Security

Profs. Bracy and Van Renesse

based on slides by Prof. Sirer

Security in the real world

- Security decisions based on:
  - Value: How much is it worth?
  - Locks: How hard is it to circumvent protection?
  - Police: What are the repercussions of getting caught?
- Some observations:
  - Security involves overhead
    - Cost in maintaining security
    - Reduces ease of access
  - Security is only as good as the weakest link

Security in Computer Systems

- In computer systems, this translates to:
  - **Au**thentication: who?
  - **Au**thorization: what?
  - **Au**dit when? (aka accounting)
- Gold Standard for Security (Lampson)
- Another good trio:
  - **C**onfidentiality: no leaking of data (aka secrecy)
  - **I**ntegrity: no tampering of data
  - **A**vailability: no denial of service
- *Privacy* is slightly different:
    - no abusing of user's information
    - For example: doctor can read medical record, but not tell about it to others

Principle of Least Privilege

Every program and every privileged user of the system should operate using the least amount of privilege necessary to complete the job.

- Jerome Saltzer

(same person of end-to-end argument)

Security Threats

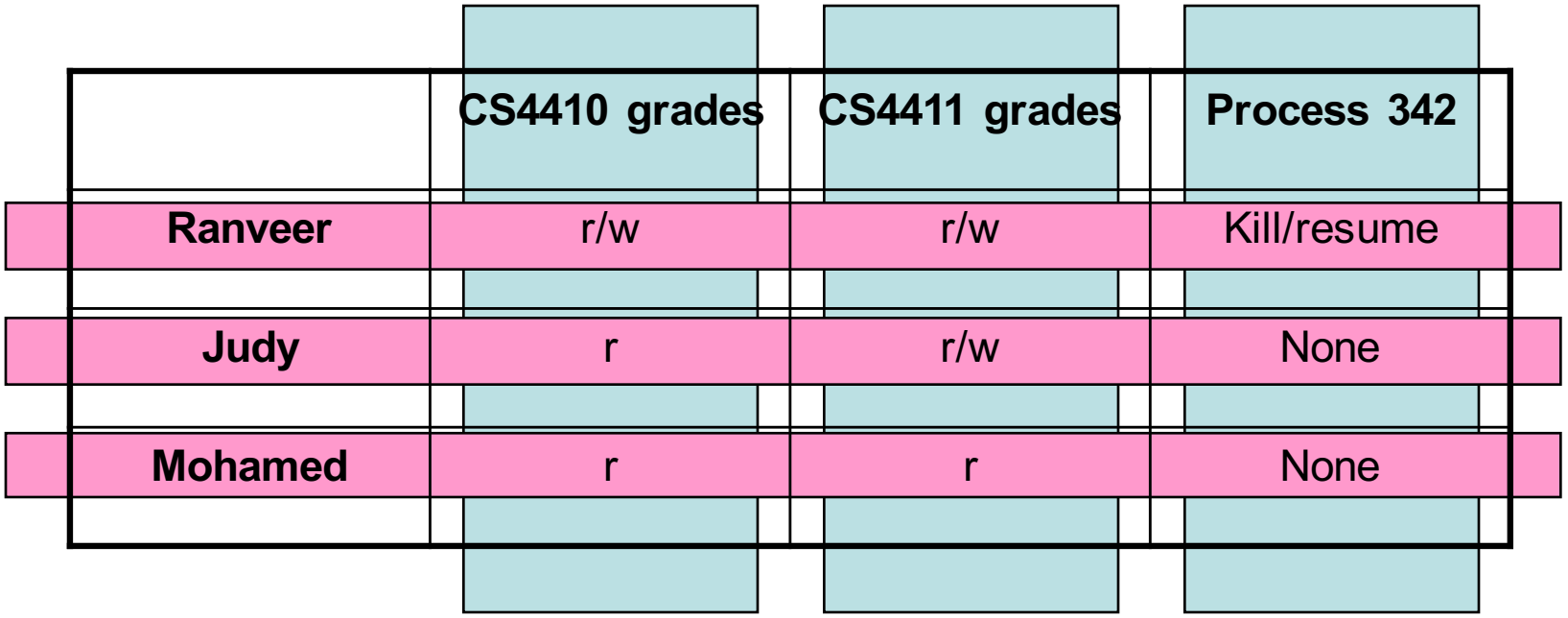
Identified by Defense Science Board:

- Incomplete, inquisitive, and unintentional blunders.
- Hackers driven by technical challenges.
- Disgruntled employees or customers seeking revenge (“insider threats”).
- Criminals interested in personal financial gain or stealing services.
- Organized crime with the intent of hiding something.
- Terrorist groups.
- Espionage agents.
- Tactical countermeasures intended to disrupt military defense.
- Multifaceted tactical information warfare.

Protection: ACLs & Capabilities

Access Control Matrix

- *Access Control List*: column for each object stored as a list for the object
- *Capabilities*: row for each subject stored as list for the subject

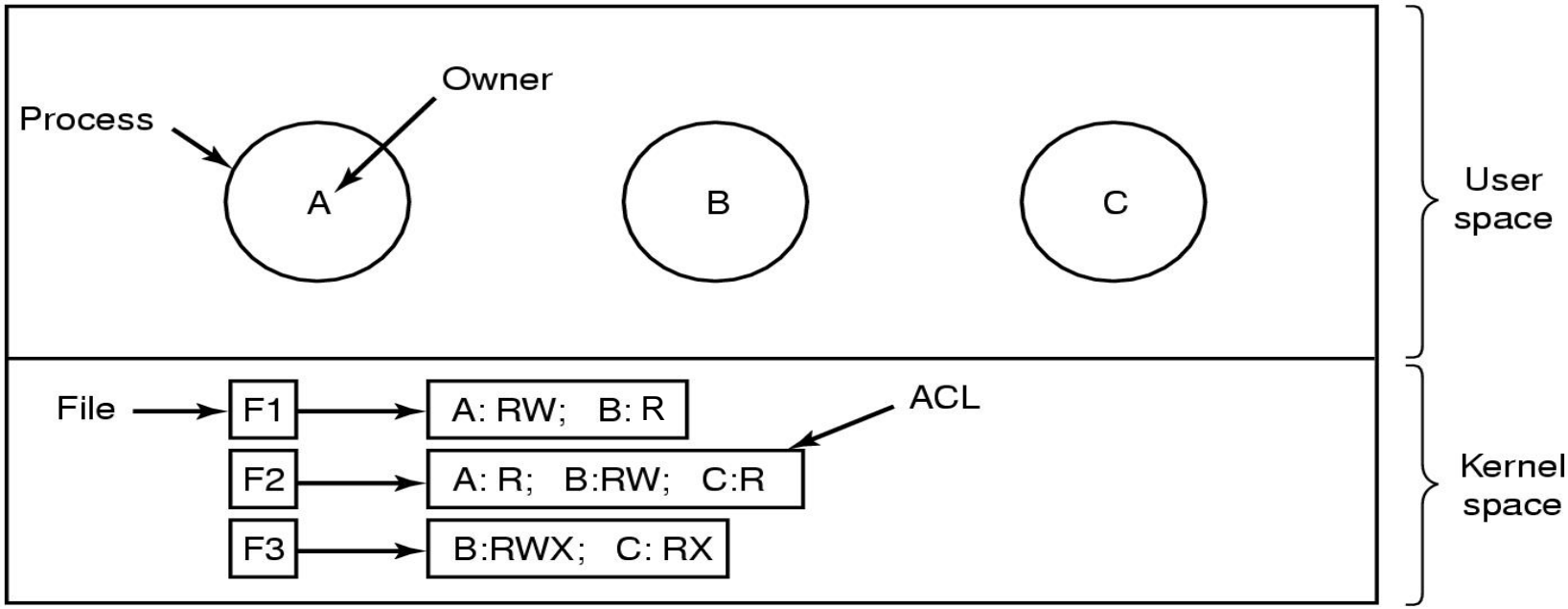


	CS4410 grades	CS4411 grades	Process 342
Ranveer	r/w	r/w	Kill/resume
Judy	r	r/w	None
Mohamed	r	r	None

Access Control Lists

Example: to control file access

– Each file has an ACL associated with it



Categories of Users (“Roles”)
-- Authentication --

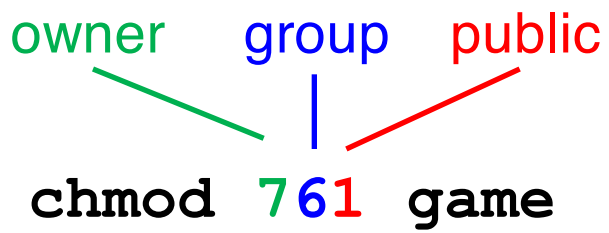
- Individual user
  - Log in establishes a user-id
  - Might be just local on the computer or could be through interaction with a network service
- Groups to which the user belongs
  - For example, “einar” is in “facres”
  - Again could just be local or could involve talking to a service that might assign, say, a temporary cryptographic key

Linux Access Rights

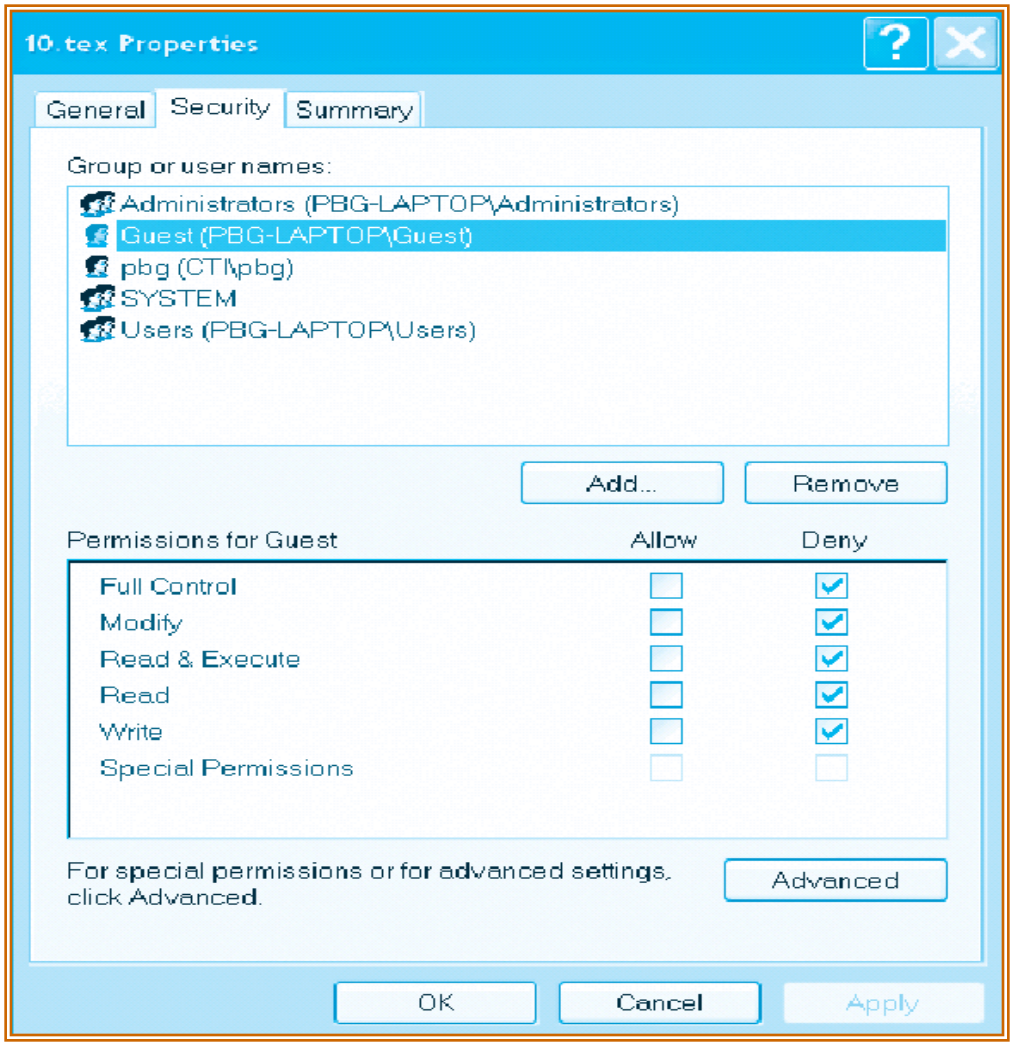
- Mode of access: Read, Write, eXecute
- For directory, X bit means ability to enter
- Three classes of users (9 bits total)

			RWX
a) **owner access**	7	⇒	1 1 1
b) **group access**	6	⇒	1 1 0
c) **public access**	1	⇒	0 0 1

- For a particular file (say *game*) or directory, define an appropriate access.



XP ACLs


10.tex Properties
General
Security
Summary
Group or user names:
Administrators (PBG-LAPTOP\Administrators)
Guest (PBG-LAPTOP\Guest)
pbg (CTI\pbg)
SYSTEM
Users (PBG-LAPTOP\Users)
Add...
Remove
Permissions for Guest
Allow
Deny
Full Control
Modify
Read & Execute
Read
Write
Special Permissions
For special permissions or for advanced settings, click Advanced.
Advanced
OK
Cancel
Apply

Linux vs Windows

- Linux: Each file just defines rights for a single owner, a single group, and the public
  - Pro: Compact enough to fit in a few words
  - Con: Not very expressive
- Windows*:* A per-file list that tells who can access that file in which ways
  - Pro: Highly expressive
  - Con: Harder to represent in a compact way

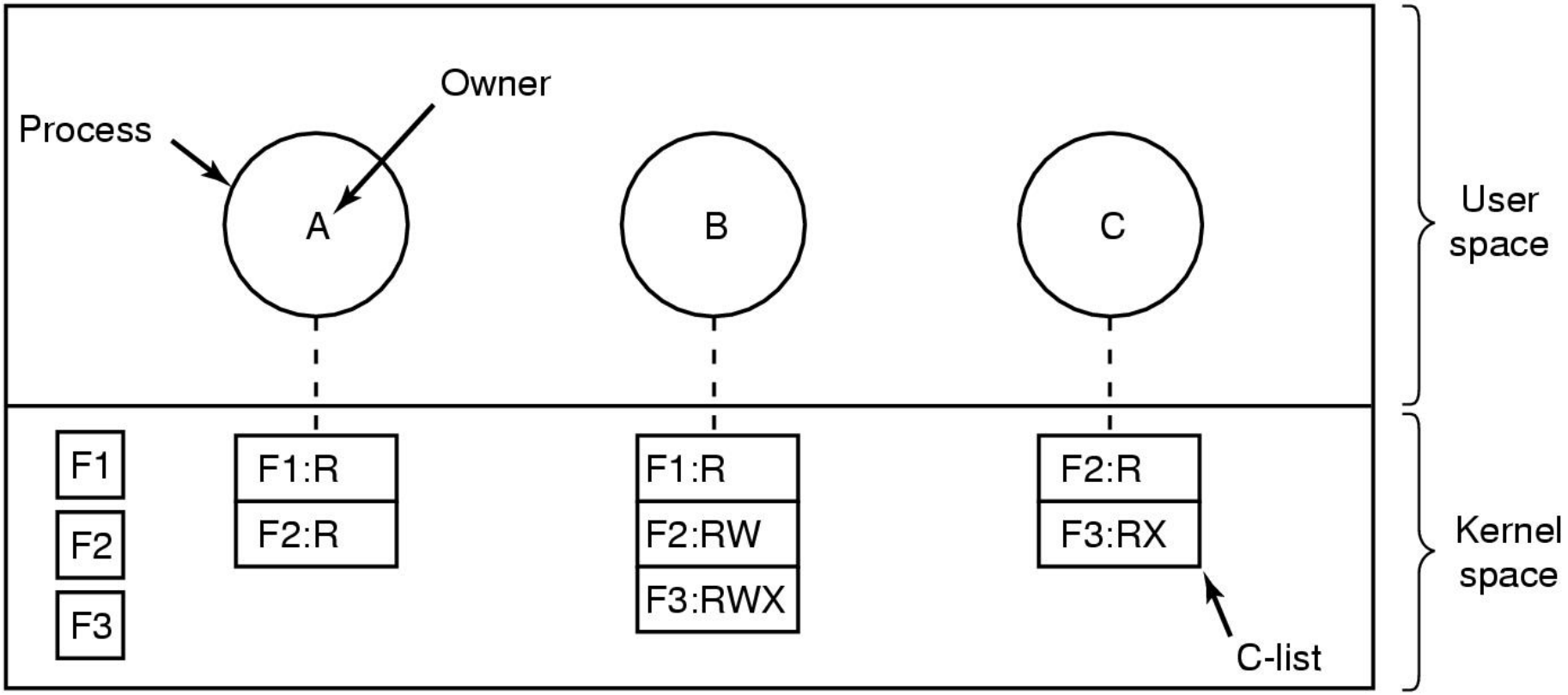
Capabilities

- To access an object, subject presents the capability
  - ‘capability’ word coined by Dennis and Van Horn in 1966
  - Capability is (x, r) list. x is object and r is set of rights
  - Capabilities are transferable
    - Access not based on who you are, but on what you have!
    - Capabilities can be attenuated (rights can be removed before transfer).
- Need to protect capabilities from being forged by others

Protecting Capabilities

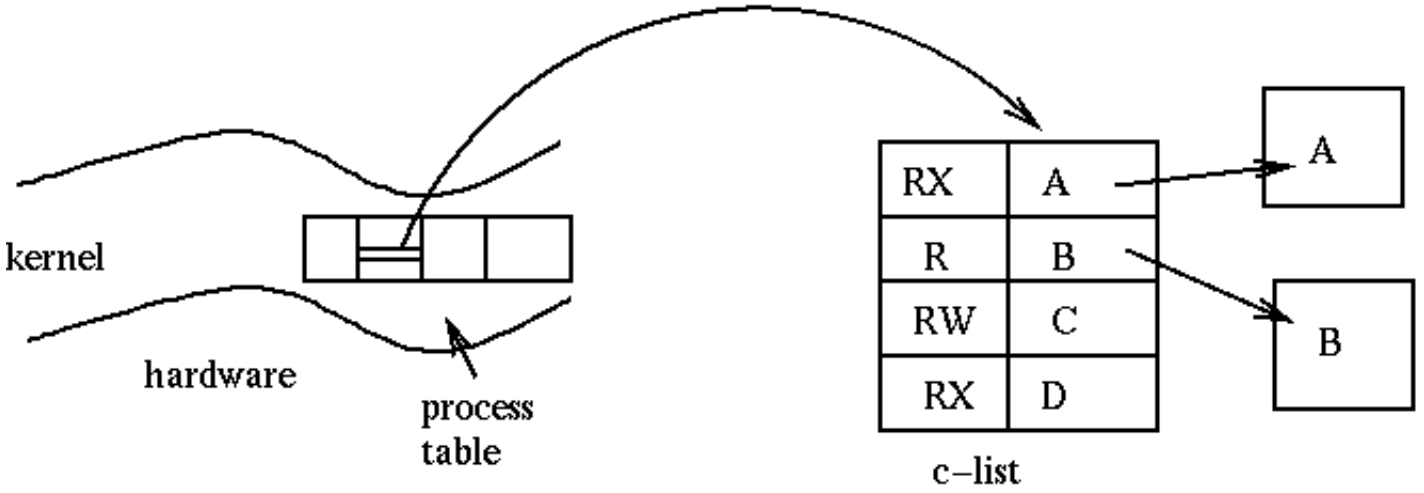
- Kernel Protection
  - Kernel keeps track of the capabilities that a user has (C-list)
- Language Protection
  - Programming language limits manipulation
  - Java references are an example of capabilities
    - C pointers are not. Why not?
- Tagged Architecture Protection (hardware)
  - Each memory word has extra bit indicating that it is a capability
  - These bits can only be modified in kernel mode
- Cryptographic Protection
  - For example, capability could be a large random number (hard to guess)
  - Alternatively, capabilities could be cryptographically signed

Kernel Protection



Kernel Protection cont'd

- Process access capabilities by offset into the C-list
- Indirection used to make capabilities unforgeable
- System calls to add/delete/modify/transfer capabilities



Comparing ACLs & Capabilities

- Capabilities support Principle of Least Privilege: only give programs the capabilities they need
- But capabilities are relatively hard to revoke once given out; with an ACL access can be retracted.
- Where does a user keep its capabilities?
- With capabilities, how do you find out who has access?
- Capabilities make it easier to share rights to objects.
- Capabilities do not require user authentication.
  - Good for simplicity and anonymity, bad for auditing

Authentication

- Establish the identity of user/machine by
  - Something you know (password, secret)
  - Something you have (credit card, smart card)
  - Something you are (retinal scan, fingerprint)
- In the case of an OS this is done during login
  - OS wants to know who the user is
- Passwords: secret known only to the subject
  - Simplest OS implementation keeps (login, password) pair
  - Authenticates user on login by checking the password
- Passwords should be made secure:
  - Length, case, digits, not from dictionary
  - Can be imposed by the OS! This has its own tradeoffs

Online passwords attacks

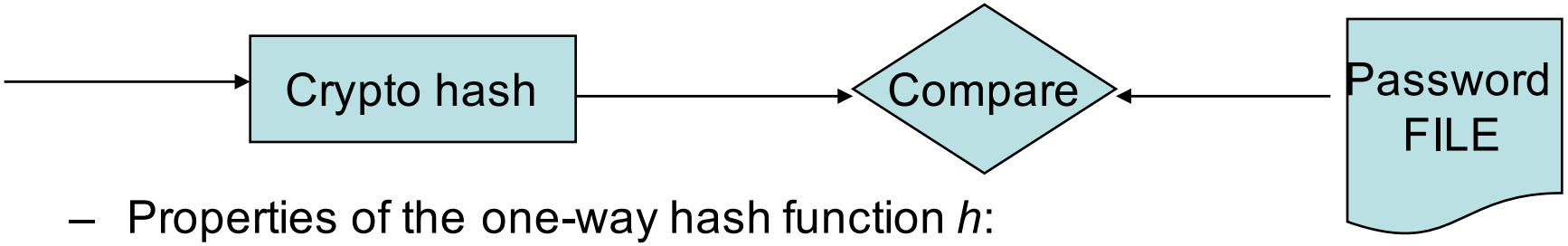
- Online attacks: system used to verify the guesses
  - How someone broke into LBL

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

  - Thwart these attacks:
    - limit the number of guesses
    - better passwords

Offline password attacks

- Depends on how passwords are stored
- Approach 1: store username/password in a file
  - Attacker only needs to read the password file
  - Security of system now depends on protection of this file!
- Approach 2: store username/encrypted password in file



  - Properties of the one-way hash function *h*:
    - *h* is not invertible: *h(m)* easy to compute, $h^{-1}(m)$ difficult
    - It is hard to find *m* and m‘ such that *h(m)* = *h(m’)*
  - Standard functions available, such as SHA, etc.
  - Ideally, hash function is slow (takes, say, a second to compute)
    - Unfortunately, most hash functions are very very fast…

More offline attacks

- Previous scheme can be attacked: Dictionary Attack
  - Attacker builds dictionary of likely passwords offline
  - At leisure, builds hash of all the entries
    - So-called “Rainbow Table”
  - Checks file to see if hash matches any entry in password file
  - There will be a match unless passwords are truly random
  - 20-30% of passwords in UNIX are variants of common words
    - Morris, Thompson 1979, Klein 1990, Kabay 1997
- Solutions:
  - Shadow files: move password file to /etc/shadow
    - This is accessible only to users with root permissions
  - Salt: store *(user name, salt, E(password+salt))*
    - Simple dictionary attack will not work. Search space is more.

Salting Example

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

- If the hacker guesses Dog, he has to try Dog0001, …

One time passwords

- Password lasts only once
  - User gets book with passwords
  - Each login uses next password in list

  - UNBREAKABLE..
    - but where do you keep that book?

Challenge Response Scheme

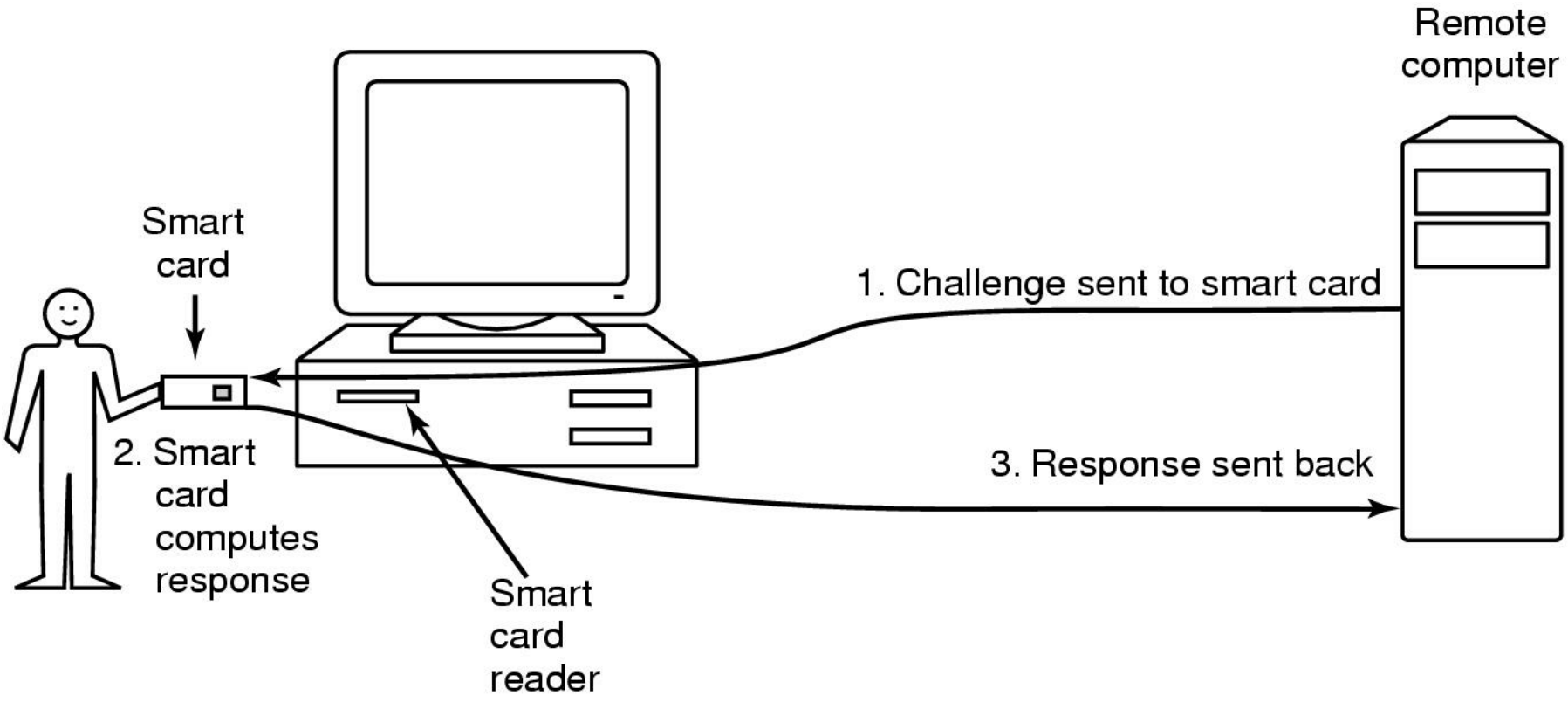
- New user provides server with list of question/answer pairs
  - Server asks one of them at random
  - Requires a long list of question/answer pairs
- Prove identity by computing a secret function
  - User picks an algorithm, e.g. x^2
  - Server picks a challenge, e.g. x=7
  - User sends back 49
  - Should be difficult to deduce function by looking at results
- In practice
  - The algorithm is fixed, e.g. one-way hash, but user selects a key
  - The server's challenge is combined with user's key to provide input to the function

Auth. Using Physical Objects

- Door keys have been around long
- Plastic card inserted into reader associated with comp
  - Also a password known to user, to protect against lost card
- Magnetic stripe cards: about 140 bytes info glued to card
  - Is read by terminal and sent to computer
  - Info contains encrypted user password (only bank knows key)
- Chip cards: have an integrated circuit
  - Stored value cards: have EEPROM memory but no CPU
    - Value on card can only be changed by CPU on another comp
  - Smart cards: 4 MHz 8-bit CPU, 16 KB ROM, 4 KB EEPROM, 512 bytes RAM, 9600 bps comm. channel

Smart Cards

- Better security than stored value cards
  - Card sends a small encrypted msg. to merchant, who can later use it to get money from the bank
  - Pros: no online connection to bank required
- Perform local computations, remember long passwords



Biometrics: something you are

- System has 2 components:
  - Enrollment: measure characteristics and store on comp
  - Identification: match with user supplied values
- What are good characteristics?
  - Finger length, voice, hair color, retinal pattern, voice, blood
- Pros: user carries around a good password
- Cons: difficult to change password, can be subverted

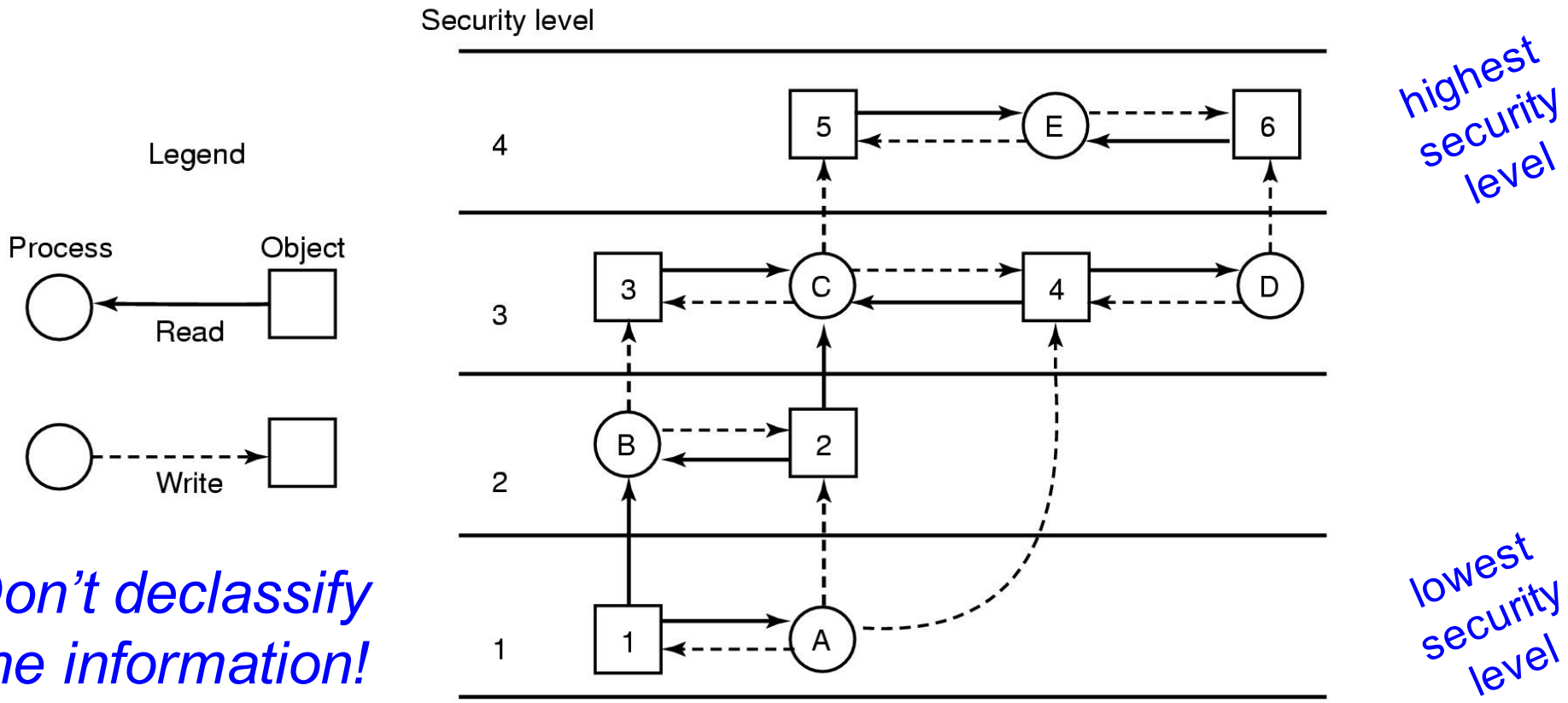
Discretionary vs Mandatory Access Control

- Discretionary Access Control (DAC)
  - What we've seen so far: *Who can access which objects*
- Mandatory Access Control (MAC)
  - *Who can do what, when, and with which objects?*
  - For example: after you've read a file, what can you do with it? Can you write it to another file? Can you print it?
  - Multi-level Security as an example of MAC
    - MLS is environment where there are various security levels
      - Eg. Classify info as unclassified, confidential, secret, top secret
      - General sees all documents, lieutenant can only see below confidential
  - Restrict information flow

Bell-La Padula Model

"no read up, no write down"

- Properties to satisfy for information flow
  - User at level 'k' can read objects at level 'j' , j <= k
  - User at level 'k' can write objects at level 'j', j >= k



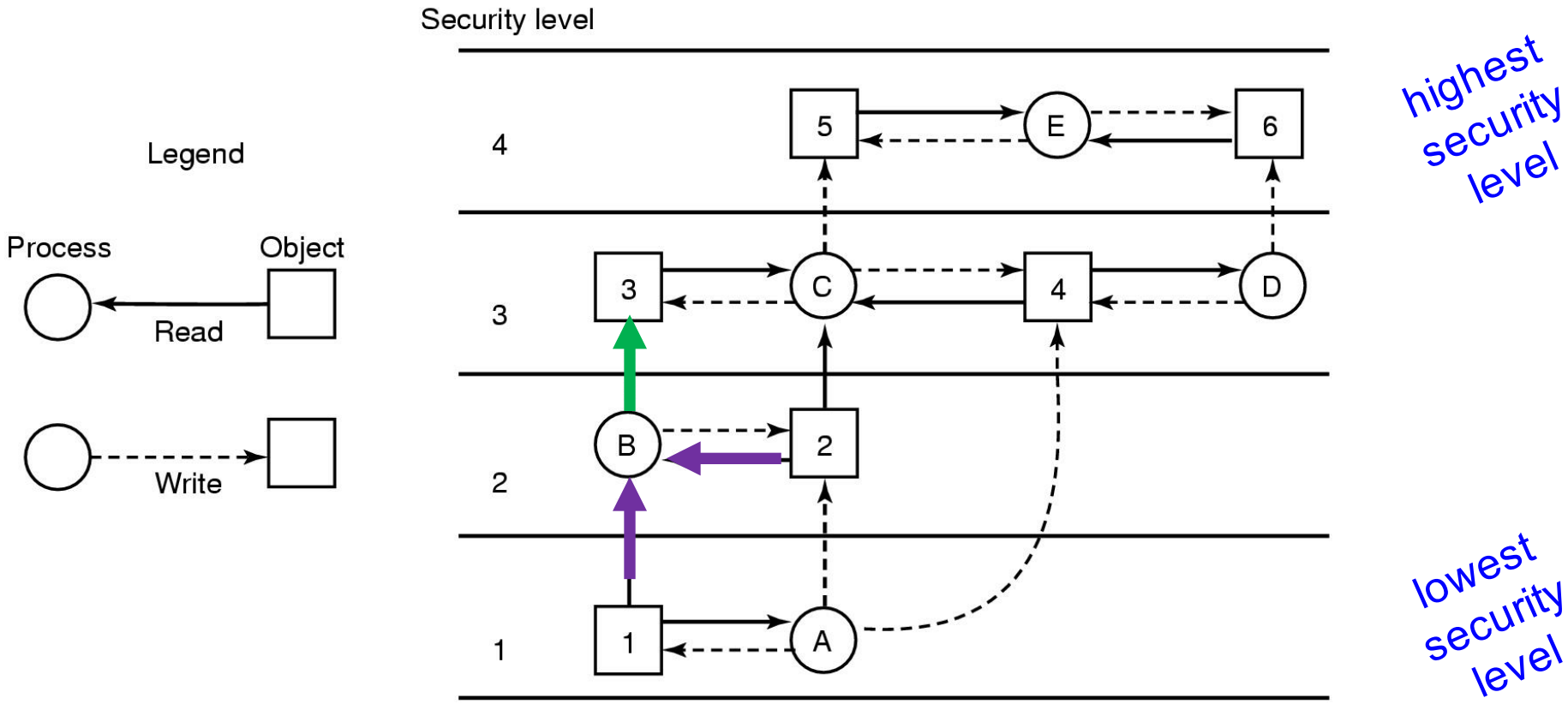
Don't declassify the information!

Bell-La Padula Model

Example:

Process B at Level 2 can read anything from an equal or lower security level

Process B at Level 2 can write to higher security levels



Biba Model

"No write up, no read down"

- A user at security level k can write only objects at level j, j <= k
  - higher level users don't trust this user
- A user at level k can read only objects at level j, j >= k
  - this user doesn't trust lower level users
- Bell-LaPadula + Biba → users only read and write at the same level
  - useless
- Better lattice-based models available that combine confidentiality and integrity levels together

Don't contaminate the information!

Trusted Systems

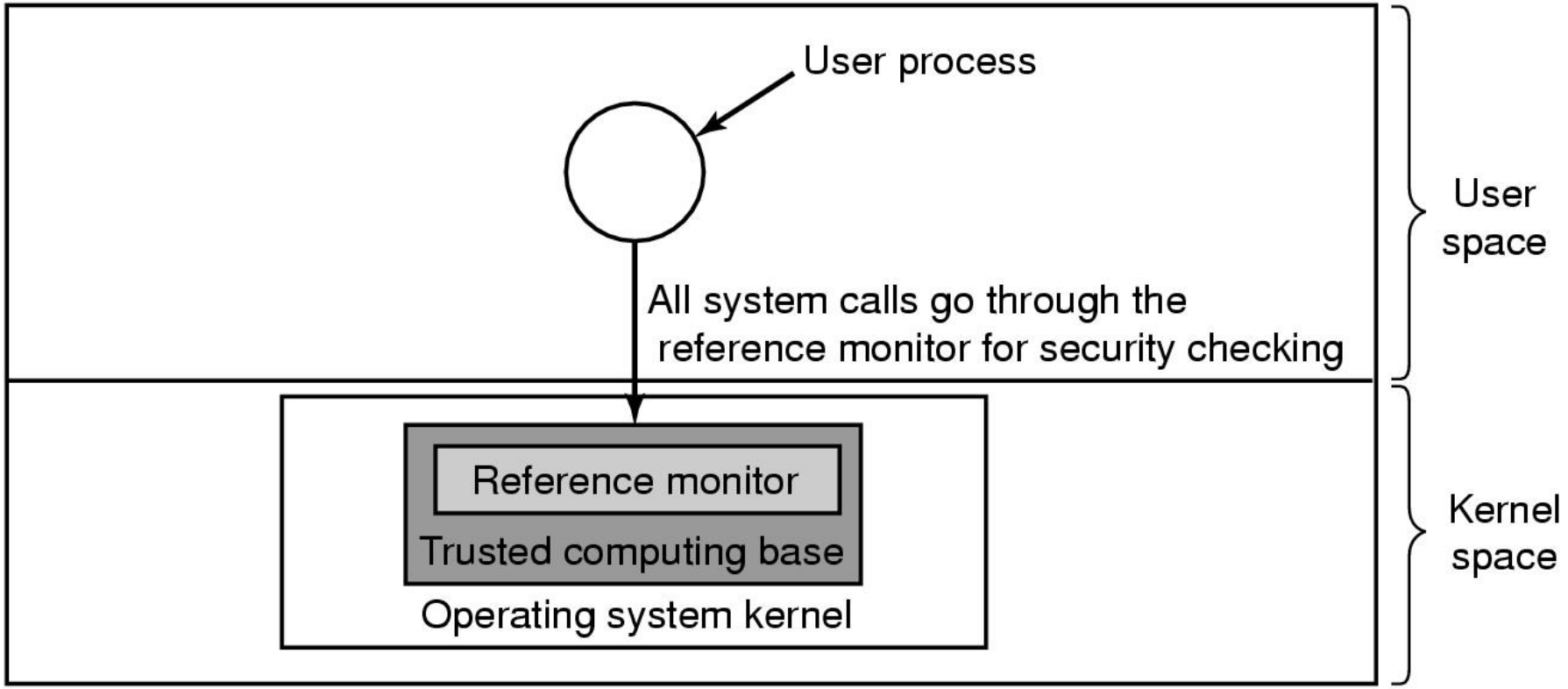
A trusted system has formally stated security requirements, and a proof of how they are met

Trusted Computing Base

- Heart of every trusted system has a small TCB
  - Hardware and software necessary for enforcing all security rules
  - Typically has:
    - most hardware,
    - Portion of OS kernel, and
    - most or all programs with superuser power
- Desirable features include:
  - Should be small
  - Should be separable and well-specified
  - Should be easy to audit independently

Reference Monitor

- Critical component of the TCB
  - All sensitive operations go through the reference monitor
  - Monitor decides if the operation should proceed



Covert Channels

- Do these ideas make our system completely secure?
  - No. Security leaks possible even in a system proved secure mathematically. Lampson 1973
- Model: 3 processes. The client, server and collaborator
  - Server and collaborator collude
  - Goal: design a system where it is impossible for server to leak to the collaborator info received from the client (Confinement)
- Solution: Access Matrix prevents server to write to a file that collaborator has read access; no IPC either
- BUT: Covert Channel => compute hard for 1, sleep for a 0
- Others: paging, file locking with ACKs, pass secret info even though there is censor

Steganography

- Pictures appear the same
- Picture on right has text of 5 Shakespeare plays
  - encrypted, inserted into low order bits of color values

Zebras

Hamlet, Macbeth, Julius Caesar
Merchant of Venice, King Lear

Orange Book

- Dept. of Defense Standards DoD 5200.28 in 1985
  - Known as Orange Book for the color of its cover
- Divides OSes into categories based on security property
  - **D** – Minimal security.
  - **C** – Provides discretionary protection through auditing. Divided into **C1** and **C2**. **C1** identifies cooperating users with the same level of protection. **C2** allows user-level access control.
  - **B** – All the properties of **C**, however each object may have unique sensitivity labels. Divided into **B1**, **B2**, and **B3**.
  - **A** – Uses formal design and verification techniques to ensure security.

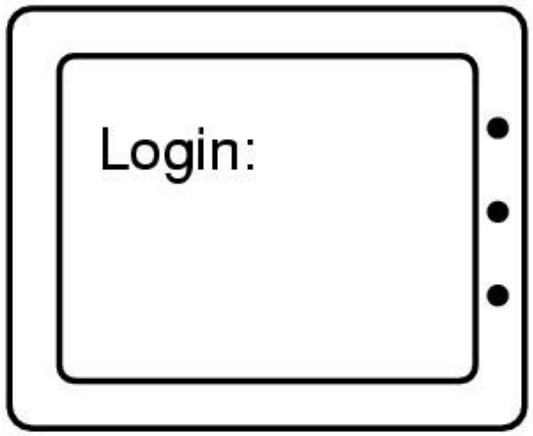
Security: Attacks

Trojan Horse

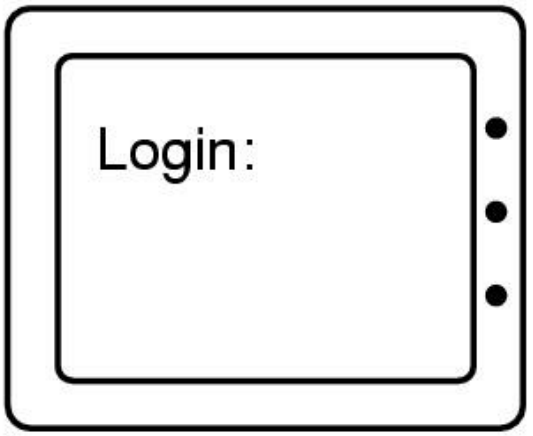
- Malicious program disguised as an innocent one
  - Could modify/delete user's file, send important info to cracker, etc
- The program has to get to the computer somehow
  - Cracker hides it as a new game, e-card, windows update site, etc.
- When run, Trojan Horse executes with user's privileges
- Examples:
  - Hide program in path directory as a common typo: *la* for *ls*
  - Malicious user puts malicious *ls* in directory, and attracts superuser
    - Malicious *ls* could make user the superuser

Login Spoofing

- Specialized case of Trojan Horse
  - Attacker displays a custom screen that user thinks belong to the system
  - User responds by typing in user name and password



(a)



(b)

Logic Bombs

- Piece of code, in the OS or app, which is dormant until a certain time has elapsed or event has occurred
  - Event could be missing employee record from payroll
- Could act as a Trojan Horse/virus once triggered
- Also called “slag code” or “time bomb”
- Recovery options for a firm include:
  - Calling the police
  - Rehiring the (disgruntled?) programmer

Trap Doors

- Code in system inserted by programmer to bypass normal check
- Ken Thompson "Reflections on Trusting Trust"
  - Hole in UNIX system utility; enforced by C compiler

```
while (TRUE) {
        printf("login: ");
        get_string(name);
        disable_echoing( );
        printf("password: ");
        get_string(password);
        enable_echoing( );
        v = check_validity(name, password);
        if (v) break;
}
execute_shell(name);
```

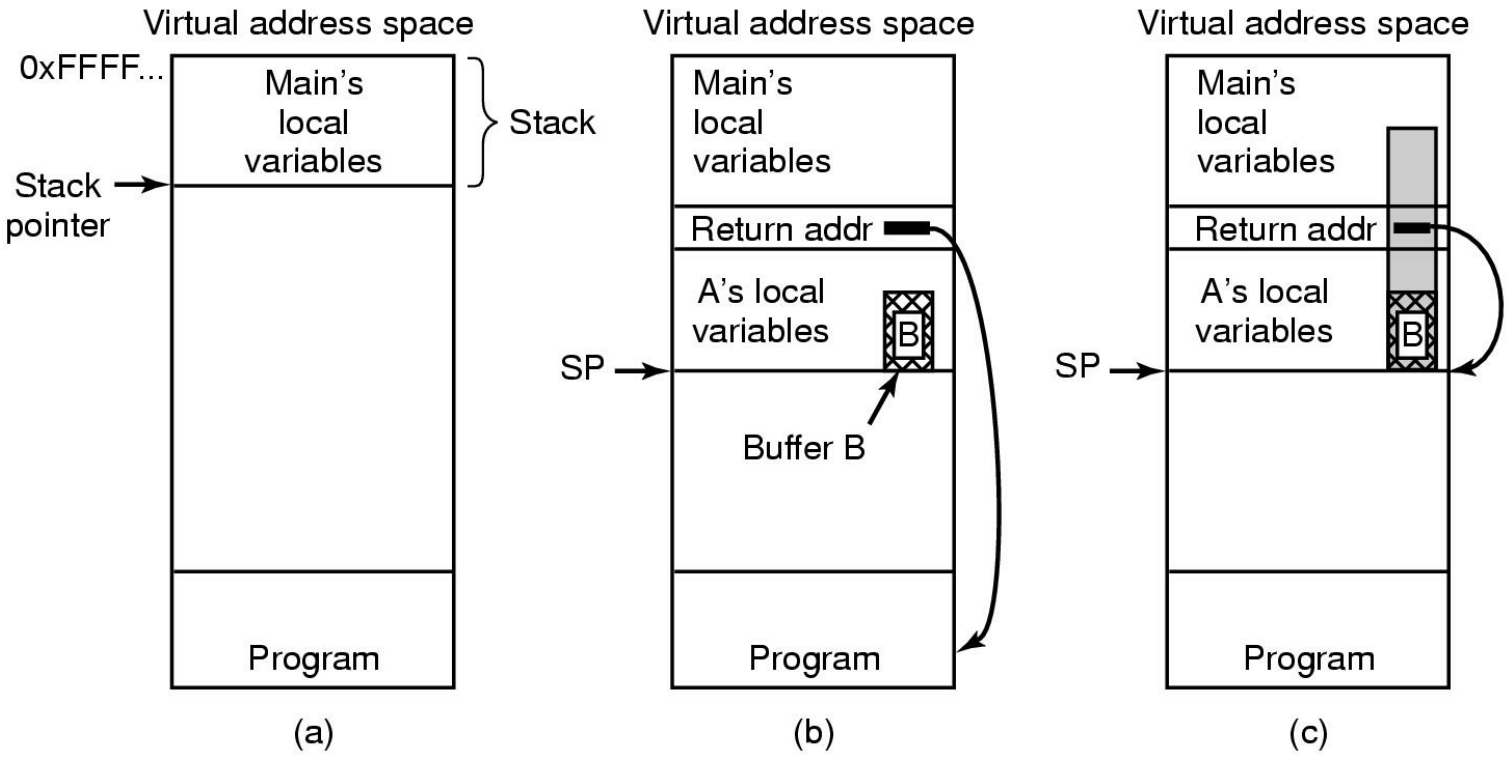
(a)

```
while (TRUE) {
        printf("login: ");
        get_string(name);
        disable_echoing( );
        printf("password: ");
        get_string(password);
        enable_echoing( );
        v = check_validity(name, password);
        if (v || strcmp(name, "zzzzz") == 0) break;
}
execute_shell(name);
```

(b)

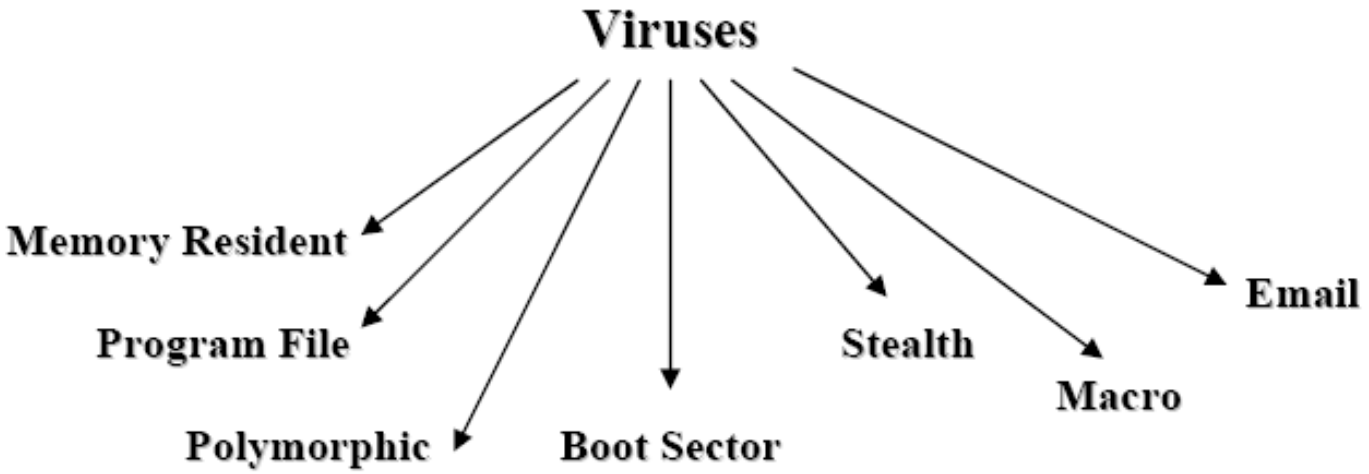
Buffer Overflow

- C compiler does no array bounds checking
  - Cracker can force his routine to run by violating array bounds



Viruses and Worms

- Virus is a program that reproduces itself by attaching its code to another program
  - They require human intervention to spread
- Worms actively replicate without a helper program
  - Is a subclass of virus, but does not require user intervention
  - Sasser and Blaster targeted machines with out of date software



November 1988: Internet Worm

Internet Worm attacks thousands of Internet hosts

Best Wikipedia quotes:

> “According to its creator, the Morris worm was not written to cause damage, but to gauge the size of the Internet. The worm was released from MIT to disguise the fact that the worm originally came from Cornell.”
>
> “The worm …determined whether to invade a new computer by asking whether there was already a copy running. But just doing this would have made it trivially easy to kill: everyone could run a process that would always answer "yes”. To compensate for this possibility, Morris directed the worm to copy itself even if the response is "yes" 1 out of 7 times. This level of replication proved excessive, and the worm spread rapidly, infecting some computers multiple times. Morris remarked, when he heard of the mistake, that he "should have tried it on a simulator first”.”

Computer Virus TV News Report 1988

Denial of Service

- Client sends a legitimate-looking request for service to a service provider
- Service provider commits the necessary resources to provide the service
  - Ports, buffer space, bandwidth
- The resources are wasted, legitimate users get diminished service
  - Usually launched from many computers controlled by attackers
- Possible whenever the cost to ask for service is far cheaper than the cost of providing it
  - Challenge-response mechanism, selective packet tagging

Other Network Attacks

- Protocol attacks:
  - E.g. IEEE 802.11 WEP
- Brute force attacks
- Use Network Firewalls to reduce security risk