COMP 3000 Lecture #20
Chapter 16: Security
November 25, 2004
Guest Lecture: Prof. Paul Van Oorschot
Notes by Brian Fetter

Operating system functions relate to security because it is responsible for making sure things don't go wrong.

In the next 10-15 years Prof. Oorschot predicts there will be more jobs in security than in most other fields of computer science.

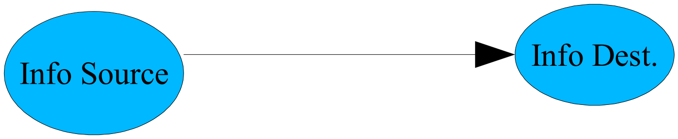
To understand security you need to know how operating systems, networking and cryptography works.

Most of security is common sense.

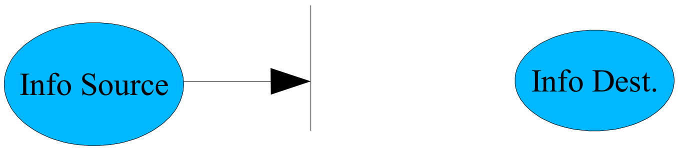
Computer & network security addresses 4 requirements: "CIA + A":

- Confidentiality: Info should be available only to authorized parties.
- Integrity: Modification of information should be allowed only by authorized parties.
- Availability: System assets should be continuously available. This deals with ensuring that attacks cannot cause disruptions in server.
- Authenticity: System should be able to identify users. Sometimes thought of as part of Confidentiality and Integrity

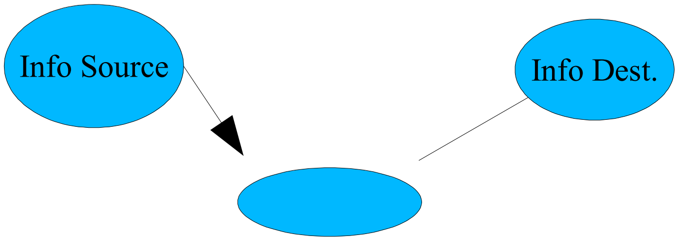
Attacks on these 4 requirements are depicted in Fig. 16.2 (p. 679)



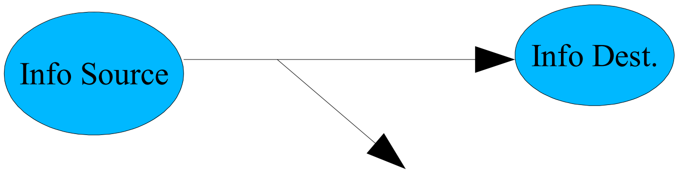
This is the proper way.



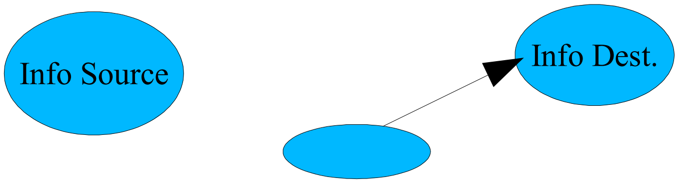
Interruption



Modification



Interception



Fabrication

What are the assets that are at risk?

- Hardware
  - Client, Server, Firewall, Router
  - Includes CPU and Memory
- Software
  - Application
  - O/S, O/S Kernel
  - Device Drivers
- Data
  - Application, O/S, drivers
- Communications links/channels
  - Connectivity
  - “Bandwidth” In the sense of the word as being the amount of data that someone can push through a network.

Network security attacks are classified as:

- Passive
  - No direct impact on system.
  - Eavesdropping on unencrypted transmissions.
  - Traffic analysis on encrypted transmissions. Useful for when the presence or absence of traffic is almost as important as the content of traffic itself.
- Active
  - The system is directly impacted in some way.
  - Someone injects a new message impersonating another party. (Masquerade)
  - Replaying a message, such as when someone tries to add additional credits to their bank account similar to a valid one in the past
  - Modification of a message, such as changing a deposit for $10 to a deposit for $1000
  - Denial of service (DOS), which causes a system to be too bogged down by false request to be able to process legitimate ones, affects availability.

16.2 Protection

Multi-programming environment -> sharing of resources.

If a program for marking is on the same machine that has the database for payroll you must make sure that memory from one program cannot interfere with data from another or a TA could change their hourly wage.

Options:

- No protection
  - Sensitive applications/procedures run at different times
- Isolation
  - Each process runs/operates separately from others
  - Each having their own address space
  - No sharing or communication between processes
- Share all or nothing
  - An object owner declares an object to be public or private
- Share via access limitation
  - O/S checks access permissions on a user-object basis
- Sharing via dynamic capabilities
  - Dynamic creation of sharing rights for objects
- Limit the use of an object
  - Access to the object as well as type of access is controlled (file Read, Write, Execute)

Protection of memory:

If we have two programs running at the same time and one program with a bug tries to access memory from other programs, this will likely cause corruption or a crash. This area deals with both intentional and unintentional attacks.

- Separating memory space between processes is done (naturally) via a virtual memory scheme. (eg. segmentation or paging or some combination)
- For complete isolation, O/S assures each segment or page accessible to only 1 process.

Look at page 686 for an example including IBM zSeries machines.

Access Control For User Accounts (userid-password)

- Userid is necessary for determining which account is being accessed
- Password is the specific password for that particular account
- Finding userids are quite easy to find, and therefore passwords must be secret.
- Password serves to authenticate the user, in the sense that if the system receives a userid and a password that is correct for that userid the system accepts that the person that entered that password is the legitimate person associated with this account.
- Passwords are problematic as they are forgotten, are sometimes revealed unintentionally, people who have the password may change from being a valid user to an invalid one, people typically choose bad passwords to facilitate remembering them.

Access Control For Objects

A login typically gives access to one machine, but it does not necessarily give you access to everything on that machine. Additional controls may need to be kept to ensure that certain other processes running on that machine that a user does not have access to require additional authentication in order to modify them.

- Most common mechanism for ensuring this is an access matrix.

	Object 1	***Object 2***	***Object 3***	**...**
Alice				
Bob				
Charlie				
...				

Access Permissions

Columns in an access matrix are access control lists for objects. These show all users allowed to use a particular object.

Rows in an access matrix are capability lists for users and specify all permissions of a particular user. This is more difficult to manage as associating everything with each user is hard to manage. Must be verifiable and unforgeable.

Objects can trust their own access control lists but it is more difficult to authenticate capability lists.

16.3 Intruders

In the world we typically are trusting. As software developers we typically test things without thinking about deliberate attack. Security experts must always think about what goes wrong.

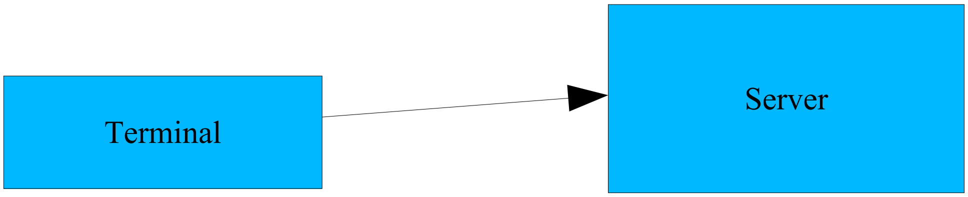
Typically designers of any product don't think too much about malicious actions, creating a tire you don't think about trying to prevent someone from cutting the tire with a knife.

Classifications:

- Masquerader -> not allowed on system, uses someone else's account (outsider)
- Misfeaser -> allowed on system but bypasses controls (insider)
- Clandestine User -> gets access to system without having any auditing or being considered a user by the system.

Easiest way to get into someone else's account is to get a hold of their userid and password.

The biggest problem with passwords is that people choose passwords can be guessed relatively easily.



Sends password to Server

/etc/password
userid .... password

Can be intercepted.

One way to fix this is to use a one-way function f to change the password p so that f(p) is sent and compared to the server. It is very difficult to get the password from this.

SSH is used to encrypt transmissions typically when used remotely so that someone can't get f(p) and then directly inject it into the system.

Another problem is that people use real words for passwords and as such dictionary attacks can be used to guess a password by going over every valid word.

One way to stop these attacks is to lock accounts after a certain number of tries. But this generates a lot of extra work for administrators. Ebay couldn't do this as it caused denial of service attacks where when people were bidding against each other they would lock their opponent out so that they could bid at the last minute uncontested.

Guessing people's password using various heuristics is also possible. Also keystroke loggers and trojan horse programs can be used to find passwords.

If you can get f(p) then you can run a dictionary attack against the function f and get the password without the server knowing anything about it.

Ways to make passwords more secure:

Proactive password checking: when creating a password checks against a dictionary to ensure the password is secure. Rule checking can be used to force rules on a user.

Reactive password: constantly tries to crack passwords already in use and changes the bad ones, but this is very inconvenient and resource intensive.

Homework Reading:

see Tables 16.2 & 16.3
pp 694 – 695

Unix Crypt, based on DES