CS177 Computer Security Discussion
Spring 2020 - Week 7

Dongyu (Hector) Meng
May 11th

Today

- **minecraft_hello** and **minecraft** (more on **lazy_panel** next week)
- Challenge roadmap
- Know your tools
    - Pwntools (connection, shellcoding)
    - Ghidra
    - GDB

minecraft_hello

- Run it
- How the service is being run on the server?
- Take a look at the source
- Compilation options and ***checksec***

minecraft_hello

- ***mine*** leaks the values on stack
- Overflow ***buf*** when you leave your name ***carefully*** (more later)
- Jump to win to get a shell
- Cat the flag

Talk to the program

- Exploit locally before you try remote exploit!
- Talk to the program with ***pwntools***

```python
from pwn import *
context(arch = 'i386', os = 'linux')
context.log_level = "DEBUG"

r = remote('exploitme.example.com', 31337)
# r = process('./your_binary')
r.send(asm(shellcraft.sh()))
r.interactive()
```

Dig the binary

- Source code is not low-level enough!
- Compiler makes a lot of decisions
- Inspect the binary with ***Ghidra*** (IDA, r2, Binary Ninja)
- Get some intuitions to map things together (run helper)

Things go wrong

- Debug!
- **GDB** is your friend.
- GDB disables ASLR by default, enable follow instructions here:
  https://cysecguide.blogspot.com/2017/10/enable-and-disable-aslr-on-linux-gdb.html
- Or just use ***attach*** with pwntools to attach to a running process
- Example

The catch

- In the epilog, the challenge binary does something different from the calling convention introduced in class
- Return address is not a special place on memory!
- ***ret*** jumps to the address right on top of the stack and pops
- Solve that and you get the challenge

Registers

%rax	
%rbx	
%rcx	0x4
%rdx	0x100
%rdi	
%rsi	

Memory	Word Address
0x400	0x120
0xF	0x118
0x8	0x110
0x10	0x108
0x1	0x100

```
leaq (%rdx,%rcx,4), %rax
movq (%rdx,%rcx,4), %rbx
leaq (%rdx), %rdi
movq (%rdx), %rsi
```

Registers

%rax	0x110
%rbx	0x8
%rcx	0x4
%rdx	0x100
%rdi	0x100
%rsi	0x1

Memory	Word Address
0x400	0x120
0xF	0x118
0x8	0x110
0x10	0x108
0x1	0x100

```
leaq (%rdx,%rcx,4), %rax
movq (%rdx,%rcx,4), %rbx
leaq (%rdx), %rdi
movq (%rdx), %rsi
```

minecraft

- *mine* leaks the values on stack
- Find out where the stack is as well as the canary with ***mine***
- Overflow ***buf*** when you leave your name ***carefully***
- Leave the shellcode you crafted at the same time
- Jump to shellcode to get a shell
- Cat the flag

Shellcoding with Pwntools

Pwntools shellcraft

https://docs.pwntools.com/en/stable/shellcraft.html