CSci 4271W
Development of Secure Software Systems
Day 9: More Unix Access Control

Stephen McCamant
University of Minnesota, Computer Science & Engineering

Outline

Setting: files related to this class

- Student and course staff materials
- Imagine everything is in Unix files on CSE Labs
  - Versus reality of a mixture of Unix with web-based systems like Canvas

Users and groups

- Users: smccaman (instructor), paul1155 (TA), stude003 (student)
- Groups: csci4271staff (instructor and TA), csci4271all (staff and students)

What I want from you

- First, think of a kind of file/directory/information that would be relevant to the class
- Then, decide on the appropriate octal permissions bits, plus owner and group, that would be appropriate
- Then repeat with a new resource, looking for one with different permissions bits

Outline

Injection vulnerabilities

- Common dangerous pattern: interpreter code with attacker control
- Interpreted language example: `eval`
- OS example: shell script injection
- Web examples: JavaScript (XSS), SQL injection
- C library example: `printf` format string

`printf` reminder

- `printf` (and related functions like `fprintf` are a convenient way to produce formatted output
- The format string argument contains format specifiers (starting with %) controlling how the other arguments are interpreted

```
printf("Function %s is at address %016x\n",
       name, addr);
```

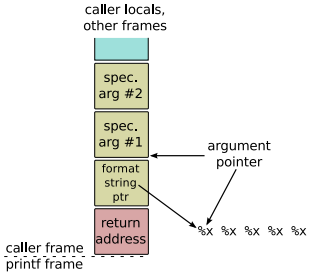
Variable arguments functions

- C has special features for functions like `printf` that take a varying number of arguments
  - Macros `va_start`, `va_arg`, etc.
- Compiler can't check type or number of arguments
- Args will be stored on stack, for pointer access

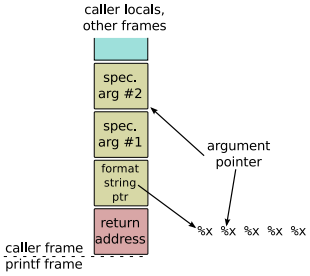
Format string attack

- In secure code, format strings should not be under external control
  - Common case: just constant strings
- What malicious things can an attacker do via a format string?
- Step one: add extra integer specifiers, dump stack
  - Already useful for information disclosure

Format string attack layout



Format string attack layout



Format string attack: overwrite

- `%n` specifier: store number of chars written so far to pointer arg
  - Benign but uncommon use: account for length in other formatting
- Advance format arg pointer to other attacker-controlled data
- Control number of chars written with padding
- Net result is a "write-what-where" primitive

Practical format string challenges

- Attacker usually must control format as well as one or more arguments
- Writing a big value requires impractical output size
  - Workaround 1: overwrite two bytes with `%hn`
  - Workaround 2: use overlapping unaligned write to control byte by byte

Format string defenses

- Compilers will warn for `printf` that looks like it should just be `puts`
- Several platforms have decided to just remove `%n`
  - Android Bionic, Visual Studio
- Linux glibc by default will block `%n` if the format string is writeable
- Major remaining use is information disclosure

Outline

Complete project 1 instructions posted

- Provides more detail beyond previous in-class announcements
  - Available from Assignments page of public site
- Most important reminder: initial report due Friday by 11:59pm

Supplemental office hours

- I will host another office hour after class (5:15-6:15) today
- May continue based on demand
- Please also take advantage of Piazza, we'll be active there too

Outline

Exercise: using Unix permissions

Injection vulnerabilities: format strings

Logistics announcements

Good technical writing (pt. 1)

More Unix permissions

Writing in CS versus other writing

- Key goal is accurately conveying precise technical information
- More important: careful use of terminology, structured organization
- Less important: writer's personality, appeals to emotion

Still important: concise expression

- Don't use long words or complicated expressions when simpler ones would convey the same meaning
- Beneficial for both clarity and style

Know your audience

- When technical terminology makes your point clearly, use it
- But provide definitions if a concept might be new to many readers
  - Be careful to provide the right information in the definition
  - Define at the first instead of a later use
- On other hand, avoid introducing too many new terms
  - Reuse the same term when referring to the same concept

Precise explanations

- Don't say "we" do something when it's the computer that does it
  - And avoid passive constructions
- Don't anthropomorphize (computers don't "know")
- Use singular by default so plural provides a distinction:
  - The students take tests
  + Each student takes a test
  + Each student takes multiple tests

Provide structure

- Use plenty of sections and sub-sections
- It's OK to have some redundancy in previewing structure
- Limit each paragraph to one concept, and not too long
  - Start with a clear topic sentence

Outline

Exercise: using Unix permissions

Injection vulnerabilities: format strings

Logistics announcements

Good technical writing (pt. 1)

More Unix permissions

Process UIDs and `setuid(2)`

- UID is inherited by child processes, and an unprivileged process can't change it
- But there are syscalls root can use to change the UID, starting with `setuid`
- E.g., login program, SSH server

Setuid programs, different UIDs

- If 04000 "setuid" bit set, newly exec'd process will take UID of its file owner
  - Other side conditions, like process not traced
- Specifically the *effective UID* is changed, while the *real UID* is unchanged
  - Shows who called you, allows switching back

More different UIDs

- Two mechanisms for temporary switching:
  - Swap real UID and effective UID (BSD)
  - Remember *saved UID*, allow switching to it (System V)
- Modern systems support both mechanisms at the same time

Setgid, games

- Setgid bit 02000 mostly analogous to setuid
- But note no supergroup, so UID 0 is still special
- Classic application: setgid `games` for managing high-score files

Special case: `/tmp`

- We'd like to allow anyone to make files in `/tmp`
- So, everyone should have write permission
- But don't want Alice deleting Bob's files
- Solution: "sticky bit" 01000

Special case: group inheritance

- When using group to manage permissions, want a whole tree to have a single group
- When 02000 bit set, newly created entries with have the parent's group
  - (Historic BSD behavior)
- Also, directories will themselves inherit 02000

Other permission rules

- Only file owner or root can change permissions
- Only root can change file owner
  - Former System V behavior: "give away `chown`"
- Setuid/gid bits cleared on `chown`
  - Set owner first, then enable setuid