CSE 40567 / 60567: Computer Security

Software Security 5
Homework #5 has been released. It is due 3/26 at 11:59PM

See Assignments Page on the course website for details
Format Strings Attack

Another common bug:

```c
int func(char *user){
    fprintf(stderr, user);
}
```

Problem: what if `*user = "%s%s%s%s%s%s"`?

- Most likely, program will crash: DoS.
- If not, program will print memory contents. Privacy impact?
- Full exploit using `*user = "%n"

Correct use of `fprintf()`:

```c
fprintf(stdout, "%s", user);
```
Vulnerable functions

Any function using a format string:

Printing:

\[\text{printf}(), \text{fprintf}(), \text{sprintf}(), \ldots\]
\[\text{vprintf}(), \text{vfprintf}(), \text{vsprintf}(), \ldots\]

Logging:

\[\text{syslog}(), \text{err}(), \text{warn}()\]
Writing a format strings exploit

- Dumping arbitrary locations in memory:
  - Walk up the stack until desired pointer is found.
  - `printf("%08x.%08x.%08x.%08x|%s\")` 

- Write to arbitrary locations in memory:
  - `printf("hello %n", &temp) // writes ‘6’ into memory`
  - `printf("%08x.%08x.%08x.%08x.%08x.%n")`
Race Conditions

- General category of bug (not always a security problem)
- Affects systems where the output is dependent on the sequence or timing of other uncontrollable events
- Becomes a bug when events do not happen in the order the programmer intended
- Trouble for authentication:
  - (1) Check for authentication, (2) State changes, (3) Act on authentication
Example: race condition exploit in Starbucks gift cards

• Hole in online gift card purchases surfaced in 2015

• Initiate two identical web store transfers, trick the store into recording both
  ‣ Normal use: Move money from one card with $5 onto another, for a total of $10 on one card
  ‣ With duplicate transfer: end up with $15 on one card!

https://www.schneier.com/blog/archives/2015/05/race_condition_.html
Helpful Tools for Writing Exploits
Metasploit (www.metasplloit.com)

Tool for developing and executing exploit code against a target machine
What does metasploit do?

- Provides tools that make debugging, offset hunting and payload crafting easier
- Basic steps for exploiting a system via the framework:
  1. Choose and configure an exploit (over 900 available in the framework)
  2. Check whether a target system is vulnerable
  3. Choose and configure a payload
  4. Encode payload to evade IDS
  5. Execute the exploit
Disassembling code

• Broad knowledge of assembly language is essential for writing exploits
  ‣ Need to know where function calls exist in memory, the internal execution flow of the program, and the state of the heap / stack (eip / rip manipulation)
  ‣ Sometimes source code isn’t available; the binary must be examined in these cases
(gdb) disass main
Dump of assembler code for function main:
  0x00000000000400624 <+0>: push %rbp
  0x00000000000400625 <+1>: mov %rsp,%rbp
  0x00000000000400628 <+4>: sub $0x30,%rsp
  0x0000000000040062c <+8>: mov %edi,-0x24(%rbp)
  0x00000000000400633 <+15>: cmpl $0x4,-0x24(%rbp)
  0x00000000000400637 <+19>: je 0x40063e <main+26>
  0x00000000000400639 <+21>: jmpq 0x400731 <main+269>
  0x0000000000040063e <+26>: mov $0x100,%edi
  0x00000000000400643 <+31>: callq 0x400520 <malloc@plt>
  0x00000000000400648 <+36>: mov %rax,-0x18(%rbp)
  0x0000000000040064c <+40>: mov $0x40082c,%eax
  0x00000000000400651 <+45>: mov -0x18(%rbp),%rdx
IDA PRO

https://www.hex-rays.com/index.shtml

https://youtu.be/vb18UVF4a_o
Fuzzing

• Black-box testing methodology
• Checks code modules for vulnerability to overflows
  ‣ Many are not obvious to visual inspection
• Two fuzzing strategies are typically deployed
  1. Mutation-based fuzzers
  2. Generation-based fuzzers
CERT Basic Fuzzing Framework (BFF)

Mutational fuzzer

https://github.com/CERTCC-Vulnerability-Analysis/certfuzz

https://youtu.be/kSnc7RI5ByA
w3af

http://w3af.org/

Features:

• Daemons
• Fast HTTP Client
• Output Manager
• Fuzzing Engine
• Knowledge base
skipfish

https://code.google.com/archive/p/skipfish/

Crawl results - click to expand:

http://www.example.com/  4  6  67

New 404 signature seen

New 'Server' header value seen

.svn  1

egi-bin

.error  1  5

    .svn  1

    include  3

    README  1

    icons  4  2  57

index.html
Code: 200, length: 596, declared: text/html, charset: UTF-8 [show trace +]

Document type overview - click to expand:

application/xhtml+xml (5)

https://www.owasp.org/index.php/Automated_Audit_using_SKIPFISH