Lecture 16: Security

CSE 120: Principles of Operating Systems
Alex C. Snoeren

HW 4 Due 6/3
Security

- Computer Security
  - Techniques for computing in the presence of adversaries
  - Three categories of security goals
    - Confidentiality: preventing unauthorized release of info
    - Integrity: preventing unauthorized modification of info
    - Availability: preventing denial of service attacks
  - Protection is about providing all three on a single machine
    - Usually considered the responsibility of the OS
    - Could also be runtime (e.g., verification in JVM)

- Cryptography
  - Techniques for communicating in the presence of adversaries
Think carefully about what you trust with your data

- If you type your password on a keyboard, you’re trusting
  - The keyboard manufacturer
  - Your computer manufacturer
  - Your OS
  - The password library
  - The application that is checking the password

- TCB = set of components (hardware, software, people) that you trust your secrets with

- Public Web kiosks should not be in your TCB
  - Should your OS? (Think about IE and Flash)
UNIX program called “login” authenticates users
- Users enter their account name, password
- Program checks password against password database
- What could go wrong?

Why would administrator trust login program?
- Inspect source code, verify what it does
- I.e., no ‘backdoors’ that allowed unexpected access
- Is the program safe?

NO. Trusted computing base includes compiler
- Ken Thompson put backdoor in original UNIX login
- Hacked the C compiler to hide his tracks
Cryptography bridges TCBs

- Enables communication between trusted parties
  - Even (especially) in the face of untrusted eavesdroppers
  - Allows systems to expand their trusted computing base
- Three main goals:
  - **Authentication**: verify the identity of the communicating party
    » Distinct from authorization (e.g., ACLs, capabilities)
  - Integrity: verify the message arrives as sender intended
  - Confidentiality: only recipient can read message
    » This is NOT the same as integrity; can have one without the other.
- Implemented with a wide family of mechanisms
  - Almost all rely on some form of “key” or secret
Cryptographic Operations

- Encryption & Decryption (confidentiality)
  - Given a message and a secret key, create a cyphertext
  - Goal is that cyphertext is confidential
    » Need the key to convert cyphertext back to message

- Signing & Verifying (Authentication & integrity)
  - Messages can be signed by the sender (often using their key)
  - Given an a (signed) message and the supposed identity of a sender, verify that this message was indeed sent by the specified party
  - Also, ensure that the message has not been modified (or duplicated)

- Main design point is whether secret keys are shared
  - Symmetric (fast, cheap) vs Public/Private key pair (easy distribute)
Design Principles

- Security is much, much more than just crypto
  - Crypto is a crucial mechanism for security, but is not a panacea
  - If there is a fundamental flaw in the design of the system, then all of the crypto in the world won’t help you
  - It is usually easier to find a bug in an implementation than circumvent a crypto system

- Unfortunately, systems design is still as much an art as it is a science
  - But, decades of building systems the wrong way have helped us collect some learned wisdom
  - We’ll cover some in the rest of this part of the lecture
Principle: Least Privilege

- Figure out exactly which capabilities a program needs to run, and grant it only those
  - Not always easy, but one algorithm: start with granting none, run and see where it breaks, add new privileges, repeat
- Unix
  - Good example: Should not normally run as root to prevent against accidents
  - Bad example: Some programs run as root just to get a small privilege, such as using a port < 1024 (privileged port)
    » E.g., ftpd
    » Exploit these programs, and you get root access to system
wu-ftpd tries to run with least privilege
  - But occasionally tries to elevate its privilege with:
    ```c
    seteuid(0);
    // privileged critical section runs here
    seteuid(getuid());
    ```

However, wu-ftpd does not disable Unix signals
  - wu-ftpd doesn’t relinquish privileges after signal handler
  - While in critical section, can be “tractor-beamed” away to a signal handler
    » Does not return to original control flow

Remote user can cause a signal handler to run by terminating a download in midstream!
  - But need to catch wu-ftpd in the critical section
  - Result: Can abort a download and then use wu-ftpd as root
Least-Common Mechanism

- Be very careful integrating shared or reused code
  - Assumptions made may no longer be valid in current context

- Counter example: Outlook and Internet Explorer
  - Windows exports an API to IE’s HTML rendering code
    - Outlook and other programs use this to display HTML in email
    - By default, JavaScript and Java parsing are enabled
  - HTML rendering code knows Java(Script) is unsafe
    - Disables it when JavaScript is downloaded from Internet
    - Only enables it when loaded from trusted sources
      - Your own file system is trusted
  - But…email is spooled on disk. D’oh!
Complete Mediation

- Check every access to every object
  - Of course, this introduces overhead
  - So, implementers try to get away with less (caching)
  - But only when nothing relevant in environment has changed

- Counter example: NFS and file handles
  - Client contacts remote “mountd” to get a file handle to a remotely exported NFS file system
    » Remote mountd checks access control at mount time
  - File handle is a capability: client presents it to read/write file
    » Client responsible for enforcing per-file restrictions
  - An eavesdropper can sniff file handle and access file system
ToCtToU

- Time of Check to time of Use
  - Check permissions as close as possible to action
- Complete mediation gets even tougher with multiprogramming
  - Attacker can execute concurrently with TCB
  - Improper synchronization can lead to race conditions
  - Period between verifying authorization and execution is a critical section
- Counter example: set-uid UNIX programs
  - Many utilities run with effective ID of root; allows regular users to perform super-user actions. May also access user’s files

```c
if (access(filename, W_OK) == 0) {
    if ((fd = open(filename, O_WRONLY)) == NULL) {
        return (0);
    }
    // Access file
```
Fail-Safe Defaults

- Deny all access first, then allow only that which has been explicitly permitted
  - Oversights will then show up as “false negatives”
    » Somebody is denied access who should have it
    » They will complain.
  - Opposites lead to “false positives”
    » Somebody is given access that shouldn’t get it
    » Not much incentive to report this kind of failure…

- Counter examples
  - SunOS shipped with “+” in /etc/hosts.equiv
    » Essentially lets anyone login as any local user to host
  - Irix shipped with “xhost +”
    » Any remote client can connect to local X server
Security through obscurity

- Attempting to gain security by hiding implementation details
- Claim: A secure system should be secure even if all implementation details are published
  - In fact, systems become more secure as people scour over implementation details and find flaws
  - Rely on mathematics and sound design to provide security
- Many well-published algorithms are still secure (e.g., SSL)

Counter example: GSM cell phones

- GSM committee designed their own crypto algorithm, but hid it from the world
  - Social engineering + reverse engineering revealed the algorithm
  - Turned out to be relatively weak, easy to subvert
Computer programs have many side effects

- A large number of these can be monitored by an adversary
- E.g., length of time a computation takes or the reason it failed
- Why do systems always ask for password even if account is unknown?

Counter example: VMS password checking flaw

- Password checking algorithm admits timing attack:
  ```java
  for (I = 0; I < password.length(); I++) {
    if (password[I] != entered_password[I])
      return false;
  }
  return true;
  ```
Outlook For The Future

- Doesn’t look bright…
  - More and more complex systems are being deployed
    » More and more lives are being trusted to them
- Bruce Schneier: 3 waves of security attacks
  - 1\textsuperscript{st} wave: physical attacks on wires and hardware
    » Physical security to defend against this
  - 2\textsuperscript{nd} wave: syntactic attacks on crypto protocols and systems
    » E.g., buffer overflows, DDoS attacks
  - 3\textsuperscript{rd} wave: semantic attacks: humans and computers trust information that they shouldn’t
    » E.g., Phishing, falsified press announcements
      - Emulex corp stock hoax: CEO “resigns”, 61% stock drop
      - Semantic attack against people with preprogrammed sell orders
Next Time

- Homework 4 due Tuesday
- TA evaluations now; CAPEs on Tuesday