place, an opponent might still be able to observe the pattern of these messages. The opponent could determine the location and identity of communicating hosts and could observe the frequency and length of messages being exchanged. This information might be useful in guessing the nature of the communication that was taking place.

Passive attacks are very difficult to detect because they do not involve any alteration of the data. Typically, the message traffic is sent and received in an apparently normal fashion, and neither the sender nor the receiver is aware that a third party has read the messages or observed the traffic pattern. However, it is feasible to prevent the success of these attacks, usually by means of encryption. Thus, the emphasis in dealing with passive attacks is on prevention rather than detection.

Active Attacks

Replay Passive capture of a data unit and subsequent re-transmission to produce an unauthorized effect

Masquerade Takes place when one entity pretends to be a different entity

Modification of messages Some portion of a legitimate message is altered, delayed, or reordered to produce an unauthorized effect

Denial of Service
- Prevents or inhibits normal use or management of communications
- Disruption of entire network by disabling the network or overloading the network so as to degrade performance

Active attacks involve some modification of the data stream or the creation of a false stream and can be subdivided into four categories: replay, masquerade, modification of messages, and denial of service. Replay involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect. A masquerade takes place when one entity pretends to be a different entity. A masquerade attack usually includes one of the other forms of active attack. For example, authentication sequences can be captured and replayed after a valid authentication sequence has taken place, thus enabling an authorized entity with few privileges to obtain extra privileges by impersonating an entity that has those privileges. Modification of messages simply means that some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect. For example, a message stating, “Allow John Smith to read confidential file accounts ” is modified to say, “Allow Fred Brown to read confidential file accounts .” The denial of service prevents or inhibits the normal use or management of communications facilities. This attack may have a specific target; for example, an entity may suppress all messages directed to a particular destination (e.g., the security audit service). Another form of service denial is the disruption of an entire network, either by disabling the network or by overloading it with messages so as to degrade performance.

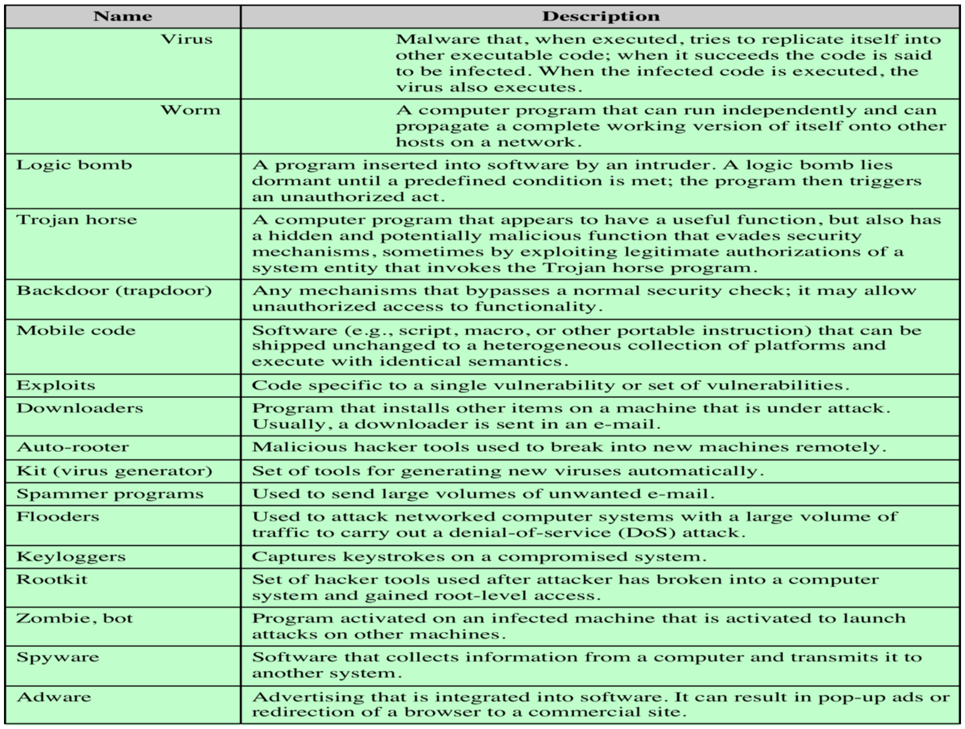
Active attacks present the opposite characteristics of passive attacks. Whereas passive attacks are difficult to detect, measures are available to prevent their success. On the other hand, it is quite difficult to prevent active attacks absolutely, because to do so would require physical protection of all communications facilities and paths at all times. Instead, the goal is to detect them and to recover from any disruption or delays caused by them. Because the detection has a deterrent effect, it may also contribute to prevention.

3 Malware

Malware

- General term for any malicious software
- Software designed to cause damage or use up the resources of a target computer
- Frequently concealed or masquerades as legitimate sofware
- In some cases, it spreads itself to other computers

Types of Malware



Name	Description
Virus	Malware that, when executed, tries to replicate itself into other executable code; when it succeeds the code is said to be infected. When the infected code is executed, the virus also executes.
Worm	A computer program that can run independently and can propagate a complete working version of itself onto other hosts on a network.
Logic bomb	A program inserted into software by an intruder. A logic bomb lies dormant until a predefined condition is met; the program then triggers an unauthorized act.
Trojan horse	A computer program that appears to have a useful function, but also has a hidden and potentially malicious function that evades security mechanisms, sometimes by exploiting legitimate authorizations of a system entity that invokes the Trojan horse program.
Backdoor (trapdoor)	Any mechanisms that bypasses a normal security check; it may allow unauthorized access to functionality.
Mobile code	Software (e.g., script, macro, or other portable instruction) that can be shipped unchanged to a heterogeneous collection of platforms and execute with identical semantics.
Exploits	Code specific to a single vulnerability or set of vulnerabilities.
Downloaders	Program that installs other items on a machine that is under attack. Usually, a downloader is sent in an e-mail.
Auto-rooter	Malicious hacker tools used to break into new machines remotely.
Kit (virus generator)	Set of tools for generating new viruses automatically.
Spammer programs	Used to send large volumes of unwanted e-mail.
Flooders	Used to attack networked computer systems with a large volume of traffic to carry out a denial-of-service (DoS) attack.
Keyloggers	Captures keystrokes on a compromised system.
Rootkit	Set of hacker tools used after attacker has broken into a computer system and gained root-level access.
Zombie, bot	Program activated on an infected machine that is activated to launch attacks on other machines.
Spyware	Software that collects information from a computer and transmits it to another system.
Adware	Advertising that is integrated into software. It can result in pop-up ads or redirection of a browser to a commercial site.

Backdoors

- A secret entry point into a program that allows someone to gain access without going through the usual security access procedures
- **Maintenance hook**: A backdoor that programmers use to debug and test programs

A backdoor,also known as a trapdoor, is a secret entry point into a program that allows someone who is aware of the backdoor to gain access without going through the usual security access procedures. Programmers have used backdoors legitimately for many years to debug and test programs; such a backdoor is called a maintenance hook . This usually is done when the programmer is developing an application that has an authentication procedure, or a long setup, requiring the user to enter many different values to run the application. To debug the program, the developer may wish to gain special privileges or to avoid all the necessary setup and authentication. The programmer may also want to ensure that there is a method of activating the program should something be wrong with the authentication procedure that is being built into the application. The backdoor is code that recognizes some special sequence of input or is triggered by being run from a certain user ID or by an unlikely sequence of events.

Backdoors become threats when unscrupulous programmers use them to gain unauthorized access. The class is example was found in the Multics operating system during the 1960s and 1970s, penetration tests were conducted by an Air Force “tiger team” (simulating adversaries). One tactic employed was to send a bogus operating system update to a site running Multics. The update contained a Trojan horse that could be activated by a backdoor and that allowed the tiger team to gain access. The threat was so well implemented that the Multics developers could not find it, even after they were informed of its presence.

It is difficult to implement operating system controls for backdoors. Security measures must focus on the program development and software update activities.

Trojan Horse

- Useful, or apparently useful, program or procedure that contains hidden code that, when invoked, performs some unwanted or harmful function
- Three types:
  1. Does the normal thing AND some separate malicious activity
  2. Acts like it's doing the normal thing but modifies the function to perform malicious activity or disguise other malicious activity
  3. Performs a malicious function that completely replaces the function of the original program

Trojan horse programs can be used to accomplish functions indirectly that an unauthorized user could not accomplish directly. For example, to gain access to the files of another user on a shared system, a user could create a Trojan horse program that, when executed, changes the invoking user's file permissions so that the files are readable by any user. The author could then induce users to run the program by placing it in a common directory and naming it such that it appears to be a useful utility program or application. An example is a program that ostensibly produces a listing of the user's files in a desirable format. After another user has run the program, the author of the program can then access the information in the user's files. An example of a Trojan horse program that would be difficult to detect is a compiler that has been modified to insert additional code into certain programs as they are compiled, such as a system login program. The code creates a backdoor in the login program that permits the author to log on to the system using a special password. This Trojan horse can never be discovered by reading the source code of the login program.

Another common motivation for the Trojan horse is data destruction. The program appears to be performing a useful function (e.g., a calculator program), but it may also be quietly deleting the user's files. For example, a CBS executive was victimized by a Trojan horse that destroyed all information contained in his computer's memory [TIME90]. The Trojan horse was implanted in a graphics routine offered on an electronic bulletin board system.

Viruses

- Software that "infects" other programs by modifying them
  - carries code to self duplicate
  - becomes embedded in a program on a computer
  - passes fresh copies of itself to other computers
- Three parts:
  - An infection mechanism
  - Trigger
  - Payload

Biological viruses are tiny scraps of genetic code—DNA or RNA—that can take over the machinery of a living cell and trick it into making thousands of flawless replicas of the original virus. Like its biological counterpart, a computer virus carries in its instructional code the recipe for making perfect copies of itself. The typical virus becomes embedded in a program on a computer. Then, whenever the infected computer comes into contact with an uninfected piece of software, a fresh copy of the virus passes into the new program. Thus, the infection can be spread from computer to computer by unsuspecting users who either swap disks or send programs to one another over a network. In a network environment, the ability to access applications and system services on other computers provides a perfect culture for the spread of a virus.

A virus can do anything that other programs do. The only difference is that it attaches itself to another program and executes secretly when the host program is run. Once a virus is executing, it can perform any function that is allowed by the privileges of the current user, such as erasing files and programs.

Bots and Botnets

- A program that secretly takes over another Internet-attached computer and uses that computer to launch further attacks
- Typically planted on many, many computers belonging to unsuspected third parties
- Collections of bots acting in coordinated manner is a **botnet**
  - Botnets exhibit three characteristics: the bot functionality, a remote control facility, and a spreading mechanism to propagate the bots and construct the botnet

Rootkits

- Set of programs installed on a system to maintain root access to that system
- Root access bypasses all of the security on the operating system
- A rootkit hides by subverting the mechanisms that monitor and report on the processes, files, and registries on a computer

Biological viruses are tiny scraps of genetic code—DNA or RNA—that can take over the machinery of a living cell and trick it into making thousands of flawless replicas of the original virus. Like its biological counterpart, a computer virus carries in its instructional code the recipe for making perfect copies of itself. The typical virus becomes embedded in a program on a computer. Then, whenever the infected computer comes into contact with an uninfected piece of software, a fresh copy of the virus passes into the new program. Thus, the infection can be spread from computer to computer by unsuspecting users who either swap disks or send programs to one another over a network. In a network environment, the ability to access applications and system services on other computers provides a perfect culture for the spread of a virus.

A virus can do anything that other programs do. The only difference is that it attaches itself to another program and executes secretly when the host program is run. Once a virus is executing, it can perform any function that is allowed by the privileges of the current user, such as erasing files and programs.

Bots and Botnets

- A program that secretly takes over another Internet-attached computer and uses that computer to launch further attacks
- Typically planted on many, many computers belonging to unsuspected third parties
- Collections of bots acting in coordinated manner is a **botnet**
  - Botnets exhibit three characteristics: the bot functionality, a remote control facility, and a spreading mechanism to propagate the bots and construct the botnet

Rootkits

- Set of programs installed on a system to maintain root access to that system
- Root access bypasses all of the security on the operating system
- A rootkit hides by subverting the mechanisms that monitor and report on the processes, files, and registries on a computer

A rootkit is a set of programs installed on a system to maintain administrator (or root) access to that system. Root access provides access to all the functions and services of the operating system. The rootkit alters the host's standard functionality in a malicious and stealthy way. With root access, an attacker has complete control of the system and can add or change programs and files, monitor processes, send and receive network traffic, and get backdoor access on demand.

A rootkit can make many changes to a system to hide its existence, making it difficult for the user to determine that the rootkit is present and to identify what changes have been made. In essence, a rootkit hides by subverting the mechanisms that monitor and report on the processes, files, and registries on a computer.

Firmware-based Rootkits

- Modern computer firmware can be "reflashed" either from storage devices or the network
- Standards such as the UEFI standard for machine firmware is quite functional, almost like small operating system onto themselves
- Rootkits may be installed into firmware either by reflashing or using the extension mechanisms provided by smart firmware (for adding new devices, for example)
- Can be very difficult to detect and eliminate.

4 Exploit frameworks

Exploit frameworks

- *Exploit framework*: a collection of tools and libraries to collect together exploits and payload that allow an attacker to quickly exploit a vulnerability
- Before development of such frameworks, attackers had to produce custom code that bound together the exploit and payload
  - Frameworks allow for separation of the exploit and payload
  - With greater flexibility

Three most common exploit frameworks

- *Metasploit*: Open source framework, originally written in Perl and recently rebuilt using Ruby; ships with 303 exploits, mostly targeted at Microsoft Windows with some OSX and Linux exploits included
- *CANVAS*: commercial tool that ships with over 400 exploits; source code included with API for user to add additional exploits
- *IMPACT*: another commercial framework with emphasis on industrial control system exploits

Vulnerability, Exploit, and Payload

Vulnerability A potential attack vector to exploit

Exploit A method for attacking a vulnerability

Payload An interchangeable piece of additional functionality or change in behavior that you want to put in place on the compromised machine.

Some frighteningly powerful payloads

- VNC DLL-induction tool: inserts a VNC remote access server into a vulnerable running process; user can than remotely control the GUI across the network
- Meterpreter: complete command-line environment that can be inserted into a running process
  - Avoids the detection risk of launching a new command-line shell on the target system
  - Keeps the attacker entirely in memory; leaves no trace on storage devices with less chance of detection by forensic analysis
  - Avoids attempts by administrators to lock down shell access
- PassiveX: Loads any ActiveX control of an attacker's choosing into a target and runs it; Launches IE, downloads the control, and then allows the attacker to download and run other commands

"Agent" payloads

- Payload loads a software driver onto the target machine that allows attacker to remotely make system calls on the target
- Attacker can now run programs on attack machine (such as a sniffer or vulnerability checker) that will actually execute on the target
- Creates opportunity for extremely effective pivot attacks

4.1 Using metasploit

The Metasploit framework

- Exploit framework built using Ruby
- Three parts:
  - Interfaces: Web-based and CLI front-end to the framework
  - Modules: Implements exploits and payloads
  - Libraries: Provides services to the modules and interfaces

Modules in the Metasploit framework

- Exploit: Proof-of-concept code to take advantage of an vulnerability
- Payload: Malicious code intended to provide and maintain access to a target system
- Auxiliaries: Set of tools that implement scanning, sniffing, fingerprinting, and other assessment tasks
- Encoders: Tools that provide means for avoiding antivirus, firewall, IDS/IPS, and other defensive by encoding the payload
- NOP: No Operation, special module used to cover one's tracks, includes a collection of NOP assembly language instructions

Interfaces

- Multiple interfaces provided, greatest amount of functionality provided by the CLI rather than Web GUIs
- Two CLIs available: *msfconsole* and *msfcli*
- Detailed use of the CLIs can be found in Offensive Security's *Metasploit Unleashed* tutorial
  - `http://www.offensive-security.com/metasploit-unleashed/`

Writing exploit modules

- True power of this sort of framework
  - Easy to add new exploits
  - Even easier to augment and enhance existing exploits
- Consider an exploit such as the EasyFTP Command Stack Buffer Overflow
  - `file:///usr/share/metasploit-framework/modules/exploits/windows/ftp/easftp_mkd_fixret.rb`