Sources of Security Failures

- Lack of security awareness
  - Users
    - silly passwords
    - mailing binaries
    - telnet, etc…..
  - System admins
    - guest accounts
    - default PATH ".:/bin:…"
    - Making available problematic software (e.g telnet)
    - Assuming users are intelligent and aware of security
  - System designers
    - No names ☺

Source of Security Failures

- Featurism
  - Too many features
    - too many potential bugs
  - Complexity
    - Hard to predict interactions

Sources of Security Failures

- Poor design
  - Setuid
    - Violates principle of least privilege
  - Client side validation in client/server systems
- Lazy Programmers
  - Not testing boundary conditions, not initialising storage, not testing input, overflowing stack and buffers

Exploiting Weaknesses

- Target typical failure sources
  - Silly users
  - Silly system admins
  - Lazy programmers/designers
    - Manual says "don't do *X*" then do *X*
    - Try unreasonable/null/invalid/borderline args
    - Read newly allocated storage
    - Kill when doing a privileged operation (e.g. during login)
    - Attempt stack corruption
    - Write your own client
  - Featurism
    - Try unexpected combinations of things

Famous OS/360 Hack

- OS/360 supported asynchronous reading from tape.
- Also had passwords on files
  - Validation code
    1. OS read file name
    2. OS check password
    3. OS re-reads file name
    4. OS opens file
- An exploit???

Famous OS/360 Hack

  - Validation code
    1. OS read file name
    2. OS check password
    3. OS re-reads file name
    4. OS opens file
- Have asynchronous tape overwrite file name between 1 & 3!!!!
  - Race condition in single-tasking system
  - A classic TOCTOU attack
    - Time Of Checking Time Of Use

Michigan Time Sharing System

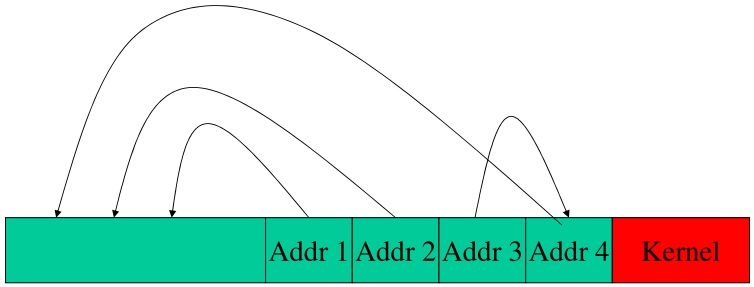
- Certain system calls used indirect addressing for parameters (input and output)

```
syscall(in addr1ptr, out addr2ptr, out addr3ptr, out addr4ptr)
```

- Syscall parameters were validated by checking whether addresses were valid user address before processing the syscall.
- Exploit????

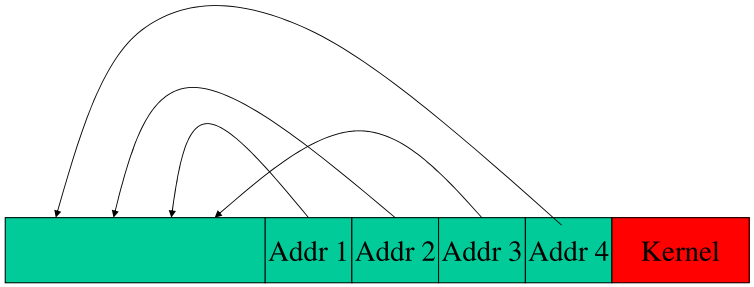
Michigan Time Sharing System

– copy(in addr1, in addr2, out addr3, out addr4)



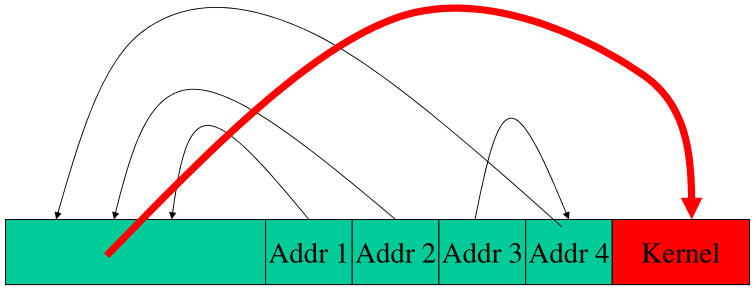
Michigan Time Sharing System

- Illustrative (fictitious) example
  - copy(in addr1, in addr2, out addr3, out addr4)
    - Copy from addr1 to addr3, add2 to addr4



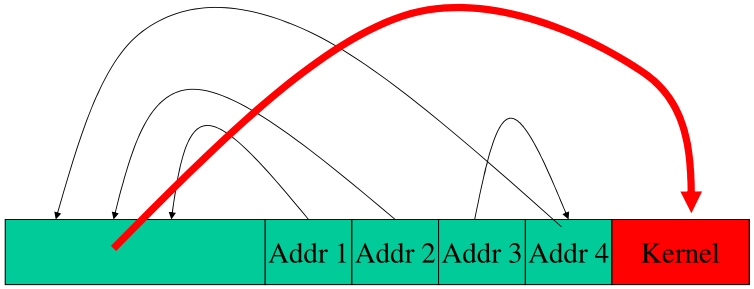
Michigan Time Sharing System

Another TOCTOU attack



Michigan Time Sharing System

– copy(in addr1, in addr2, out addr3, out addr4)



Famous Multics Hack

- Batch jobs could read a card deck and store it into a file anywhere in the file system!!
  - Write a program (card deck) to steal files and call the standard system editor
  - Call it “editor”
  - Install it in the victims ~/bin
- Designer lazyness

Famous Multics Hack

- Multics was an interactive system with excellent security
- Batch features added as an afterthought
- Batch jobs could read a card deck and store it into a file anywhere in the file system
- Exploit???

Famous Burroughs B6700 Hack

- User could backup and restore their own files on tape.
- Users could read and write the tape directly!!!
- Exploit???

Famous Burroughs B6700 Hack

- The computer did not have a privileged mode.
- The compiler (ALGOL) was strongly typed and would not generate privileged code.
  - No assembly code possible.
- Files were typed
- Only files marked as *code files* were executable.
- Only the safe compiler could generate *code files.*

The Famous TENEX Hack

- TENEX was OS for DEC-10, had paged VM
  - Supported a page fault notification function (to collect stats...)
- Had passwords on files
- Password checker:

```
int CheckPassword (char *given, char *passwd) {
   int i;
   for (i=0; i<14; i++) {
       if (passwd[i] != given[i]) {
             return EXIT_FAILURE;
       }
   }
   return EXIT_SUCCESS;
}
```


Famous Burroughs B6700 Hack

- Generate an assembly program to take over the machine
- Back it up
- Change the program type on tape to *code file* by overwriting the type field on the tape appropriately
- Restore the file.
- Designer laziness

Ken Thompson's Compiler Hack

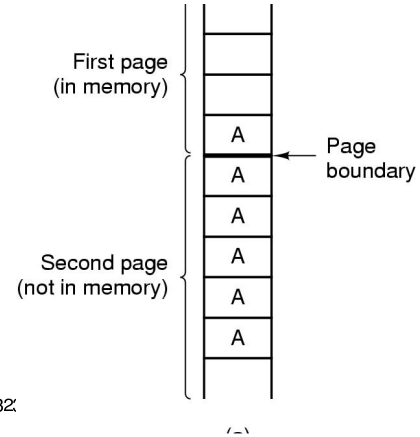
- CORE OF THE C COMPILER;

```
compile(char *s) {
      ...
}
```

  - Reads pre-processed C source (string s) and produces assembler output.

The Famous TENEX Hack

- Hack:
  - Align password across page boundary (after 1st char)
  - Ensure second page is not resident
  - Try **AAAAA**, **BAAAA**, **CAAAA**
  - Page fault
    - got first letter
  - Repeat with page boundary after 2nd char...
  - Worst case 128^n tests
- *Interference of two "harmless" features*



HACKING THE C COMPILER — STEP 2:

```
compile(char *s) {
  if( match( s, "pattern" )) {
    compile( "bug" );
    return;
  }
  if( match( s, "pattern2" )) {
    compile( "bug2" );
    return;
  }
  ...
}
```

- "pattern2": C compiler itself
- "bug2" : the C compiler with the above two *if* statements

HACKING THE C COMPILER — STEP 1:

```
compile(char *s) {
  if( match( s, "pattern" )) {
    compile( "bug" );
    return;
    }
  ...
}
```

- When processing specific code (say, *login)* compile different code (e.g a trapdoor)
- Easy to spot by looking at the compiler source

Famous UNIX Hacks

- **`lpr -r /etc/passwd`**
  - Programmer laziness
- **`ln /etc/passwd core`**
  - run setuid program (at, login, ps) & SIGQUIT
  - use command line args to put useful stuff into core file
  - Interference of "harmless" feature and setuid
- **`mkdir`** was program (setuid root)
  - did **`mknod, chown`**
  - on slow system while mkdir path:
    - **`rm path; ln /etc/passwd path`**
  - Interference between setuid and a race condition

HACKING THE C COMPILER — STEP 3:

- Compile and install the hacked compiler.
- Restore the original C compiler sources.
- Recompiling an unhacked source will still produce a hacked executable

- Morale: "Don't trust a C compiler you haven't hacked yourself!" [K. Thompson, Commun. of the ACM, 27, Aug, 1984]

Sendmail

- SetUID root
- Big and complex
  - The –C option could be used to check a config file
  - If error, the offending file would be printed with appropriate error message
- -C could be used to read any file on the system
  - **`sendmail -C/something/private`**
- SetUID is dangerous

Windows Share Exploits

- Windows 95 had a feature that would attempt to logon to a file server using various file server protocols
  - SMB
  - LANMANAGER
- LANMANAGER used plain text passwords
- IE would automatically attempt to mount a drive if you browsed a “share” URL
- Exploit?

Windows Share Exploits

- Create a web page that references a share on your “server”
- What for the client to mount your LANMAN server using plaintext password
- Example: Insecure Featurism

Windows Share Exploits II

- Samba SMBCLIENT could mount win95 shares (without being authorised)
- Problem according to Microsoft
  - “The Samba SMB client allows its users to send illegal networking commands over the network. “
- Real problem
  - Server was written assuming the behaviour of the client!!!

Windows Share Exploits III

- Windows 95/98/ME can export a share with a password for authentication
- Problem:
  - Server only checked the password up to the length supplied by the client
    - Client could send “a” and match “aardvark”
  - A client program could easily extract the whole password by trying
    - “a-z” to get first letter, then “aa”, “ab”, …, “az” to get next and so forth
- Real problem
  - Stupid server writer

Mobile Phone Featurism

- Send an SMS to some siemens mobile phones
  - Containing “%Language”
    - Where language = English, Deutsch, etc.
- Phone Hangs
  - DOS attack

DLINK DSL 500 Modem/Router/Firewall

- SNMP access enabled by default on external side of firewall
- Default community strings exist
  - Act as passwords
  - “public” give RO access
  - “private” gives RW access
- Can use SNMP to fetch confidential info on modem
  - ISP account name and *password*
- Problem
  - Stupid manufacturer defaults!!!!

Personal Experiences as a System Admin

Apollo Domain OS

- Shipped setUID-root csh
- Blatant manufacturer stupidity

/etc/hosts.equiv

- Used to allow logins from trusted hosts without having to supply a password
  - Only matching login names required
- SUNOS 4.0 shipped with '+' in hosts.equiv
  - Trust everybody
  - Manufacturer carelessness

LD_LIBRARY_PATH

- Build your own libc that adds **`exec("/bin/sh")`**
- Run a setUID dynamically linked binary
- Interference between "harmless" features.

LD_LIBRARY_PATH

- Users can compile their own dynamic libraries but can't install libraries in the system directory
- LD_LIBRARY_PATH allows the system to find a user's libraries
  - `LD_LIBRARY_PATH="/home/kevine/lib:/usr/lib"`
- Exploit???

xterm

- Xterm has a logging feature that would dump all terminal traffic to a log file.
  - Code:
    1. Access("filename) /* check permissions */
    2. Open("filename")
- Between 1 & 2, link the name "filename" to the password file
  - Type in an entry for uid 0

xterm

- Xterm was setUID root in order to write utmp entries
  - Utmp recorded who was logged on
- Xterm has a logging feature that would dump all terminal traffic to a log file.
  - Code:
    1. Access("filename) /* check permissions */
    2. Open("filename")
- Exploit???

WORM'S METHODS OF ATTACK:

- Use rsh to infect trusted hosts
  - /etc/hosts.equiv
  - ~user/.rhosts
- Exploit fingerd bug to get remote root shell:
  - finger param@host
  - param: 536B string containing Sun 3 and VAX code
  - overflow fingerd's argument bufffer
  - overwrite stack
  - exec("/bin/sh")
- Exploit sendmail bug
  - allows mailing and executing bootstrap code
- Crack passwords
  - ... using local on-line dictionary

The Internet Worm

- On 2 Nov. 1988 Robert Tappan Morris, Cornell PG student, released a self-replicating program affecting Sun 3 and VAX computers running BSD.
- Program consisted of two parts:
  - bootstrap code l1.c (99 lines):
    - to run on affected host
    - uploads worm
  - worm:
    - hides traces
    - exits in 6/7 cases when host already infected
    - attempts to spread

What is YOUR Responsibility?

- EXAMPLE: IEEE CODE OF ETHICS
  - http://www.ieee.org/about/whatis/code.html
- We, the members of the IEEE, ..., do hereby commit ourselves to the highest ethical and professional conduct and agree:

1. to accept responsibility in making engineering decisions consistent with the safety, health and welfare of the public, and to disclose promptly factors that might endanger the public or the environment;

9. to avoid injuring others, their property, reputation, or employment by false or malicious action;

RESULTS

- Worm's replication overloaded infected hosts
  - 6/7 "safety margin" was far too low
- Spread world-wide within hours
- Created world-wide chaos on internet
  - most Suns & VAXen down
- Huge cost of lost time world-wide
- Lead to institution of CERT
  - Computer Emergency Response Team
- Morris convicted & sentenced
  - US$10k fine
  - 400h community service
  - 3 years probation
  - estimated >US$150k in legal costs

3. A person who, without authority or lawful excuse, intentionally obtains access to a program or data stored in a computer, being a program or data that the person knows or ought reasonably to know relates to:
   a. a. confidential government information in relation to security, defence or inter-governmental relations; or
   b. b. the existence or identity of any confidential source of information in relation or the enforcement or administration of the law; or
   c. c. the enforcement or administration of the criminal law; ore
   d. d. the maintenance or enforcement of any lawful method or procedure for protecting public safety; or
   e. e. the personal affairs of any person (whether living or deceased); or
   f. f. trade secrets; or
   g. g. records of a financial institution; or
   h. h. information (other than trade secrets) that has a commercial value to any person that could be destroyed or diminished if disclosed,

   is liable to imprisonment for 2 years, or to a fine of $50,000, or both.

NSW Crimes Act
SECTIONS DEALING WITH COMPUTER CRIME
309. Unlawful access to a computer

1. A person who, without authority or lawful excuse, intentionally obtains access to a program or data stored in a computer is liable, on conviction before two justices, to imprisonment for 6 months, or to a fine of $5,000, or both.
2. A person who, with intent:
   1. to defraud any person; or
   2. to obtain for himself or herself or another person any financial advantage of any kind; or
   3. to cause loss or injury to any person

   obtains access to a program or data stored in a computer is liable to imprisonment for 2 years, or to a fine of $50,000, or both.

4. A person who:
   a. without authority or lawful excuse, has intentionally obtained access to a program or data stored in a computer; and
   b. after examining part of that program or data, knows or ought reasonably to know that the part of the program or data examined relates wholly or partly to any of the matters referred to in subsection (3); and
   c. continues to examine that program or data,

is liable to imprisonment for 2 years, or to a fine of $50,000, or both.

310. Damaging data in computer

A person who intentionally and without authority or lawful excuse:

a. destroys, erases or alters data stored in or inserts data into a computer; or
b. interferes with, or interrupts or obstructs the lawful use of a computer,

is liable to penal servitude for 10 years, or to a fine of $100,000, or both.

The aim of this lecture

- Make you aware of security related issues
  - Learn from the mistake of others
- Make you aware of your responsibilities as soon-to-be professionals
- Encourage you to educate yourself
  - On your own equipment
- Warn you that the school has zero tolerance of breaching local system security
  - http://www.cse.unsw.edu.au/school/facilities/yellowform.html