

## Operating Systems Design

### 17. Protection & Security

Paul Krzyzanowski  
pzk@cs.rutgers.edu

## Protection & Security

- Security
  - Prevention of unauthorized access to a system
    - Malicious or accidental access
    - "access" may be:
      - user login, a process accessing things it shouldn't, physical access
    - The access operations may be reading, destruction, or alteration
- Protection
  - The mechanism that provides and enforces controlled access of resources to processes
  - A protection mechanism *enforces* security policies

## Principle of Least Privilege

- At each abstraction layer, every element (user, process, function) should be able to access *only* the resources necessary to perform its task
- Even if an element is compromised, the scope of damage is limited
- Consider:
  - Violation: a compromised print daemon allows one to add users
  - Violation: a process can write a file even though there is no need to
  - Private member functions & Local variables in functions limit scope
  - Admin privileges set by default for any user account
- Least privilege is often difficult to define & enforce

## Privilege Separation

- Divide a program into multiple parts: high & low privilege components
- Example on POSIX systems
  - Each process has a *real* and *effective* user ID
  - Privileges are evaluated based on the effective user ID
    - Normally,  $uid == euid$
  - An executable file may be tagged with a *setuid bit*
    - `chmod +sx filename`
    - When run,  $uid = \text{user's ID}$ ;  $euid = \text{file owner's ID}$
  - Separating a program
    - Run a *setuid* program
    - Create a communication link to self (*pipe*, *socket*, *shared memory*)
    - *fork*
    - One process will call `seteuid(getuid())` to lower its privilege

## Security Goals

- **Authentication**
  - Ensure that users, machines, programs, and resources are properly identified
- **Confidentiality**
  - Prevent unauthorized access to data
- **Integrity**
  - Verify that data has not been compromised: deleted, modified, added
- **Availability**
  - Ensure that the system is accessible

## The Operating System

- The OS provides processes with access to resources

Resource	OS component
Processor(s)	Process scheduler
Memory	Memory Management + MMU
Peripheral devices	Device drivers & buffer cache
Logical persistent data	File systems
Communication networks	Sockets

- Resource access attempts go through the OS
- OS decides whether access should be granted
  - Decision = *policy*

### Domains of protection

- Processes interact with objects
  - Objects:
    - hardware (CPU, memory, I/O devices)
    - software: files, semaphores, messages, signals
- A process should be allowed to access only objects that it is authorized to access
  - A process operates in a **protection domain**
  - Protection domain defines the objects the process may access and how it may access them

### Modeling Protection: Access Matrix

- Rows: domains
- Columns: objects
- Each entry represents an access right of a domain on an object

		objects		
		$F_0$	$F_1$	Printer
domains of protection	$D_0$	read	read-write	print
	$D_1$	read-write-execute	read	
	$D_2$	read-execute		
	$D_3$		read	print
	$D_4$			print

### Access Matrix: Domain Transfers

- Switching from one domain to another is a configurable policy

		objects							
		$F_0$	$F_1$	Printer	$D_0$	$D_1$	$D_2$	$D_3$	$D_4$
domains of protection	$D_0$	read	read-write	print	-	switch	switch		
	$D_1$	read-write-execute	read			-			
	$D_2$	read-execute				switch	-		
	$D_3$		read	print					
	$D_4$			print					

### Access Matrix: Additional operations

- Copy:** allow delegation of rights
  - Copy a specific access right on an object from one domain to another
    - Rights may specify either a copy or a transfer of rights

		objects							
		$F_0$	$F_1$	Printer	$D_0$	$D_1$	$D_2$	$D_3$	$D_4$
domains of protection	$D_0$	read	read-write	print	-	switch	switch		
	$D_1$	read-write-execute	read*						
	$D_2$	read-execute				switch	-		
	$D_3$		read	print					
	$D_4$			print					

E.g., a process executing in  $D_1$  can give a read right on  $F_1$  to domain  $D_2$

### Access Matrix: Additional operations

- Owner:** allow new rights to be added or removed
  - An object may be identified as being *owned* by the domain
  - Owner can add and remove any right in any column of the object

		objects							
		$F_0$	$F_1$	Printer	$D_0$	$D_1$	$D_2$	$D_3$	$D_4$
domains of protection	$D_0$	read owner	read-write	print	-	switch	sv		
	$D_1$	read-write-execute	read*			-			
	$D_2$	read-execute				switch			
	$D_3$		read	print					
	$D_4$			print					

E.g., a process executing in  $D_0$  can give a read right on  $F_0$  to domain  $D_3$  and remove the execute right from  $D_1$

### Access Matrix: Additional operations

- Control:** change entries in a row
  - If access( $i, j$ ) includes a *control right*, then a process executing in Domain  $i$  can change access rights for Domain  $j$

		objects							
		$F_0$	$F_1$	Printer	$D_0$	$D_1$	$D_2$	$D_3$	$D_4$
domains of protection	$D_0$	read owner	read-write	print	-	switch	switch		
	$D_1$	read-write-execute	read*			-			control
	$D_2$	read-execute				switch			
	$D_3$		read	print					
	$D_4$			print					

E.g., a process executing in  $D_1$  can modify any rights in domain  $D_4$

### Implementing an access matrix

- A single table is usually impractical
  - Big size: # domains (users) x # objects (files)
  - Objects may come and go frequently
- Access Control List
  - Associate a column of the table with each object

### Implementing an access matrix

- Access Control List
  - Associate a column of the table with each object

		objects							
		F <sub>0</sub>	F <sub>1</sub>	Printer	D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	D <sub>4</sub>
domains of protection	D <sub>0</sub>	read owner	read- write	print					
	D <sub>1</sub>	read- write- execute	read*			-			
	D <sub>2</sub>	read- execute				switch	-		
	D <sub>3</sub>		read	print					
	D <sub>4</sub>			print					

ACL for file F<sub>0</sub>

### Example: Limited ACLs in POSIX systems

- Problem: an ACL takes up a varying amount of space (possibly a lot!)
  - Won't fit in an inode
- Compromise:
  - A file defines access rights for three domains: the owner, the group, and everyone else
  - Permissions
    - Read, write, execute, directory search
    - Set user ID on execution
    - Set group ID on execution
  - Default permissions set by the *umask* system call
  - chown* system call changes the object's owner
  - chmod* system call changes the object's permissions

### Example: Full ACLs in POSIX systems

- Extended attributes: stored outside of the inode
- Enumerated list of permissions on users and groups
  - Operations on all objects:
    - delete, readattr, writeattr, readextattr, writeextattr, readsecurity, writesecurity, chown
  - Operations on directories
    - list, search, add\_file, add\_subdirectory, delete\_child
  - Operations on files
    - read, write, append, execute
  - Inheritance controls

### Implementing an access matrix

- Capability List
  - Associate a row of the table with each domain

		objects							
		F <sub>0</sub>	F <sub>1</sub>	Printer	D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	D <sub>4</sub>
domains of protection	D <sub>0</sub>	read owner	read- write	print	-	switch	switch		
	D <sub>1</sub>	read- write- execute	read*			-			
	D <sub>2</sub>	read- execute				switch	-		
	D <sub>3</sub>		read	print					
	D <sub>4</sub>			print					

Capability list for domain D<sub>1</sub>

### Capability Lists

- List of objects together with the operations allowed on the objects
- Each item in the list is a *capability*: the operations allowed on a specific object
- A process presents the capability along with a request
  - Possessing the capability means that access is allowed
- A process cannot modify its capability list

## Access Control Models: MAC vs. DAC

- **DAC: Discretionary Access Control**
  - A subject (domain) can pass information onto any other subject
  - In some cases, access rights may be transferred
  - *Most systems use this*
- **MAC: Mandatory Access Control**
  - Policy is centrally controlled
  - Users cannot override the policy

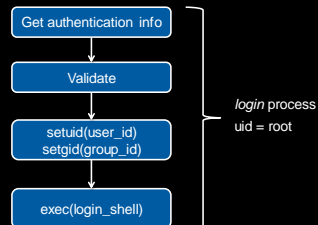
## Multi-level Access Control

- Typical MAC implementations use a **Multi-Level Secure (MLS)** access model
- **Bell-LaPadula** model
  - Identifies the ability to access and communicate data
  - Objects are classified into a hierarchy of sensitivity
    - Unclassified, Confidential, Secret, Top Secret
  - Users are assigned a clearance
  - “**No read up; no write down**”
    - Cannot read from a higher clearance level
    - Cannot write to a lower clearance level
- Works well for government information
- Does not translate well to civilian life

## Authentication

## Authentication

- Establish & verify identity
  - Then decide whether to allow access to resources
- Example: login



## Authentication

### Three factors:

- something you have *key, card*
  - can be stolen
- something you know *passwords*
  - can be guessed, shared, stolen
- something you are *biometrics*
  - costly, can be copied (sometimes)

## Authentication

### factors may be combined

- ATM machine: **2-factor authentication**
  - **ATM card** *something you have*
  - **PIN** *something you know*

## Identification vs. Authentication

---

- **Identification:**
  - Who are you?
  - User name, account number, ...
- **Authentication:**
  - Prove it!
  - Password, PIN, encrypt nonce, ...

## Versus Authorization

---

### Authorization defines access control

Once we know a user's identity:

- Allow/disallow request
- Operating system enforces system access based on user's credentials
  - Network services usually run in another context
  - Network server may not know of the user
  - Application takes responsibility
- May contact an authorization server
  - Trusted third party that will grant credentials
  - Kerberos ticket granting service
  - **RADIUS** (centralized authentication/authorization)

## Accounting

---

If security has been compromised

... *what happened?*

... *who did it?*

... *how did they do it?*

### Log transactions

- Logins
- Commands
- Database operations
- *Who looks at audits?*

### Log to remote systems

- Minimize chances for intruders to delete logs

## Auditing

---

Go through software source code and search for security holes

- Need access to source
- Experienced staff + time
- E.g., OpenBSD

Complex systems will have more bugs

- And will be harder to audit

## Threats

## You need to get into a vault

---

- Try all combinations.
- Try a subset of combinations.
- Exploit weaknesses in the lock's design.
- Open the door (drilling, torch, ...).
- Back-door access: walls, ceiling, floor.
- Observe someone else opening
  - note the combination.

## You need to get into a vault

- Ask someone for the combination.
  - Convince them that they should give it.
  - Force it (gunpoint/threat).
- Convince someone to let you in
- Find a combination lying around
- Steal a computer or file folder that has the combination.
- Look through the trash

## What can the bank do?

- Install a better lock
  - What if theirs is already good?
- Restrict physical access to the vault (guards)
  - You can still use some methods
- Make the contents of the vault less appealing
  - Store extra cash, valuables off-site
  - This just shifts the problem
- Impose strict policies on whom to trust
- Impose strict policies on how the combination is stored
  - Policies can be broken

## Computer security... then

Issue from the dawn of computing:

- Colossus at Bletchley Park: breaking codes
- ENIAC at Moore School: ballistic firing tables
- single-user, single-process systems
- data security needed
- physical security



## Computer security... now

- Sensitive data of different users lives on the same file servers
- Multiple processes on same machine
- Authentication and transactions over network
  - open for snooping
- We might want to run other people's code in our process space
  - Device drivers, media managers
  - Java applets, games
  - not just from trusted organizations

## Systems are easier to attack

### Automation

- Data gathering
- Mass mailings

### Distance

- Attack from your own home

### Sharing techniques

- Virus kits
- Hacking tools

## Cryptographic attacks

### Ciphertext-only attack

- Recover plaintext given ciphertext
- Almost never occurs: too difficult
- Brute force
- Exploit weaknesses in algorithms or in passwords

### Known plaintext attack

- Analyst has copy of plaintext & ciphertext
- E.g., Norway saying "Nothing to report"

### Chosen plaintext attack

- Analyst chooses message that gets encrypted
- E.g., start military activity in town with obscure name

## Protocol attacks

- Eavesdropping
- Active attacks
  - Insert, delete, change messages
- Man-in-the-middle attack
  - Eavesdropper intercepts
- Malicious host

## Penetration

### Guess a password

- system defaults, brute force, dictionary attack

### Crack a password

- Online vs offline
- Precomputed hashes (see [rainbow tables](#))
  - Defense: Salt

## Penetration: Guess/get a password

To access the Web-based Utility of the Router:

- Launch a web browser, such as Internet Explorer or Mozilla Firefox, and enter the Router's default IP address, 192.168.1.1, in the Address field. Press the Enter key.
- A screen will appear asking you for your User name and Password. Enter admin in the User/Name field, and enter your password (Default password is admin) in the Password field. Then click the OK button.

Address: http://192.168.1.1

Figure 6-1: Router's IP Address



Figure 6-2: Login Screen for Web-based Utility

Page 29 of the  
Linksys Wireless-N Gigabit  
Security Router with VPN  
user guide

## Penetration

### Social engineering

- people have a tendency to trust others
- *finger* sites – deduce organizational structure
- facebook, twitter, blogs, personal home pages
- look through dumpsters for information
- impersonate a user
- Phishing: impersonate a company/service

## Penetration

### Trojan horse

- program masquerades as another
- Get the user to click on something, run something, enter data

```
.....
The DCS undergrad machines are for DCS coursework only.
.....
```

```
Getting "No valid accounts?" Go to
http://zemus.rutgers.edu/newaccount.html
and add yourself back.
```

```
login: pxk
Password:
Login incorrect
```

## Trojan horse

### Disguising error messages

#### New Windows XP SP2 vulnerability exposed

Munir Kotadias  
ZDNet Australia  
November 22, 2004, 12:50 GMT

A vulnerability in Microsoft's Windows XP SP2 can allow an executable file to be run by hackers on target machines, according to security researchers

... it is possible to **craft a special error message** that is able to **bypass a security function in IE that was created to warn users** before they download potentially harmful content. ... a malicious Web site could prompt all its visitors with a standard grey dialogue box welcoming a user to the site before allowing access to the site's content. If a user clicks on the welcome box they could unknowingly install a file that gives control of their computer to a third party.

<http://tinyurl.com/5mj9f>

## Phishing

### Masqueraded e-mail



## Malicious Files and Attachments

### Take advantage of:

- Programs that automatically open attachments
- Systems that hide extensions yet use them to execute a program
- trick the user

love-letter.txt.vbs *looks like* love-letter.txt

resume.doc.scr *looks like* resume.doc

## Exploiting bugs

### Exploit software bugs

- Most (all) software is buggy
- Big programs have lots of bugs
  - *sendmail, wu-ftp*
- some big programs are *setuid* programs
  - *lpr, uucp, sendmail, mount, mkdir, eject*

### Common bugs

- buffer overflow (blindly read data into buffer)
  - e.g., *gets*
- back doors and undocumented options

## The classic buffer overflow bug

### gets.c from V6 Unix:

```
gets(s)
char *s;
{ /* gets (s) - read a string with cgetc and store in s */
  char *p;
  extern int cin;
  if (nargs () == 2)
    IEHzap("gets ");
  p=s;
  while ((*s = cgetc(cin)) != '\n' && *s != '\0')
    s++;
  if (*p == '\0') return (0);
  *s = '\0';
  return (p);
}
```

## Dealing with buffer overflows – 1

- **Executable space protection**
  - Disallow code execution on the stack or heap
  - Set MMU per-page execute permissions
- Most (all) processors support this now.

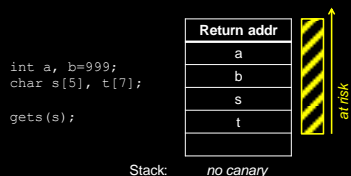
## Dealing with buffer overflows – 2

- **Address Space Layout Randomization**
  - Dynamically-loaded libraries used to be loaded in the same place each time ... ditto for the stack & memory-mapped files
  - Well-known locations make them branch targets in a buffer overflow attack
  - Position the stack and memory-mapped files (including libraries) to random locations
  - Position-independent executables can have their code placed randomly
  - Implemented in
    - OpenBSD, Windows Vista+, Windows Server 2008, Linux 2.6.15, OS X (sort of)

## Dealing with buffer overflows – 3

### Stack canaries

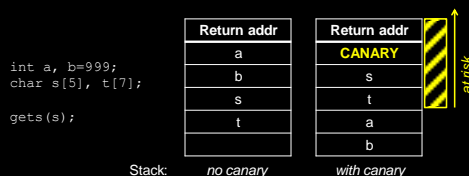
- Place a random integer before the return address on the stack
- Before a return, check that the integer is there and not overwritten: a buffer overflow attack will likely overwrite it



## Dealing with buffer overflows – 3

### Stack canaries

- Place a random integer before the return address on the stack
- Before a return, check that the integer is there and not overwritten: a buffer overflow attack will likely overwrite it
- Allocate arrays into higher memory in the stack so they won't clobber other automatic variables



## Mistakes (?)

### HP admits to selling infected flash-floppy drives

Hybrid devices for ProLiant servers pre-infected with worms, HP says  
 Gregg Keizer 08/04/2008 07:08:06

Hewlett-Packard has been selling USB-based hybrid flash-floppy drives that were pre-infected with malware, the company said last week in a security bulletin.

Dubbed "HP USB Floppy Drive Key," the device is a combination flash drive and compact floppy drive, and is designed to work with various models of HP's ProLiant Server line. HP sells two versions of the drive, one with 256MB of flash capacity, the other with 1GB of storage space

Seriously bad when combined with Windows' autorun when a USB drive is plugged in!

- This feature cannot be disabled easily

<http://tinyurl.com/5sddlq>

## Penetration: the network

### Fake ICMP, RIP packets (router information protocol)

### Address spoofing

- Fake a server to believe it's talking to a trusted machine

### ARP cache poisoning

- No authentication in ARP; blindly trust replies
- Malicious host can provide its own Ethernet address for another machine.

## Penetration: the network

### Session hijacking

- sequence number attack:** fake source address and TCP sequence number responses

## Penetration

### UDP

- no handshakes, no sequence numbers
- easy to spoof

## Penetration

Many **network services** have holes

- fake email with SMTP
- *sendmail* bugs
- snoop on *telnet* sessions
- *finger*
  - old versions have *gets* buffer overflow
  - social engineering
- unauthenticated RPC
  - access remote procedures
  - fake *portmapper*, causing your programs to run instead of real service

## Penetration

### IE

- Malformed URLs
- Buffer overflows
- ActiveX flaws
- PNG display bugs
- Jscript
- Processing of XML object data tags
- Registry modification to redirect URLs

## Penetration

### NFS

- stateless design
- once you have a file handle, you can access files or mount the file system in the future
- data not encrypted

### rlogin, rsh

- modify *.rhosts* or */etc/hosts.equiv*
- snoop on session
- fake your machine or user name to take advantage of *.rhosts*

## Penetration

### X windows

- tap into server connection (port 6000+small int) [hard!]
  - get key strokes, contents of display

### Remote administration servers

- E.g. Microsoft BackOffice

### Java applets

### Visual Basic scripts

### Shell script bugs

### URL hacking

- SQL injection

### et cetera, et cetera ....

## Denial of Service: SYN Flooding

### SYN flooding

take a machine out of service

### Background:

3-way handshake to set up TCP connection

- Send **SYN** packet
  - receiver allocates resources – limit to number of connections
  - new connections go to backlog queue
  - further SYN packets get dropped

Receiver sends *acknowledgement (SYN/ACK)* and waits for an ACK

Sender sends **ACK**

## Denial of Service: SYN Flooding

- Send SYN masqueraded to come from an unreachable host
  - receiver times tries to send SYN/ACK
  - times out eventually
    - 23 minutes on old Linux systems
    - BSD uses a Maximum Segment Life = 7.5 sec
    - Windows server 2003 recommends 120 sec.

## Denial of Service and DDoS Attacks

- Other denial of service attacks:
  - Software bugs (esp. OS)
  - ICMP floods
  - ICMP or RIP redirect messages to alter routes to imposter machines
  - UDP floods
  - application floods
- **Distributed Denial of Service (DDoS)** attacks
  - Multiple compromised machines attack a system (e.g., MyDoom)

## Direct System Access

- Boot alternate OS to bypass OS logins
  - E.g., Linux on a CD
- Third-party drivers with backdoors or bugs
- Then ... modify system files
  - Encrypted file system can help
- Rogue administrators

## Worms

Type of process that spawns copies of itself

- potentially using system resources and hurting performance
- possibly exploiting weaknesses in the operating system to cause damage

## Example: 1988 Internet worm

### Robert Tappan Morris Jr.'s Internet worm

- exploit *finger's gets* bug to load a small program (99 lines of C)
- program connects to sender and downloads the full worm
- worm searches for other machines:
  - .host files
  - finger daemon
  - sendmail DEBUG mode
  - password guessing via dictionary attack: 432 common passwords and combinations of account name and user name

## Virus

- Does not run as a self-contained process
- code is attached onto another program or script
- **File infector**
  - primarily a problem on systems without adequate protection mechanisms
- **Boot-sector**
- **Macro** (most common ... visual basic scripting)
- **Hypervisor**
  - install on virtual machines (newest form of attack)

## Virus scanning

- Search for a "**signature**"
  - Stream of bits in a virus that (we hope!) is unique to the virus and not any legitimate code
  - *NOT a cryptographic signature!*
- Some viruses are encrypted
  - Signature is either the code that does the decryption or the scanner must be smart enough to decrypt the virus
- Some viruses mutate to change their code every time they infect another system
  - Run the code through an emulator to detect the mutation

## Virus scanning

- You don't want to scan through hundreds of thousands of files
  - Search in critical places likely to be infected (e.g., \windows\system32 or removable media)
- **Passive disk scan** vs. **active I/O scan** ... or both

## Key loggers

- Record every keystroke
- Windows **hook** mechanism
  - Procedure to intercept message traffic before it reaches a target windows procedure
  - Can be chained
  - Installed via **SetWindowsHookEx**
    - **WH\_KEYBOARD** and **WH\_MOUSE**
      - Capture key up, down events and mouse events
- Hardware loggers



## Rootkits

- Replacement commands (or parts of OS) to hide the presence of an intruder
  - ps, ls, who, netstat, ...
- Hide the presence of a user or additional software (backdoors, key loggers, sniffers)
- OS can no longer be trusted!

E.g., Sony BMG DRM rootkit (October 2005)

- Creates hidden directory; installs several of its own device drivers; reroutes Windows system calls to its own routines
- Intercepts kernel-level APIs and disguises its presence with cloaking (hides \$\$ys\$ files)

## Dealing With Rootkits

- Restrict permission to modify system files
  - To avoid installing a rootkit in the first place
  - But users often grant permissions during installation
    - And permissions may be needed for drivers
- Signed software and operating system components
  - Microsoft Vista & Windows 7:
    - Requires kernel-mode software to have a digital signature (x64-based systems only)
- Tripwire
  - Software to monitor for changes in files and components in a system

The End