# Chapter 9

# Security

1. The security environment
2. Basics of cryptography
3. User authentication
4. Attacks from inside the system
5. Attacks from outside the system
6. Protection mechanisms
7. Trusted systems

# 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

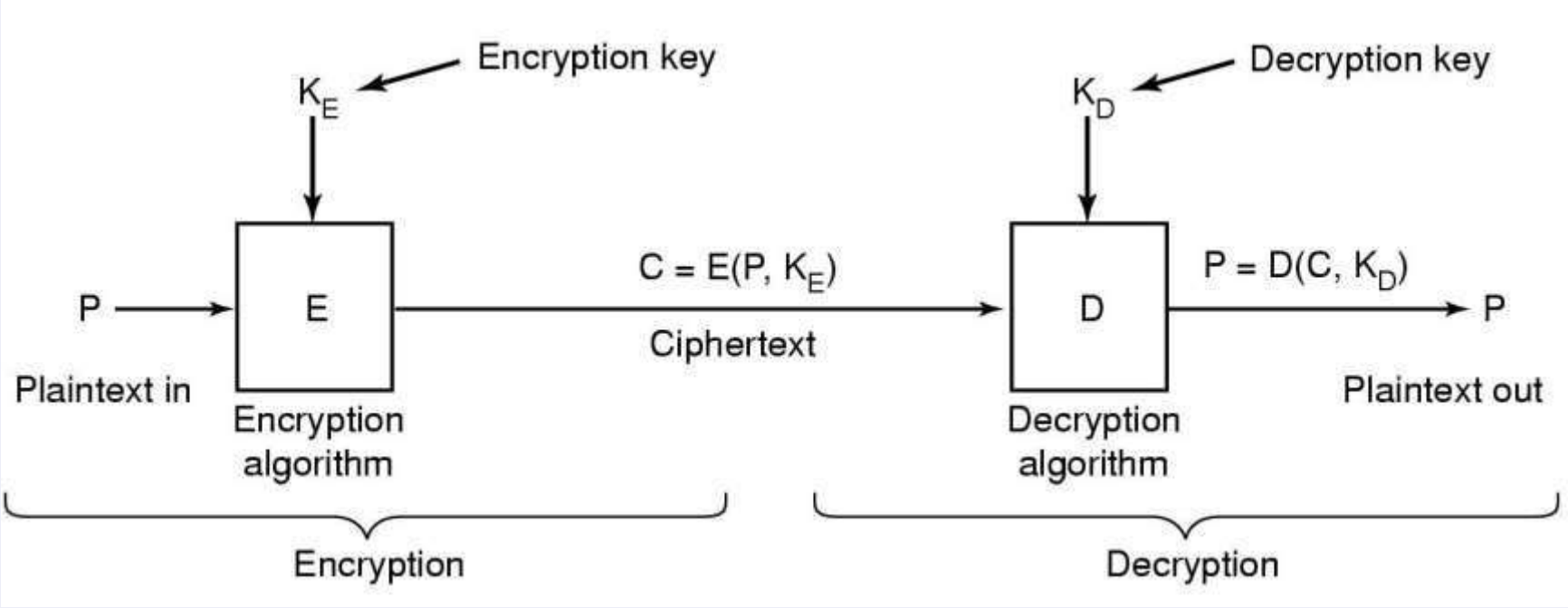
1. Casual prying by nontechnical users
2. Snooping by insiders
3. Determined attempt to make money
4. Commercial or military espionage

# 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

# 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

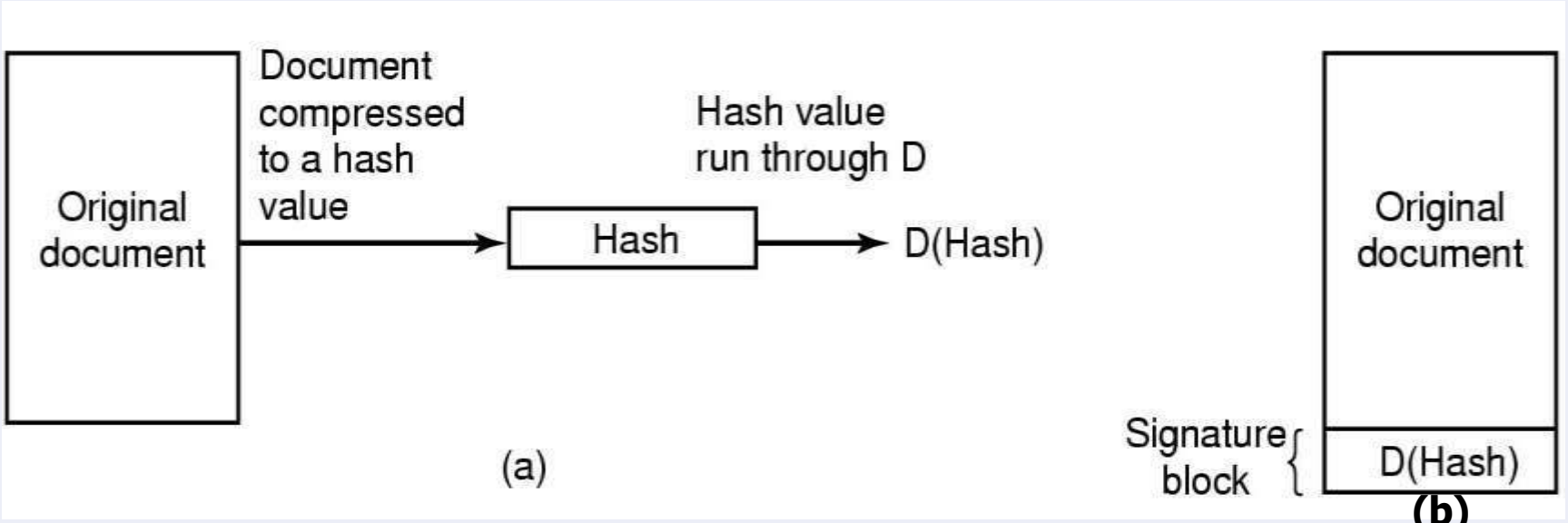
# Public-Key Cryptography

- 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

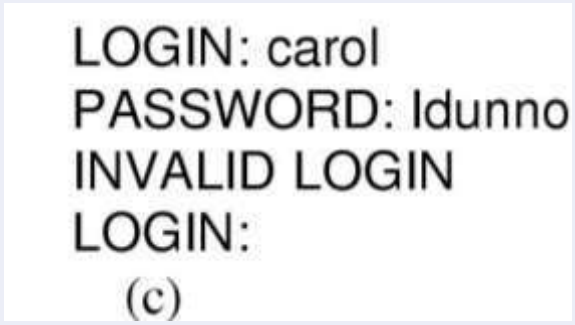
Basic Principles. Authentication must identify:

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




(a) A successful login

(b) Login rejected after name entered

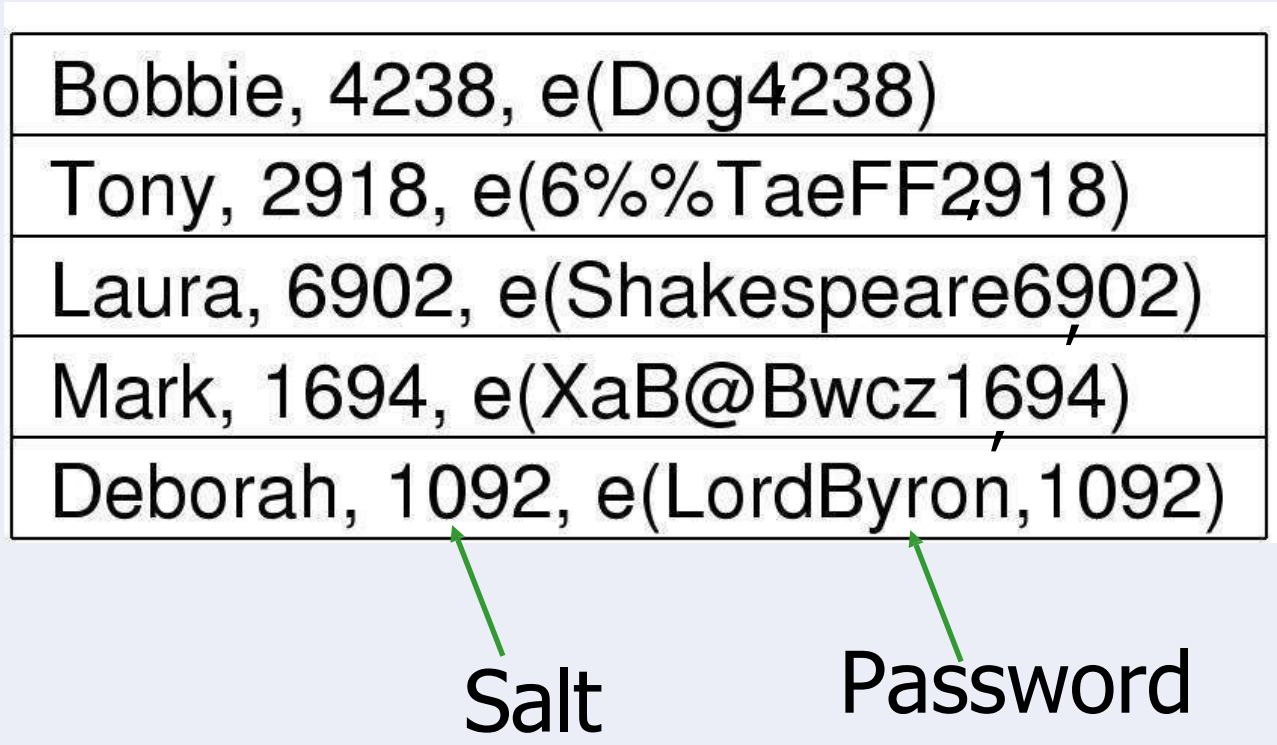
(c) Login rejected after name and password typed

# 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: uucp
PASSWORD: uucp
WELCOME TO THE ELXSI COMPUTER AT LBL
```

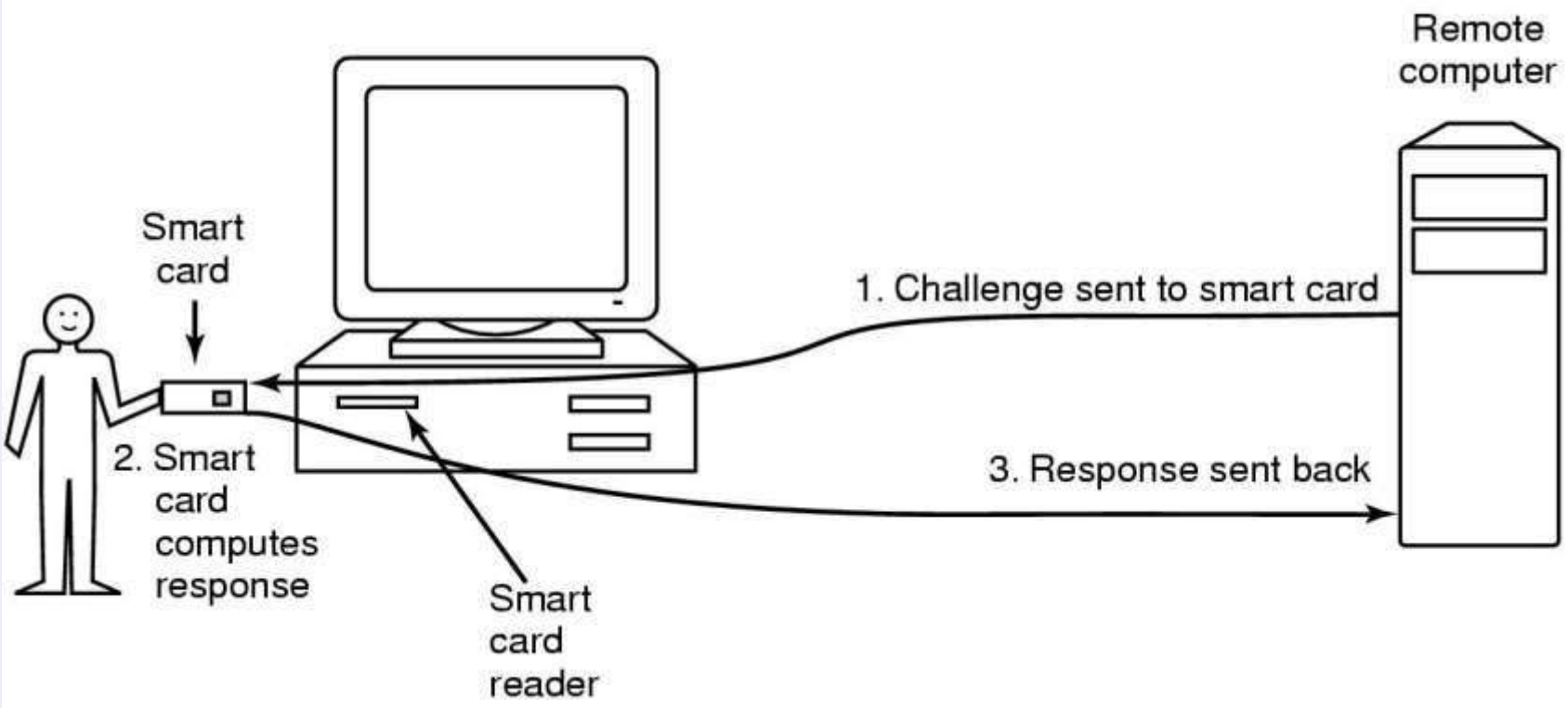
- How a cracker broke into LBL
  - a U.S. Dept. of Energy research lab

# Authentication Using Passwords



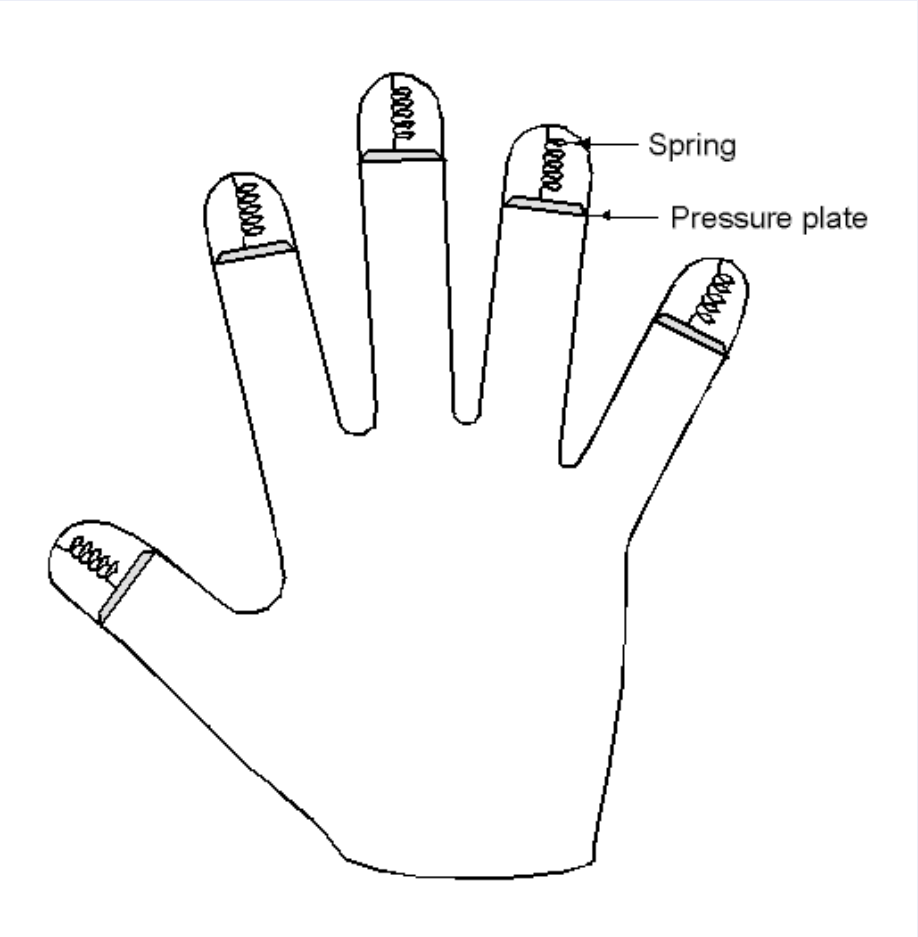

The use of salt to defeat  putation of encrypted passwords

# 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

# Operating System Security
## Trojan Horses

- Free program made available to unsuspecting user
  - Actually contains code to do harm

- Place altered version of utility program on victim's computer
  - trick user into running that program

# Login Spoofing




(a) Correct login screen

(b) Phony login screen

# Logic Bombs

- Company programmer writes program
  - potential to do harm
  - OK as long as he/she enters password daily
  - ff programmer fired, no password and bomb explodes

# Trap Doors

```
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)

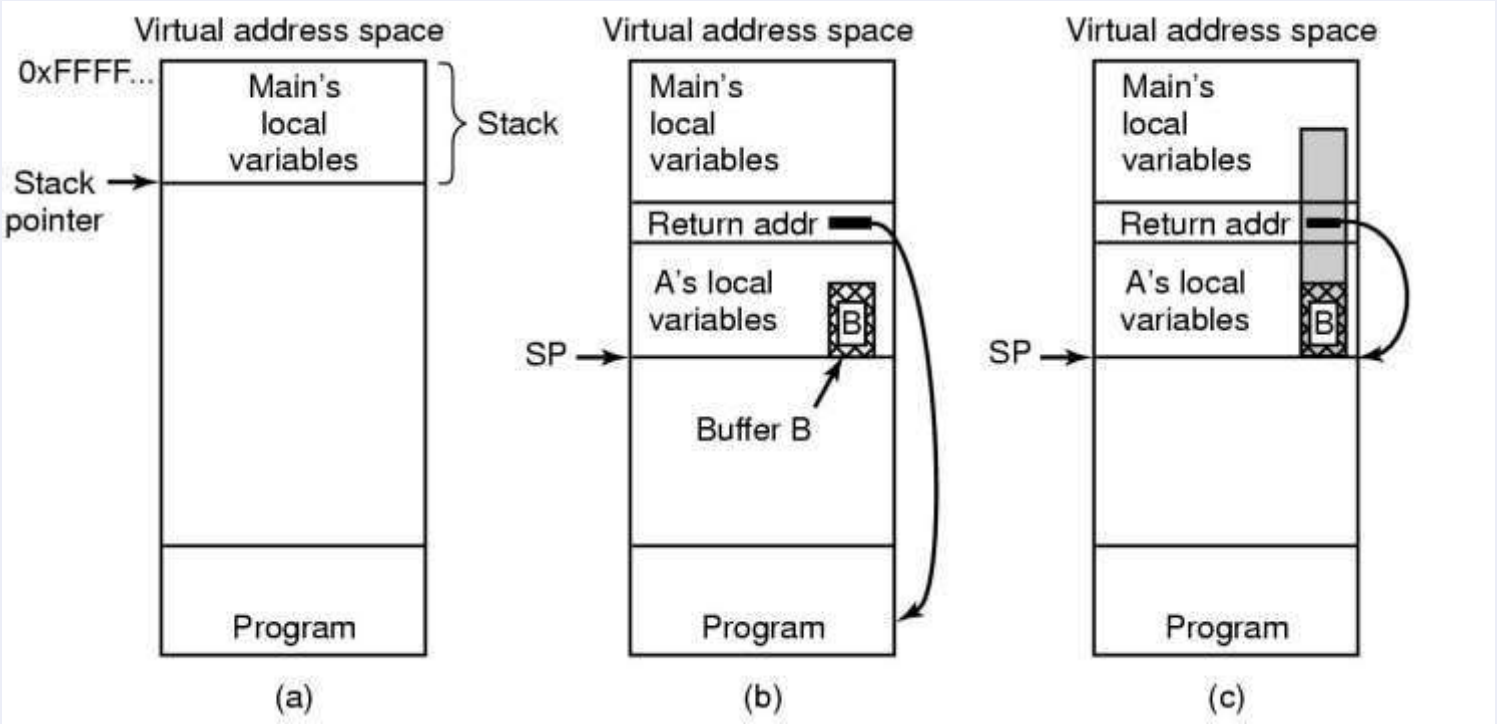
```
while (TRUE) {
        printf("login: ");
        get_string(name);
        disable_echoing( );
        printf("password: ");
        get_string(password);
        enable_echoing( );
        v = check_validity(name, password);
        if (v || strcmp(name, "zzzzz") == 0) break;
}
execute_shell(name);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

(b)

(a) Normal code.

(b) Code with a trapdoor inserted

# Buffer Overflow



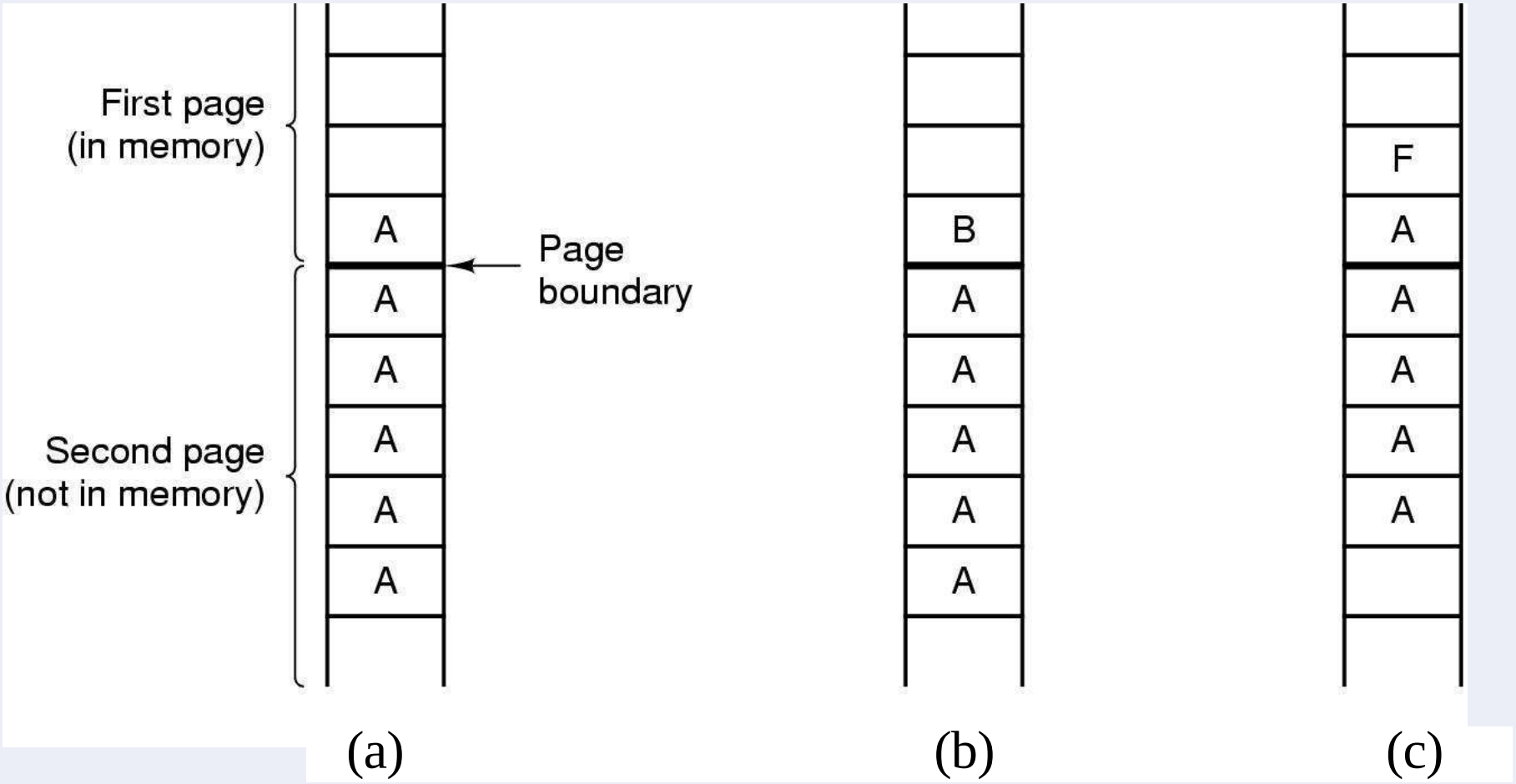

- (a) Situation when main program is running
- (b) After program *A* called
- (c) Buffer overflow shown in gray

# Generic Security Attacks

Typical attacks

- Request memory, disk space, tapes and just read
- Try illegal system calls
- Start a login and hit DEL, RUBOUT, or BREAK
- Try modifying complex OS structures
- Try to do specified DO NOTs
- Convince a system programmer to add a trap door
- Beg admin's sec’y to help a poor user who forgot password

# Famous Security Flaws




## The TENEX – password problem

# Design Principles for Security

1. System design should be public
2. Default should be n access
3. Check for current authority
4. Give each process least privilege possible
5. Protection mechanism should be
   - simple
   - uniform
   - in lowest layers of system
6. Scheme should be psychologically acceptable

And … keep it simple

# Network Security

- External threat
  - code transmitted to target machine
  - code executed there, doing damage
- Goals of virus writer
  - quickly spreading virus
  - difficult to detect
  - hard to get rid of
- Virus = program can reproduce itself
  - attach its code to another program
  - additionally, do harm

# Virus Damage Scenarios

- Blackmail
- Denial of service as long as virus runs
- Permanently damage hardware
- Target a competitor's computer
  - do harm
  - espionage
- Intra-corporate dirty tricks
  - sabotage another corporate officer's files

# How Viruses Work (1)

- Virus written in assembly language
- Inserted into another program
  - use tool called a “dropper”
- Virus dormant until program executed
  - then infects other programs
  - eventually executes its “payload”

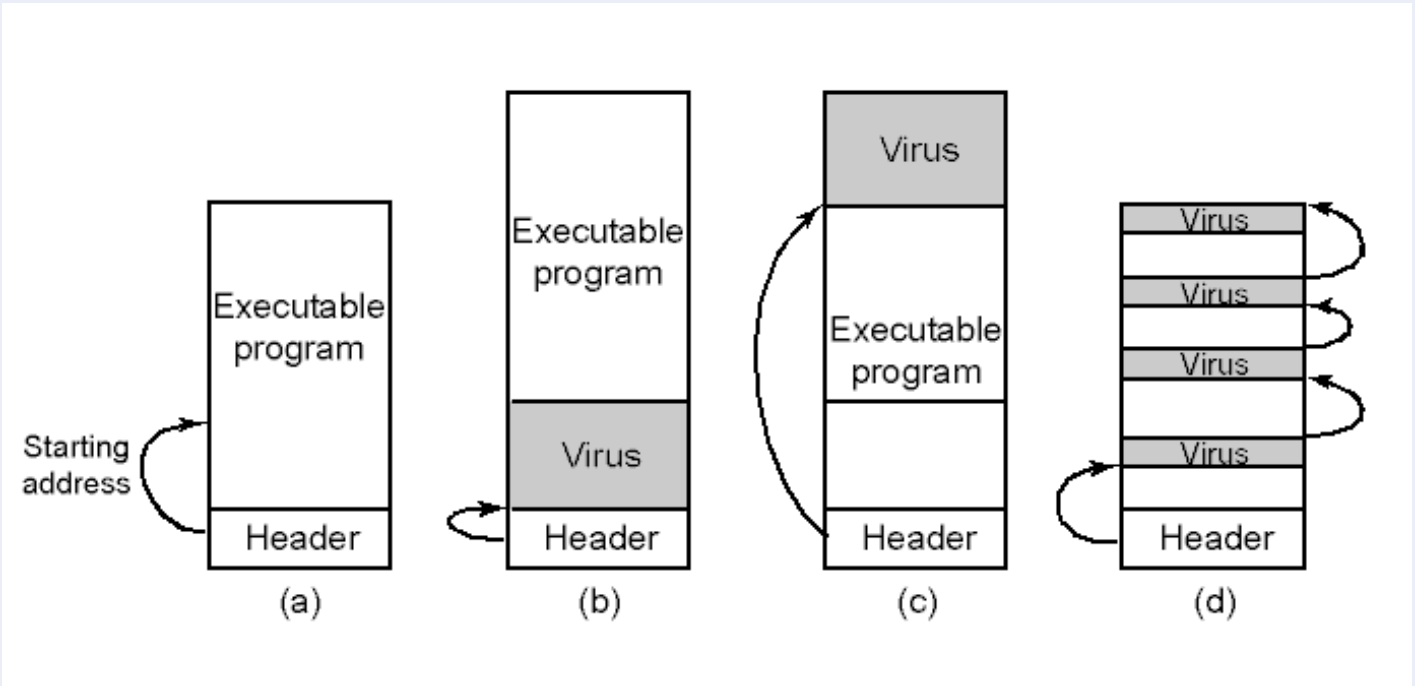
# How Viruses Work (2)

Recursive procedure that finds executable files on a UNIX system

Virus could infect them all

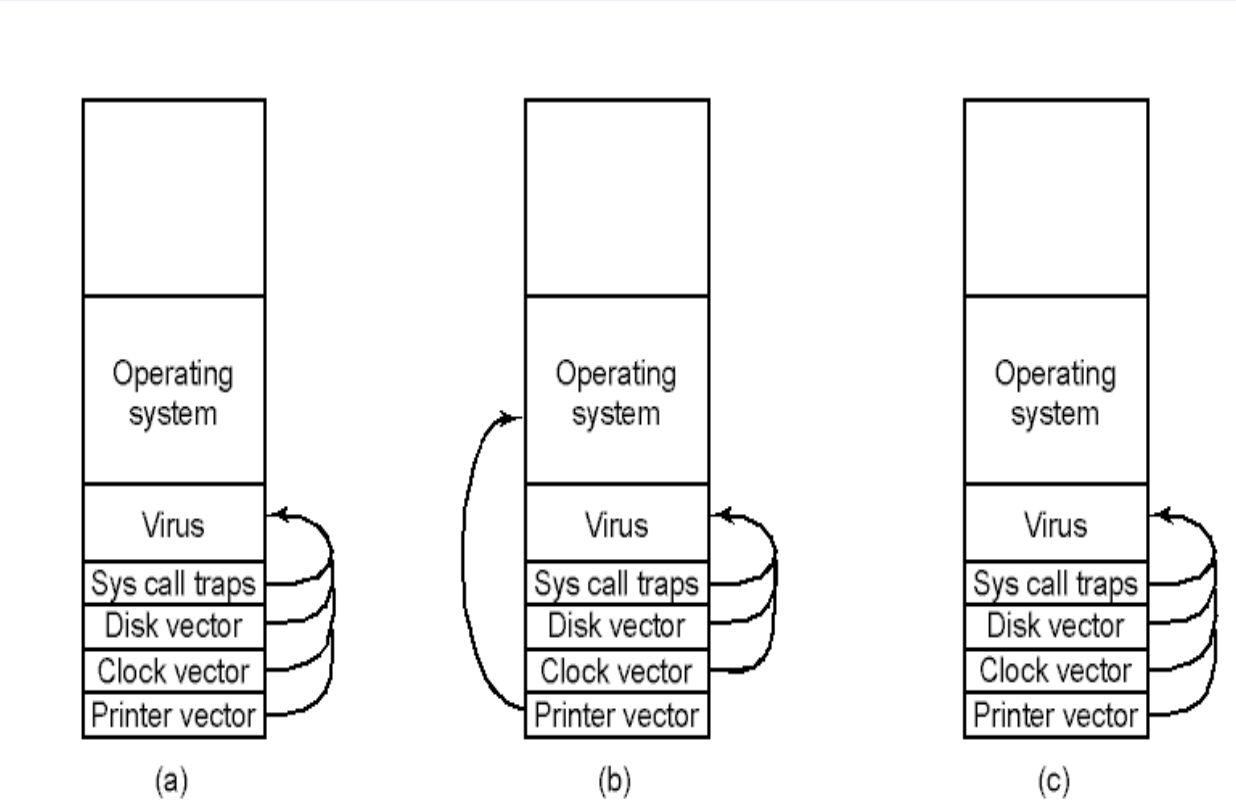
```
#include <sys/types.h>                  /* standard POSIX headers */
#include <sys/stat.h>
#include <dirent.h>
#include <fcntl.h>
#include <unistd.h>
struct stat sbuf;                       /* for lstat call to see if file is sym link */

search(char *dir_name)
{                                       /* recursively search for executables */
    DIR *dirp;                          /* pointer to an open directory stream */
    struct dirent *dp;                  /* pointer to a directory entry */

    dirp = opendir(dir_name);           /* open this directory */
    if (dirp == NULL) return;           /* dir could not be opened; forget it */
    while (TRUE) {
        dp = readdir(dirp);             /* read next directory entry */
        if (dp == NULL) {               /* NULL means we are done */
        chdir ("..");                   /* go back to parent directory */
        break;                          /* exit loop */
    }
    if (dp->d_name[0] == '.') continue; /* skip the . and .. directories */
    lstat(dp->d_name, &sbuf);           /* is entry a symbolic link? */
    if (S_ISLNK(sbuf.st_mode)) continue; /* skip symbolic links */
    if (chdir(dp->d_name) == 0) {       /* if chdir succeeds, it must be a dir */
        search(".");                    /* yes, enter and search it */
    } else {                                /* no (file), infect it */
        if (access(dp->d_name,X_OK) == 0) /* if executable, infect it */
            infect(dp->d_name);
    }
    closedir(dirp);                     /* dir processed; close and return */
}
```

# How Viruses Work (3)




- An executable program
- With a virus at the front
- With the virus at the end
- With a virus spread over free space within program

# How Viruses Work (4)




- After virus has captured interrupt, trap vectors
- After OS has retaken printer interrupt vector
- After virus has noticed loss of printer interrupt vector and recaptured it

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：https://d.book118.com/898114104002006025