# CS323 Operating Systems Security

Yuanyuan Zhou

Lecture 38

5/2/2003

## Reminder

- Conflict exam signup
- Graduate students 1-unit project
- Senior thesis, under-graduate research, independent study



## Content of this lecture

- Security Flaws in Operating Systems
- Attacks on O/S Security
- Viruses and Worms



## Discussion

- How do you break into the machine in DCL 1320?

## Security Flaws in Operating Systems

- Authentication
  - E.g., a dummy program, pretending to be the signon program, asking for the user's password and then storing it.
- Line disconnect
  - When a line is disconnected with a user logged in over it, the system must either log the user out, or at least put the line in a state in which the user must re-authenticate his identity after reconnecting before proceeding with the session.
- Operator carelessness
  - E.g., tricking the operator into mounting a counterfeit operating system disk..
- Residue
  - Interesting information often turns up in wastebaskets; use paper shredders! Information is often left in central memory from a previous user, possibly a system routine; variables that contain sensitive information should be overwritten before they are deallocated!
- Shielding
  - One can inductively ``tap'' a cable, phone line, or in fact any wire over which information passes, without making physical connection to it. Electrical shielding can protect against this.
- Passwords
  - Password guessing, etc.

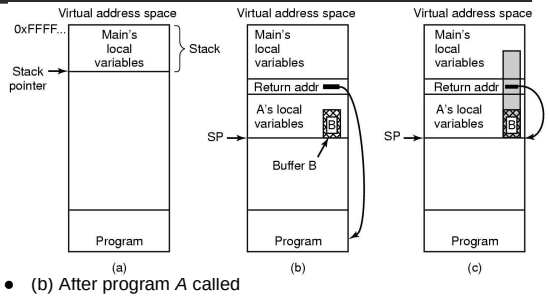


## Bugs

- Legality checking
  - The system may fail to check parameters supplied by the user in calling system procedures.
- Implicit trust
  - One routine assumes that parameters passed to or shared with another routine are correct; each routine ought to check parameters supplied by another.
- Prohibitions
  - Features which are advertised not to work, or not to work correctly, but which are still executable, with ``interesting'' results.
- Implementation
  - An improper implementation of a well thought out design for a security mechanism.



## Buffer Overflow




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



## Design Weaknesses

- Encryption
  - Lack of encryption of the password file or other security codewords.
- Implied sharing
  - The system may deposit critical information in a user's space.
- Parameter passing
  - by reference or by value. Passing by reference offers the possibility that the user may present valid arguments for checking, and then modify them just before their use by the system.
- Inter-process communication
  - Use of a send / receive mechanism to test possibilities, e.g., to test for a correct password.

## Classic Problems

- Privilege
  - Systems may give programs and/or users more privilege than they need. *Principle of Least Privilege*
- Trojan Horse
  - Executing a program written by someone else could use the executor's privileges to send information to the penetrator, or to damage the executor's files, or to open a hole into which the penetrator may later enter.



## Attacks from Inside of the System

- **Trojan Horse**
  - seemingly innocent program contains code to perform an unexpected and undesirable function.
- Examples
  - Modifying, deleting or encrypting the user file; copying them into a place where cracker can retrieve them later, or even sending them to the cracker via email of FTP.
- One approach to do this is to place the program as a free, exciting new game, MP3 viewer, or something that attracts attention.
- The Trojan horse approach does not require the user to break into the computer.



## Other Inside Attacks

- **Login Spoofing**
  - attacker writes a false login program that displays on the screen login prompt. This program asks for name, password, user types in login name and password. The false information is written to a file and the phony login program sends a signal to kill the shell. This action logs the attacker out and triggers the real login program. The user assumes that he.she wrote the wrong password and repeats the steps.
- **Logic Bombs**
  - build in bad behavior (e.g., erase a disk) into operating system if certain action is not taken. For example, as long the programmer feeds in a password every day, the behavior is not visible. When a programmer is fired, the password is not given and the bad behavior is triggered.
- **Trap Doors**
  - code is inserted into the system by the system programmer to bypass some normal check. For example, a login program could be written which allows a user to login independent of what password he/she types. The trap-door bypasses the whole authentication process.
-



## 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
  - if 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



## Other Inside Attacks on O/S Security

- Asynchronism
  - One process modifies the arguments another process has passed to an operating system procedure after they have been tested for validity but before they have been used.
- Browsing
  - A user searches the system simply trying things, looking for privileged information.
- Between lines
  - A user taps into a line being used by an inactive but logged-in terminal.
- Clandestine code
  - A patch is made to the system which, instead of or in addition to doing what it is supposed to do, provides a hole that a penetrator can use later.



## Other Inside Attacks on O/S Security

- Denial of access
  - A user writes a program to deliberately crash the system, send it into an infinite loop, or otherwise disrupt use of it by legitimate users.
- Disconnected lines
  - A penetrator tries to find an incoming line that disconnected while someone was logged in using it.
- Masquerade
  - A penetrator assumes another user's identity, typically by stealing his password.
- NAK attack
  - The system may be vulnerable when a running process is interrupted (NAK, Negative Acknowledgment, U, sometimes used as the interrupt key).
  - A penetrator might be able to catch the system in an unprotected state during interruption, and thus seize control.

## 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

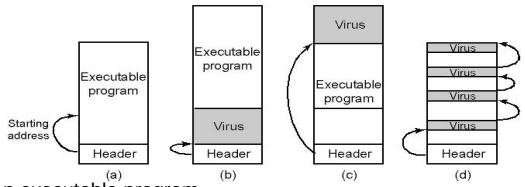
Virus could infect them all

```
#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 */
}
```

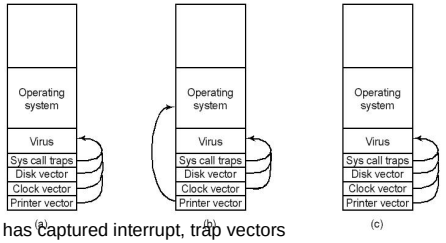
## How Viruses Work (3)




- 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



## How Viruses Spread

- Virus placed where likely to be copied
- When copied
  - infects programs on hard drive, floppy
  - may try to spread over LAN
- Attach to innocent looking email
  - when it runs, use mailing list to replicate



## Antivirus and Anti-Antivirus Techniques

- Integrity checkers
- Behavioral checkers
- Virus avoidance
  - good OS
  - install only shrink-wrapped software
  - use antivirus software
  - do not click on attachments to email
  - frequent backups
- Recovery from virus attack
  - halt computer, reboot from safe disk, run antivirus

## The Internet Worm

- Free-standing program designed to travel between systems for some particular purpose.
- Consisted of two programs
  - bootstrap to upload worm
  - the worm itself
- Worm first hid its existence
- Next replicated itself on new machines