CSci 5271
Introduction to Computer Security
Day 2: Intro to Software and OS Security

Stephen McCamant
University of Minnesota, Computer Science & Engineering

Outline

Security risk and management
Some terminology
Logistics intermission
Example security failures
Software security engineering
Vulnerabilities in OS interaction

Security as an economic good

- Security is a good thing (for defenders)
- But, must trade off other things to get it
- Rational to not purchase all available
- In the big picture, always a compromise

Risk budgeting with ALE

- Annual loss expected = (loss amount)$\times$(incidence)
- Net risk reduction = ΔALE - (security cost)
- Like with a budget, spreadsheet may not match reality
- Like other cost-benefit analysis, can make trade-offs more explicit

Failure: Displacement activity

Security "syllogism" (attributed to: politicians):

1. We must do something
2. This is something
3. Therefore we must do this.

- Example: airport security
- Example: external vs. internal threats

Failure: Risk compensation

- Some benefits of security are taken back by riskier behavior
- Example: H-Day in Sweden
- We'll return to human factors later

This class's perspective

- We'll mostly ignore management issues
- For this class, maximize security at all costs

Outline

Security risk and management

Some terminology

Logistics intermission

Example security failures

Software security engineering

Vulnerabilities in OS interaction

"Trusted"

- In security, "trusted" is a bad word
- X is trusted: X can break your security
- "Untrusted" = okay if it's evil
- Trusted Computing Base (TCB): minimize

"Trusted" vs. "trustworthy"

- Something you actually should trust is "trustworthy"
- Concise definition of security failure: something trusted is not trustworthy

"Privilege"

- Privilege is the power to take security-relevant actions
- Concise definition of security failure: the adversary gets privilege they shouldn't

3 common privilege levels

1. Administrator/root/OS kernel
2. Regular user of system
3. Evil people on the Internet

3 common privilege levels

1. Administrator/root/OS kernel

   ⇑ Local exploit
2. Regular user of system

   ⇑ Remote exploit
3. Evil people on the Internet

Outline

Security risk and management

Some terminology

Logistics intermission

Example security failures

Software security engineering

Vulnerabilities in OS interaction

Posting slides before lecture

- I'll try for 11:55pm on the night before, no guarantees
- Announcements are most likely to change, recheck after

Anyone still not registered?

- On waiting list, came first day ⇒ should have gotten permission number Thursday
- Anyone else left?
- Trade-off: somewhat high student/TA ratio

Exercise set 1

- Available on website
- Due Thursday, September 28th, on Moodle
- Groups of 1-3, turn in one copy

Moodle, group-forming

- Moodle page started
- Group-finding mechanism: forum

Hands-on assignment 1

- Register your groups now, by email to Se Eun (`seoh@umn`)
- Description and code available now
- First, relatively easy attack: due Friday
- But requires learning how to use VM, exploit scripting

Finding project topics

- Pre-proposal due 9/20 (one week from Wednesday)
- Don't skimp on topic: hard but important
- Conference papers linked from class site

More on choosing topics

- Can't: wait to see what part of class you like best
  - But feel free to look ahead
- Think about your group's skills
  - Also: available hardware/software
- Think about where to find novelty
- Topic changes allowed, but will hurt you

Outline

Security risk and management

Some terminology

Logistics intermission

Example security failures

Software security engineering

Vulnerabilities in OS interaction

Classic buffer overflow

```
char buf[20];
gets(buf);
```

- Vulnerability in `finger` daemon
- Morris worm brought down 1988 Internet (4.3BSD VAXes)

Buffer overflow classification

- Bug: stack buffer overflow
- Attack: return address overwrite
- Consequence: (binary) code injection

Read It Twice (WOOT'12)

- Smart TV (running Linux) only accepts signed apps on USB sticks

1. Check signature on file
2. Install file

- Malicious USB device replaces app between steps
- TV "rooted"/"jailbroken"

Confused deputy compiler

- Compiler writes to billing database
- Compiler can produce debug output to user-specified file
- Specify debug output to billing file, disrupt billing
- How to write policy preventing this?

Leaky intelligence analysts

- 1000s of analysts need to view 1000s of classified documents to do their job
- Can we prevent it if one wants to send them to the Washington Post?
- More than regular access control
- (Reality: many non-technical problems)

Outline

Security risk and management

Some terminology

Logistics intermission

Example security failures

Software security engineering

Vulnerabilities in OS interaction

Vulnerabilities are bugs

- Security bugs "just a special case" of bugs
- Like regular bugs, only obscure ones make it through testing
- Key difference:
  - Rare regular bug has limited impact
  - Attackers seek out vulnerability circumstances

Security and quality

- Security correlated with other software quality:
  - Developers understand code well
  - Interactions between modules controlled
  - Well tested

Security and other features

- Security would be much easier if systems were less complex
- But, very few users want that trade-off
- Risk compensation with improvements to development process

Contracts and checks

- Requirement: check X before doing Y
- What function's responsibility is the check?
- Answer embodied in contracts, aka specifications, preconditions and postconditions

Defensive programming

- Analogy: defensive driving
- Don't assume things are right, check
- Inbound: preconditions on arguments
- Outbound: error conditions
- Within reason: some things can't be checked at some places

Outline

Security risk and management

Some terminology

Logistics intermission

Example security failures

Software security engineering

Vulnerabilities in OS interaction

Shell code injection

- Don't pass untrusted strings to a command shell
- In C: `system`, `popen`
- `system("cmd $arg1 $arg2")`
- Fix 1: avoid shell
- Fix 2: sanitize data (preferably whitelist)

Shell code injection example

- Benign: `system("cp $arg1 $arg2")`, arg1 = `"file1.txt"`
- Attack: arg1 = `"a b; echo Gotcha"`
- Command:
  `"cp a b; echo Gotcha file2.txt"`
- Not a complete solution: blacklist ';'

Bad/missing error handling

- Under what circumstances could each system call fail?
- Careful about rolling back after an error in the middle of a complex operation
- Fail to drop privileges $\Rightarrow$ run untrusted code anyway
- Update file when disk full $\Rightarrow$ truncate

Race conditions

- Two actions in parallel; result depends on which happens first
- Usually attacker racing with you

1. Write secret data to file
2. Restrict read permissions on file

- Many other examples

Classic races: files in `/tmp`

- Temp filenames must already be unique
- But “unguessable” is a stronger requirement
- Unsafe design (`mktemp(3)`): function to return unused name
- Must use `O_EXCL` for real atomicity

TOCTTOU gaps

- Time-of-check (to) time-of-use races
  1. Check it’s OK to write to file
  2. Write to file
- Attacker changes the file between steps 1 and 2
- Just get lucky, or use tricks to slow you down

TOCTTOU example

```
int safe_open_file(char *path) {
  int fd = -1;
  struct stat s;
  stat(path, &s)
  if (!S_ISREG(s.st_mode))
    error("only regular files allowed");
  else fd = open(path, O_RDONLY);
  return fd;
}
```

TOCTTOU example

```
int safe_open_file(char *path) {
  int fd = -1, res;
  struct stat s;
  res = stat(path, &s)
  if (res || !S_ISREG(s.st_mode))
    error("only regular files allowed");
  else fd = open(path, O_RDONLY);
  return fd;
}
```

TOCTTOU example

```
int safe_open_file(char *path) {
  int fd = -1, res;
  struct stat s;
  res = stat(path, &s)
  if (res || !S_ISREG(s.st_mode))
    error("only regular files allowed");
  else fd = open(path, O_RDONLY);
  return fd;
}
```

Changing file references

- With symbolic links
- With hard links
- With changing parent directories

Directory traversal with ..

- Program argument specifies file, found in directory `files`
- What about `files/../../../../etc/passwd`?

Environment variables

- Can influence behavior in unexpected ways
  - `PATH`
  - `LD_LIBRARY_PATH`
  - `IFS`
  - ...
- Also umask, resource limits, current directory

IFS and why it's a problem

- In Unix, splitting a command line into words is the shell's job
  - String $\rightarrow$ argv array
  - `grep a b c` vs. `grep 'a b' c`
- Choice of separator characters (default space, tab, newline) is configurable
- Exploit `system("/bin/uname")`

Next time

- Bugs particular to low-level (e.g., C) programs