

Outline for Today's Lecture

Administrative:

- More tests to give back
- Homeworks to return
- Final? Mostly since midterm, token from before, Nachos question.

Objective: Protection mechanisms

- Review of access control (allowing you to do only what you are supposed to be able to do)
- Authentication (knowing you are who you say you are)

The Security Environment

Threats

Goal	Threat
Data confidentiality	Exposure of data
Data integrity	Tampering with data
System availability	Denial of service

Security goals and threats

Intruders

Common Categories

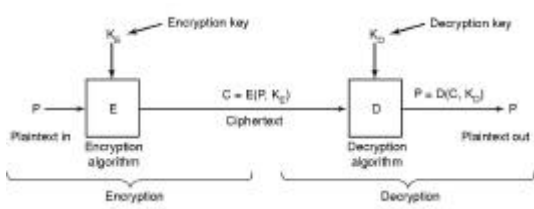
1. Casual prying by nontechnical users
2. Snooping by insiders
3. Determined attempt to make money
4. Commercial or military espionage

Accidental Data Loss

Common Causes

1. Acts of God
 - fires, floods, wars
2. Hardware or software errors
 - CPU malfunction, bad disk, program bugs
3. Human errors
 - data entry, wrong tape mounted, rm *

Basics of Cryptography



Relationship between the plaintext and the ciphertext

Secret-Key Cryptography

- Monoalphabetic substitution
 - each letter replaced by different letter
- Given the encryption key,
 - easy to find decryption key
- Secret-key crypto called symmetric-key crypto

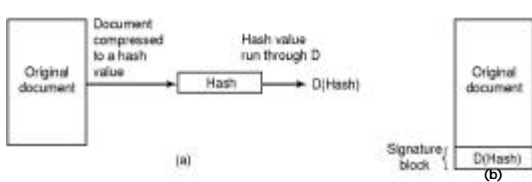
Public-Key Cryptography

- All users pick a public key/private key pair
 - publish the public key
 - private key not published
- Public key is the encryption key
 - private key is the decryption key

One-Way Functions

- Function such that given formula for $f(x)$
 - easy to evaluate $y = f(x)$
- But given y
 - computationally infeasible to find x

Digital Signatures



- Computing a signature block
- What the receiver gets

User Authentication

Basic Principles. Authentication must identify:

1. Something the user knows
2. Something the user has
3. Something the user is

This is done before user can use the system

Authentication Using Passwords

LOGIN: ken
PASSWORD: FooBar
SUCCESSFUL LOGIN

(a)

LOGIN: carol
INVALID LOGIN NAME
LOGIN:

(b)

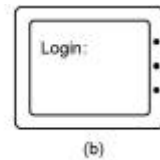
LOGIN: carol
PASSWORD: Idunno
INVALID LOGIN
LOGIN:
(c)

- (a) A successful login
(b) Login rejected after name entered
(c) Login rejected after name and password typed

Login Spoofing



(a)



(b)

- (a) Correct login screen
(b) Phony login screen

Authentication Using Passwords

```

LBL> telnet elxsi
ELXSI AT LBL
LOGIN: root
PASSWORD: root
INCORRECT PASSWORD, TRY AGAIN
LOGIN: guest
PASSWORD: guest
INCORRECT PASSWORD, TRY AGAIN
LOGIN: uuucp
PASSWORD: uuucp
WELCOME TO THE ELXSI COMPUTER AT LBL
    
```

- How a cracker broke into LBL
 - a U.S. Dept. of Energy research lab

Authentication Using Passwords

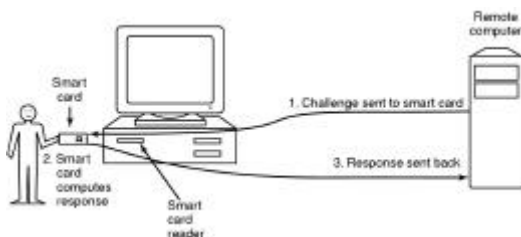
Bobbie, 4238, e(Dog4238)
Tony, 2918, e(6%%TaeFF2918)
Laura, 6902, e(Shakespeare6902)
Mark, 1694, e(XaB@Bwcz1694)
Deborah, 1092, e(LordByron,1092)

Salt

Password

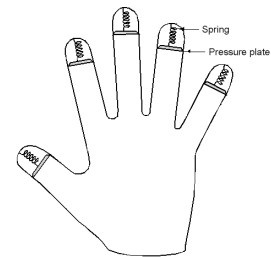
The use of salt to defeat precomputation of encrypted passwords

Authentication Using a Physical Object



- Magnetic cards
 - magnetic stripe cards
 - chip cards: stored value cards, smart cards

Authentication Using Biometrics



A device for measuring finger length.

Countermeasures

- Limiting times when someone can log in
- Automatic callback at number prespecified
- Limited number of login tries
- A database of all logins
- Simple login name/password as a trap
 - security personnel notified when attacker bites

Protection Mechanisms

Protection Domains



Examples of three protection domains

The Access Model

- Access problems can be represented abstractly by of an *access model*.
 - each row represents a subject/principal/domain
 - each column represents an object
 - each cell: accesses permitted for the $\{subject, object\}$ pair
 - read, write, delete, execute, search, control, or any other method
- In real systems, the access matrix is sparse and dynamic.
 - need a flexible, efficient representation

Protection Domains

Domain	Object							
	File1	File2	File3	File4	File5	File6	Printer1	Plotter2
1	Read	Read Write						
2			Read	Read Write Execute	Read Write		Write	
3						Read Write Execute	Write	Write

A protection matrix

Protection Domains

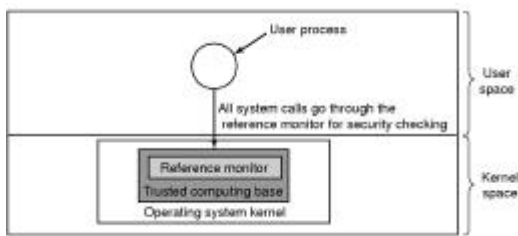
User	Object											
	File1	File2	File3	File4	File5	File6	Printer1	Printer2	Domain1	Domain2	Domain3	
1	Read	Read Write									Enter	
2			Read	Read Write Execute	Read Write		Write					
3						Read Write Execute	Write	Write				

A protection matrix with domains as objects

Access Control

- Access control lists - detailed list attached to file of users allowed (denied) access, including kind of access allowed/denied.
- UNIX RWX - owner, group, everyone
- Capabilities – permitted accesses associated with subject (user), similar to an address space.
 - Un-forgable object reference, like a pointer.

Trusted Systems Trusted Computing Base

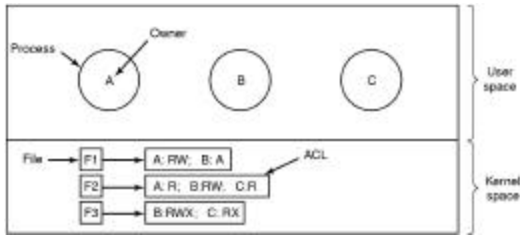


A reference monitor

Access Control Lists

- *Approach*: represent the access matrix by storing its columns with the objects.
 - Tag each object with an *access control list* (ACL) of authorized subjects/principals.
- To authorize an access requested by *S* for *O*
 - search *O*'s ACL for an entry matching *S*
 - compare requested access with permitted access
 - access checks are often made only at bind time

Access Control Lists



Use of access control lists of manage file access

Access Control Lists

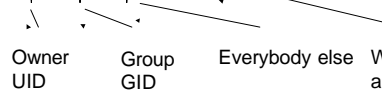
File	Access control list
Password	tana, sysadm: RW
Pigeon_data	bill, pigfan: RW; tana, pigfan: RW; ...

Two access control lists

UNI X access control

- Each file carries its access control with it.

`rwX rwx rwX setuid`



Owner UID Group GID Everybody else

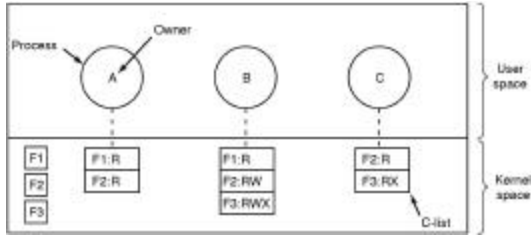
When bit set, it allows process executing object to assume UID of owner temporarily - **enter** owner domain (rights amplification)

- Owner has `chmod`, `chgrp` rights (granting, revoking)

Capabilities

- **Approach:** represent the access matrix by storing its rows with the subjects.
 - Tag each subject with a list of *capabilities* for the objects it is permitted to access.
 - A *capability* is an unforgeable object reference, like a pointer.
 - It endows the holder with permission to operate on the object
 - e.g., permission to invoke specific methods
 - Typically, capabilities may be passed from one subject to another.
 - Rights propagation and confinement problems

Capabilities



Each process has a capability list

Capabilities

- Cryptographically-protected capability

Server	Object	Rights	f(Objects, Rights, Check)
--------	--------	--------	---------------------------

- Generic Rights
 1. Copy capability
 2. Copy object
 3. Remove capability
 4. Destroy object

Calling Mechanism

- Assume a capability to execute a procedure (lpr)
- Parameter is a capability to an object of mine (read access to file foo)
- lpr's c-list has a template capability for the parameter, requires that certain rights are passed in (if lpr can't read the file foo, it obviously can't do its job)
- Sometimes, need rights amplification to do the job.

Dynamics of Protection Schemes

- How to endow software modules with appropriate privilege?
 - What mechanism exists to bind principals with subjects?
 - e.g., setuid syscall, setuid bit
 - What principals should a software module bind to?
 - privilege of creator: but may not be sufficient to perform the service
 - privilege of owner or system: dangerous

Dynamics of Protection Schemes

Problems:

- **Mutual Suspicion**

When a caller requests a service, the callee should be limited in what stuff of the caller's it has access to (only what it is passed). (entering domain - setuid vs. passing in single object)

- **Modification**

When caller does pass an object as parameter, we want to control whether the callee can modify it (I send you a photo, you remove my copyright mark)

- **Conservation**

When caller passes an object to gain service, we want to know that the callee doesn't retain or even pass on the right to the object after we think they are done with it.
(our credit card number - and selling such databases)

- **Confinement**

We want to prevent *information* leaks (besides the objects themselves)

Protection Domains

- Processes execute in a protection domain, initially inherited from subject
- Goal: to be able to change protection domains
- Introduce a level of indirection
- Domains become protected objects with operations defined on them: **owner, copy, control**

	gradefile	solutions	proj1	luv/tr	hotgossip	Domain0
TA	rw	rwo	rx	r		ctl
grp		r	rw			enter
Terry					rw	
Lynn			rw	rw		
Domain0			r			

35

- If domain contains **copy** on right to some object, then it can *transfer* that right to the object to another domain.

- If domain is **owner** of some object, it can *grant* that right to the object, with or without **copy** to another domain

- If domain is **owner** or has **ctl** right to a domain, it can *remove* right to object from that domain

- Rights propagation.

	gradefile	solutions	proj1	luv/tr	hotgossip	Domain0
TA	rw	rwo	rc	r		ctl
grp		r	rwo			enter
Terry			rc		rw	
Lynn				rw	rw	enter
Domain0			r			

36

Dynamics of Protection Schemes

Problems:

- **Mutual Suspicion**

enter my domain vs. me, as owner, granting right to another domain (temp)

- **Modification**

rights for particular operations

- **Conservation**

ctl - being able to revoke rights; not granting copy rights

- **Confinement**

remove rights to write or create anything else?