

Outline for Today's Lecture

Administrative:

Question – take-home final possibility

pro: not time-constrained, open book

con: risk of cheating too great (5 min. in google and you can find solutions all over the place; I have had 1st hand experience with plagiarism off the web within the last year)

Question – in class, open book

pro: doesn't require memorization

con: constrains my choice of problems somewhat (but in ways I can live with), can waste time during the exam

Outline for Today's Lecture

More administrative:

- More midterms to give back (see me after class)
- Problem sets (circulating)
- No discussion sessions this week.
- Demos for Nachos assignment 6 starting Friday
- Office hours next week (Tuesday afternoon)

Outline for Today's Lecture

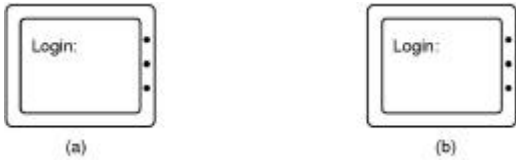
Last time: Authentication of who you are;
Access control for what you are allowed
to access in the system.

Objective: Security risks

Trojan Horses: Inside Jobs

- Free program made available to unsuspecting user
 - Contains code to do harm
 - Example of tricking user, himself or herself, into running that program
- Place altered version of utility program on victim's computer, say, in some bin directory that might be in their PATHs
- Lay a trap for sysadmin to gain root privilege
 - Install local version of ls; then do something that causes sysadmin to
 - % cd yourHomeDir
 - % ls -l

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; his/her name stays on the paycheck database; etc.
- if programmer fired, no password and bomb explodes

Trap Doors

```

while (TRUE) {
    printf("login: ");
    get_string(name);
    disable_echong();
    printf("password: ");
    get_string(password);
    enable_echong();
    v = check_validity(name, password);
    if (v) break;
}
execute_shell(name);
(a)

```

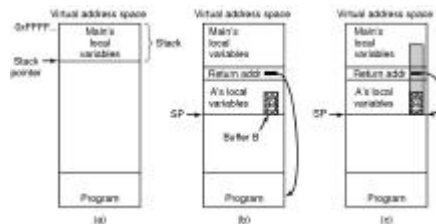
```

while (TRUE) {
    printf("login: ");
    get_string(name);
    disable_echong();
    printf("password: ");
    get_string(password);
    enable_echong();
    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 procedure A is called
(c) Fixed sized buffer B - overflow shown in gray
User supplies long enough string to overwrite return address

Generic Security Attacks

Typical attacks

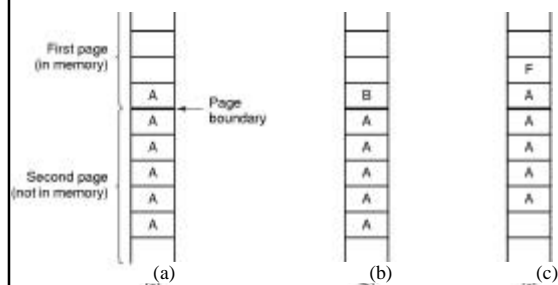
- Request "free" memory, disk space, tapes and just read what was left there (not zero filled on dealloc)
- Try illegal system calls – if the system gets confused enough, you may be in.
- Start a login and hit DEL, RUBOUT, or BREAK to possibly kill password checking
- Try modifying complex OS structures kept in user space (if any)
- 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

UNIX lpr utility, option to remove file after printing; print and remove password file

Link file *core* in working directory to the password file. Force a core dump of a SETUID program, writes on the core file, overwriting password file

Famous Security Flaws



The TENEX – password problem

- Page faults could be monitored by user – user func. called
- One char at a time password checking – either illegal or fault

Design Principles for Security

1. System design should be public – open source
2. Default should be no access
3. Check for current authority – e.g. not just at "open"
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

From the Outside: 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 by attaching 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

- Virus usually 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"
 - possibly waits for significant date
- Types: companion, executable program, memory, boot sector, device driver, macro, source code

How Viruses Work

- Bait and switch – companion viruses
- MS-DOS example:
 - run prog
 - system tries prog.com then prog.exe
 - Release a prog.com for a popular prog.exe

How Viruses Work

Recursive procedure that finds executable files on a UNIX system

Virus could infect them all

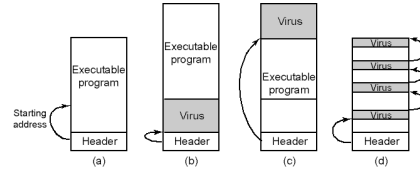
```

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

searchdir (dir_name)
{
    DIR *dirp;
    struct dirent *dp;

    dirp = opendir (dir_name);
    if (dirp == NULL) return;
    while (TRUE)
    {
        dp = readdir (dirp);
        if (dp == NULL)
            break;
        if (dp->d_name[0] == '.') continue;
        if (IS_DIRECTORY (dp->d_name, &dp->d_stat)) continue;
        if (IS_SYMLINK (dp->d_name, &dp->d_stat)) continue;
        if (dp->d_name[0] == '.') continue;
        if (access (dp->d_name, X_OK) == 0)
            infect (dp->d_name);
    }
    closedir (dirp);
}
    
```

How Viruses Work



- An executable program
- with a parasitic virus at the front
- at the end
- spread over free space within program (cavity virus)

How Viruses Work

Boot sector viruses

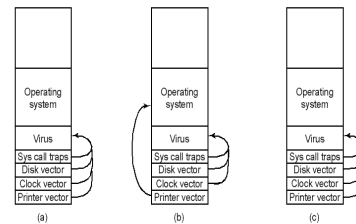
1st hide the real boot sector

When booted, copies virus into memory, making it a memory resident virus

Then boots the OS

Device driver infected with virus, loads it at boot time.

How Viruses Work



- After virus has captured interrupt, trap vectors
 - Syscall trap a good one. Can look for exec calls
- After OS has retaken printer interrupt vector
- After virus has noticed loss of printer interrupt vector and recaptured it

How Viruses Work

Macros

Applications like Word or Excel allow macros that get executed via keystroke or menu

Attach a macro to open file function and you are off and running

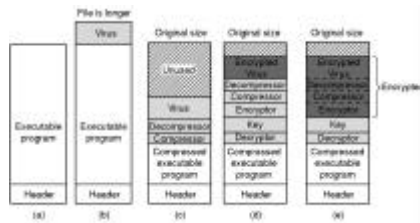
Can be sent in email attachments

Some emailers automatically open attachments

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



- (a) A program
 (b) Infected program, metadata giveaways
 (c) Compressed infected program
 (d) Encrypted virus
 (e) Compressed virus with encrypted compression code

Antivirus and Anti-Antivirus Techniques

```

MOV A,R1      MOV A,R1      MOV A,R1      MOV A,R1      MOV A,R1
ADD B,R1      NOP          ADD #0,R1     OR R1,R1      TST R1
ADD C,R1      ADD B,R1     ADD B,R1      ADD B,R1      ADD B,R1
SUB #4,R1     NOP          OR R1,R1      MOV R1,R5     MOV R1,R5
MOV R1,X      ADD C,R1     ADD C,R1      ADD C,R1      ADD B,R1
              NOP          SHL #0,R1     SHL R1,0      CMP R2,R5
              SUB #4,R1    SUB #4,R1     SUB #4,R1     SUB #4,R1
              NOP          JMP +1        ADD R5,R5     JMP +1
              MOV R1,X  MOV R1,X      MOV R1,X      MOV R1,X
              MOV R5,Y  MOV R5,Y     MOV R5,Y     MOV R5,Y
    
```

(a) (b) (c) (d) (e)

Examples of a polymorphic virus
 All of these examples do the same thing
 Mutation engine – code that morphs the signature part of the virus each time it spreads

Antivirus and Anti-Antivirus Techniques

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

The Internet Worm

- Worm = replicating program
- Nov. 1988, Robert Morris, Cornell grad student
- Consisted of two programs
 - bootstrap to upload worm
 - the worm itself
- Worm first hid its existence
- Next replicated itself on new machines
 - rsh
 - finger name@site - overflow finger daemon's stack with long string
 - Bug in sendmail to mail bootstrap & exec it
 - Tried to break user passwords and go on
- Too aggressive – let 1 in 7 re-infects live
- CERT – Computer Emergency Response Team – collects info on system flaws that can be attacked. Fields reports of security break-ins