Operating System Security
CS 236
On-Line MS Program
Networks and Systems Security
Peter Reiher

Outline

- What does the OS protect?
- Authentication for operating systems
- Memory protection
  - Buffer overflows
- IPC protection
  - Covert channels
- Stored data protection
  - Full disk encryption

Introduction

- Operating systems provide the lowest layer of software visible to users
- Operating systems are close to the hardware
  - Often have complete hardware access
- If the operating system isn't protected, the machine isn't protected
- Flaws in the OS generally compromise all security at higher levels

Why Is OS Security So Important?

- The OS controls access to application memory
- The OS controls scheduling of the processor
- The OS ensures that users receive the resources they ask for
- If the OS isn't doing these things securely, practically anything can go wrong
- So almost all other security systems must assume a secure OS at the bottom

Single User Vs. Multiple User Machines

- The majority of today's computers usually support a single user
- Some computers are still multi-user
  - Often specialized servers
- Single user machines often run multiple processes, though
  - Often through downloaded code
- Increasing numbers of embedded machines
  - Effectively no (human) user

Trusted Computing

- Since OS security is vital, how can we be sure our OS is secure?
- Partly a question of building in good security mechanisms
- But also a question of making sure you're running the right OS
  - And it's unaltered
- That's called *trusted computing*

How Do We Achieve Trusted Computing?

- From the bottom up
- We need hardware we can count on
- It can ensure the boot program behaves
- The boot can make sure we run the right OS
- The OS will protect at the application level

TPM and Bootstrap Security

- Trusted Platform Module (TPM)
  - Special hardware designed to improve OS security
- Proves OS was booted with a particular bootstrap loader
  - Using tamperproof HW and cryptographic techniques
- Also provides secure key storage and crypto support

TPM and the OS Itself

- Once the bootstrap loader is operating, it uses TPM to check the OS
- Essentially, ensures that expected OS was what got booted
- OS can request TPM to verify applications it runs
- Remote users can request such verifications, too

Transitive Trust in TPM

- You trust the app, because the OS says to trust it
- You trust the OS, because the bootstrap says to trust it
- You trust the bootstrap, because somebody claims it's OK
- You trust the whole chain, because you trust the TPM hardware's attestations

Trust vs. Security

- TPM doesn't guarantee security
  - It (to some extent) verifies trust
- It doesn't mean the OS and apps are secure, or even non-malicious
- It just verifies that they are versions you have said you trust
- Offers some protection against tampering with software
- But doesn't prevent other bad behavior

Status of TPM

- Hardware widely installed
  - Not widely used
- Microsoft Bitlocker uses it
  - When available
- A secure Linux boot loader and OS work with it
- Some specialized software uses TPM

SecureBoot

- A somewhat different approach to ensuring you boot the right thing
- Built into the boot hardware and SW
- Designed by Microsoft
- Essentially, only allows booting of particular OS versions

Some Details of SecureBoot

- Part of the Unified Extensible Firmware Interface (UEFI)
  - Replacement for BIOS
- Microsoft insists on HW supporting these features
- Only boots systems with pre-arranged digital signatures
- Some issues of who can set those

Authentication and Authorization in Operating Systems

- The OS must authenticate all user requests
  - Otherwise, can't control access to critical resources
- Human users log in
  - Locally or remotely
- Processes run on their behalf
  - And request resources
- Once authenticated, requests must be authorized

In-Person User Authentication

- Authenticating the physically present user
- Most frequently using password techniques
- Sometimes biometrics
- To verify that a particular person is sitting in front of keyboard and screen

Remote User Authentication

- Many users access machines remotely
- How are they authenticated?
- Most typically by password
- Sometimes via public key crypto
- Sometimes at OS level, sometimes by a particular process
  - In latter case, what is their OS identity?
  - What does that imply for security?

Process Authentication

- Successful login creates a primal process
  - Under ID of user who logged in
- The OS securely ties a process control block to the process
  - Not under user control
  - Contains owner's ID
- Processes can fork off more processes
  - Usually child process gets same ID as parent
- Usually, special system calls can change a process' ID

For Example,

- Process X wants to open file Y for read
- File Y has read permissions set for user Bill
- If process X belongs to user Bill, system ties the open call to that user
- And file system checks ID in open system call to file system permissions
- Other syscalls (e.g., RPC) similar

Authorization in Operating Systems

- Operating systems allow user processes to perform system calls
  - Which generally do things that not all users/processes should do
- When operation requires permissions, we need to check those
- When is that?
- When should the OS perform authorization?

Authorization and Reference Monitors

- If an operation requires authorization, it should pass through a reference monitor
- Reference monitors add overhead
  - So we don't want to use them unnecessarily
- But when will it be necessary?
- A question for OS design and implementation

OS Authorization Locations

- How do we decide where in the OS code we perform authorization?
- OS designers' best guess?
- Automatically identify dangerous operations?
- Identify and track sensitive data items?
- An area of active research