

CSE 120: Principles of Operating Systems

Lecture 12

Protection and Security

November 25, 2003

Prof. Joe Pasquale

Department of Computer Science and Engineering
University of California, San Diego

© 2003 by Joseph Pasquale

1

Before We Begin ...

Read Chapters 18 and 19 (on Protection and Security)

2

Protection

Resources need to be protected

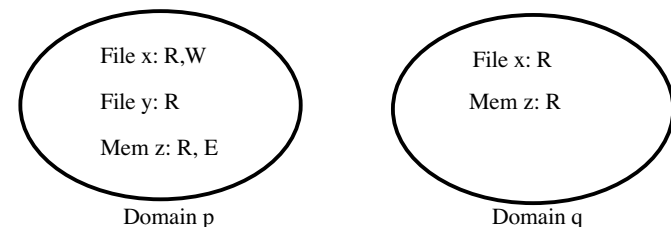
- processes without permission
- improper use by a process

Concepts

- resource
- allowed operations on resources
- user (of a resource): process
- ownership
- permissions or rights

3

Protection Domains



Protection: how to control/limit access to an object

Object: an entity (resource) that requires protection

Domain: a set of (object: rights/permissions)

Process: accesses objects, executes in some domain

4

Protection Matrix

domain \ object	File x	File y	File z	Dom 1	Dom 2	Dom 3
1	R, W	R, W	R, W			
2		R		Enter		
3	W			Enter		

Can describe all domains as a matrix

- rows are domains
- columns are objects
- matrix entry [d, o] contains rights/permissions

5

Access Control Lists

Associate list of (domain, rights) with each object

When an object is accessed, check if

- domain listed and operation permitted

If domain has no rights, don't list; min specification

Can be inefficient

- requires look up (domain, rights) each access

Revocation is easy; just remove from list

6

Capability Lists

Associate list of (object, rights) with each domain

Capability: item that signifies access to object

- [object type, rights, ptr to object]

Access is determined by having capability

Efficient: on access, just produce capability

Hard to revoke

7

Unix Protection Domains

Domain identified by

- uid, gid, user/kernel

Determines

- what files and devices you have access to
- what programs you can run
- what processes you can send signals to

Switch domains by

- entering/exiting kernel
- exec'ing a SETUID or SETGID file

8

Unix Protection

Associated with each file are permissions

- rwxrwxrwx for owner, group, other
- limited form of access control list

When a process tries to access a file

- for this domain (uid, gid), is access permitted

If access is given, provide process with a capability

- future operations are then carried out efficiently

9

How SETUID Works

Say process x is running as (uid_a, gid_a)

Executable file y with SETUID bit on

- file is owned by (uid_b, gid_b)

If process x execs file y

- process x changes domains to (uid_b, gid_b)

Used to build protected subsystems

- provide controlled access to data
- rather than direct access, access only via program

10

What is Security

Aspects

- Confidentiality: disclosure only to those authorized
- Integrity: only authorized alterations
- Authenticity
- Availability

Threats

- Interception: eavesdropping
- Interruption: destroying, denial of service
- Modification: tampering with data or programs
- Fabrication: new data or programs, replaying msgs

11

Policy vs. Mechanisms

Policy

- Specification of security for a particular system
- What each entity is allowed to do to each object
- To realize the policy, mechanisms are needed

Mechanisms

- Encryption
- Authentication
- Authorization
- Auditing

12

User Authentication

Passwords are most common method

- user and computer know secret
- user proves knowledge of secret, computer checks

Encrypted passwords

- computer stores only encrypted passwords
- user gives password; computer encrypts, checks

13

Problem with Passwords

Ex: alphabetic password (26+26), and can check 1/ μ sec

- 5 chars $52^5 = 380$ million 6.3 minutes
- 6 chars $52^6 = 19.7$ billion 5.5 hours
- 7 chars $52^7 = 1.03$ trillion 12 days
- 8 chars $52^8 = 53.5$ trillion 1.7 years

But most of these are uncommon, difficult to remember

- Using 200,000 words from dictionary: 0.2 seconds!

14

Challenge/Response Protocols

Challenge/response, algorithmic passwords

- user and system know secret algorithm
- system challenges user's knowledge, user responds

Ex: say secret algorithm is $f(x) = x^2$

- system challenges user: sends user a value, say 3
- user computes $f(3) = 9$: sends system 9
- system concludes user must know secret algorithm
- next time, system can provide different challenge
- secret never transmitted, only challenge/response

15

Trojan Horse

Program that contains hidden malicious code

- User runs program, thinking it does one thing
- In actuality, it does something else, generally bad

Since program runs as process in user's domain

- can do harm to user's environment
- can do harm under that user's name

16

Trap Door

Secret access point in program

Designer develops program, gives to someone else

Once program loaded in system, designer can access

Consider if trap door is added by compiler

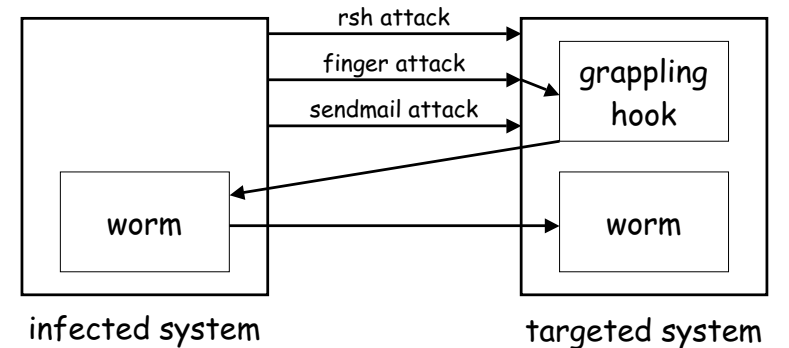
- compiler adds trap doors to programs
- designer of compiler can then access
- can't tell from program source code
- even if new compiler written, needs to be compiled!

17

Internet Worm

Worm: program that duplicates itself over network

Internet worm (Nov 2, 1988)



18

Viruses

Code fragment that gets attached to legit program

When program runs, the virus

- causes damage
- spreads, attaching itself to other programs

Disinfectants

- check for programs that look different from what they should be
- check for known virus patterns in programs

19

Denial of Service

Preventing others from using system

- by using lots of resources
- by bombarding with network requests or traffic

Example

- repeatedly request TCP connection
- don't answer responses; system times them out
- eventually, no TCP ports left available

20

Intrusion Detection

Detecting if there is an intruder, or an attack

Signature-based

- look for specific patterns of attack behavior
- ex: repeated login attempts

Anomaly-based

- look for unusual behavior
- ex: processes with unusual system call patterns

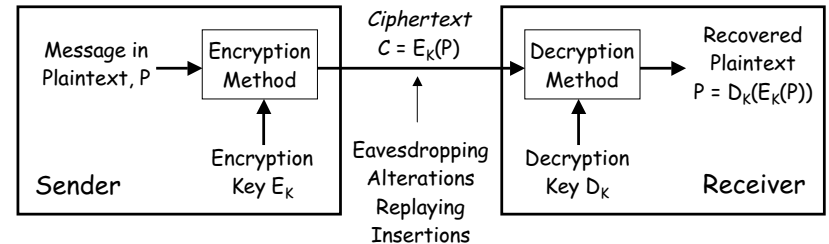
Solving: logging to create an audit trail, then analyze

21

Cryptography

Encoding messages to

- limit who can view the original message
- determine who sent a message



22

Secret Key Cryptosystem

Secret key (symmetric)

- Same key K is used to encrypt and decrypt
- Sender encrypts $E_K(P)$, Receiver decrypts $D_K(E_K(P))$

DES (Data Encryption Standard) is an example

23

Public Key Encryption

Public key (asymmetric)

- Different keys to encrypt and decrypt
- Each user has two keys: one public and one private

If A wants to send to B

- A encrypts using B's public key
- B decrypts using its private key

RSA (Rivest, Shamir, and Adelman) is an example

24

Public Key vs. Secret Key

Secret key

- operates fast
- difficult to distribute keys

Public key

- Time-consuming operation
- Convenient for key distribution

Combination

- Public key to start session: exchange secret key
- During session, use secret key

25

Authentication Using Secret Key

Alice sends Bob identity: "Hi Bob, It's me, Alice"

Bob sends "challenge" to Alice: random number R_B

Alice encrypts R_B using K : $E_K(R_B)$; sends to Bob

- Bob decrypts $E_K(R_B)$ using K ; must be Alice

Alice sends "challenge" to Bob: random number R_A

Bob encrypts R_A using K : $E_K(R_A)$; sends to Alice

- Alice decrypts $E_K(R_A)$ using K ; must be Bob

26

Authentication using Public Key

Alice uses Bob's public key: sends $K_{B, \text{pub}}(A, R_A)$ to Bob

Bob receives and does the following

- Decrypts: $K_{B, \text{priv}}(K_{B, \text{pub}}(A, R_A)) \rightarrow A, R_A$
- Encrypts and sends $K_{A, \text{pub}}(R_A, R_B, K)$ to Alice

Alice receives and does the following

- Decrypts: $K_{A, \text{priv}}(K_{A, \text{pub}}(R_A, R_B, K))$; "it's Bob"
- Encrypts and sends $K(R_B)$ to Bob

Bob receives and decrypts $K(K(R_B)) = R_B$; "it's Alice"

27

Digital Signatures

If Alice wants to digitally sign message M to Bob

- encrypt using $K_{A, \text{priv}}$; sends $K_{A, \text{priv}}(M)$ to Bob

When Bob receives, decrypts using $K_{A, \text{pub}}$

- can decrypt only if from Alice; Alice signed it

To sign and keep private

- Alice sends $K_{B, \text{pub}}(M, K_{A, \text{priv}}(M))$ to Bob
- Only Bob can decrypt: $K_{B, \text{priv}}(K_{B, \text{pub}}(M, K_{A, \text{priv}}(M)))$
- decrypts using $K_{A, \text{pub}}$ proving Alice signed it

28