Format Strings

Announcements

- DiceCTF TOMORROW
  - Be there!
  - Let's be top 3
  - There will be pizza
- TracerFire March 5 and 6
  - Form to sign up in #announcements

Meeting Flag

sigpwny{printf(user_input)}

Overview

- Format string review
- Example bad program
- Arbitrary read
- Arbitrary write

Overview

- `printf("hello world"); // prints hello world`
- `printf("%d", 123); // prints 123`
- `printf("%x", 0xcafebabe); // prints cafebabe`
- `printf("%s", "sigpwny"); // prints sigpwny`
- `printf("%3$d", 123, 456, 789); // prints 789`

Correct program

```
int main() {
    char buf[32];
   // put user input in buf
    printf("%s", buf);
}
```

Vulnerable program

```
int main() {
    char buf[32];
    // put user input in buf
    printf(buf);
}
```

Vulnerable program

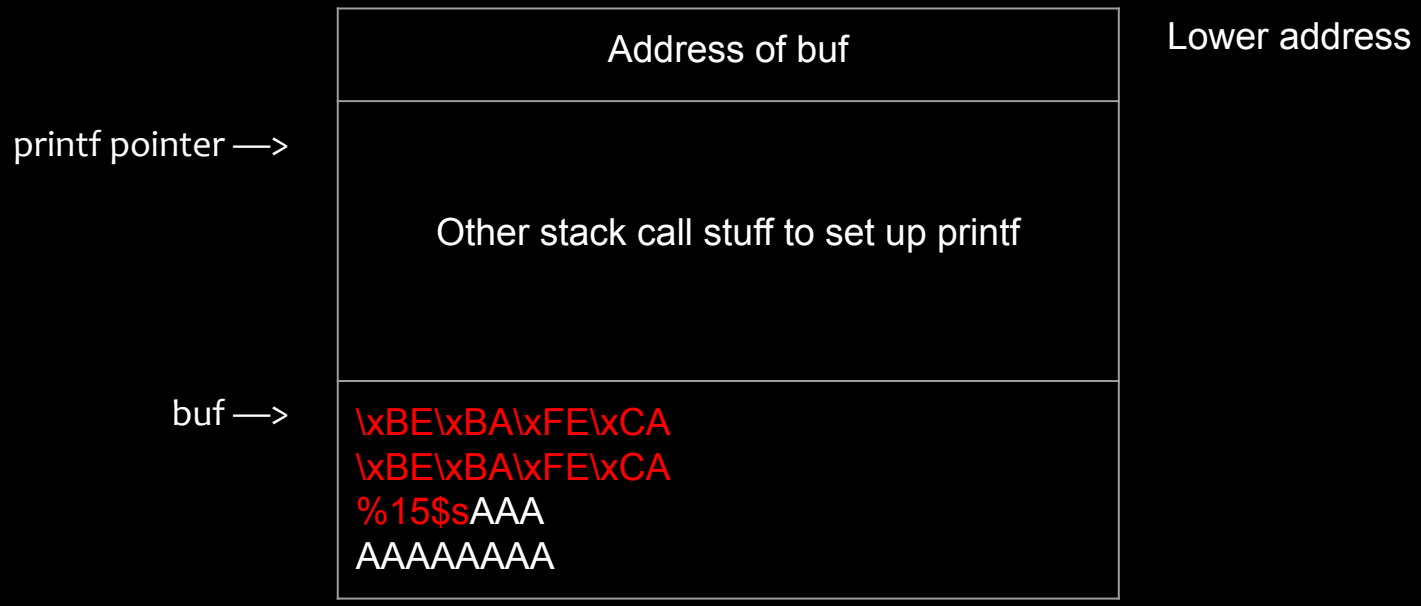
- What if user input is %x?
- What if user input is %s?

```
int main() {
    char buf[32];
    // put user input in buf
    printf(buf);
}
```

printf stack (kind of)

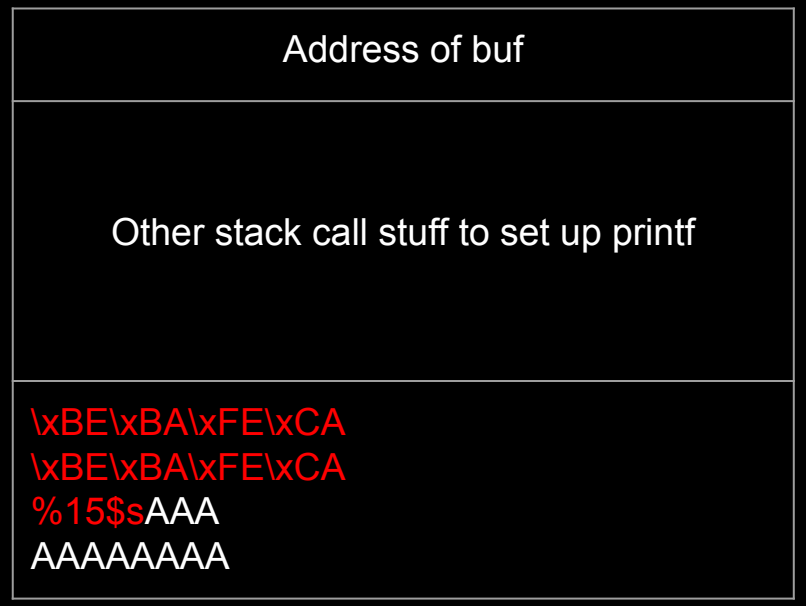

```
int main() {
   printf("%d%d", 123, 456);
}
```

printf stack (kind of)




```
int main() {
    char buf[32];
    printf(buf);
}
```

printf stack (kind of)




```
int main() {
    char buf[32];
    printf(buf);
}
```

Prints the string at address 0xCAFEBABECAFEBABE!

Format String Read Recap

- Can do arbitrary memory read
  1. Specify address to read at start of buf
  2. Spam %x until printf pointer is at start of buf or use direct parameter access
  3. Add a %s

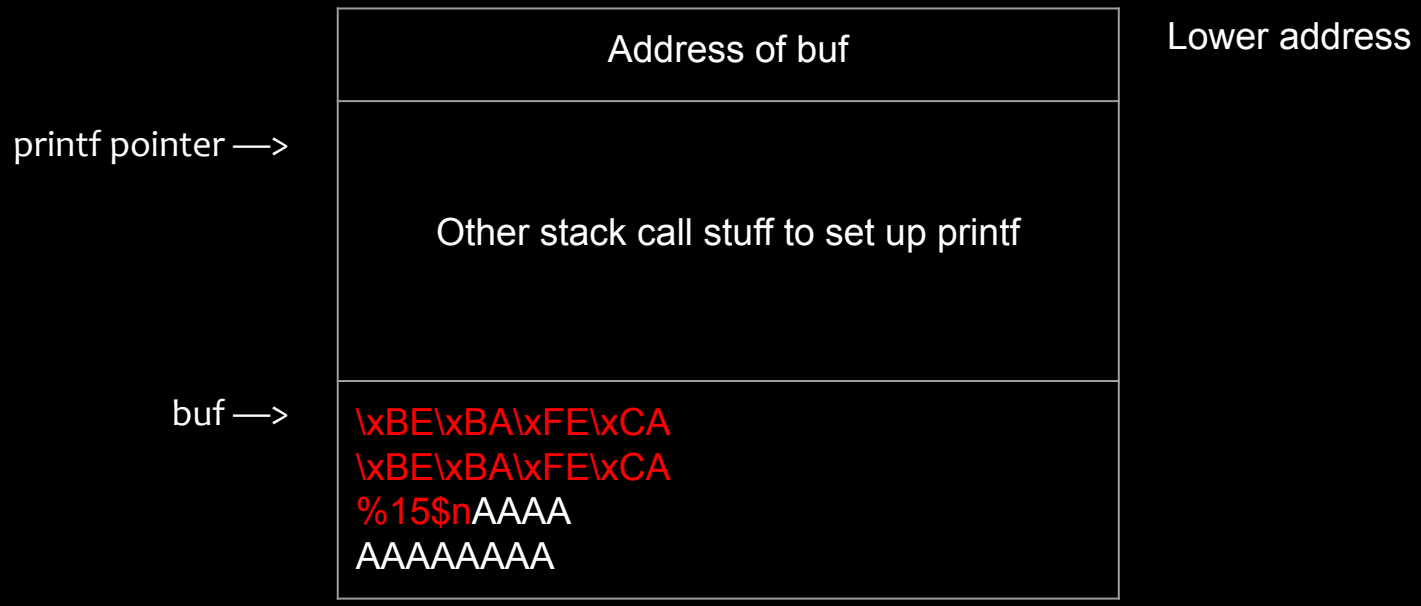
- Can also spam %x or %p to leak addresses
  - libc addresses
  - program addresses
  - stack, heap

%n

- Writes the number of bytes printed so far into a pointer
  - Added to printf just so we can exploit format string vulns???

```c
int main() {
    int bytes_written = 0;
    printf("hello world\n%n", &bytes_written);
    // bytes_written is now 12
    return 0;
}
```

printf stack (kind of)




```
int main() {
    char buf[32];
    printf(buf);
}
```

Writes 8 to address 0xCAFEBABECAFEBABE!

Exploiting arbitrary write

- Consider the format string: “(ADDRESS) (ADDRESS+1) (ADDRESS + 2) … %150x %15$n %120x %16$n %64x %17$n
- Writes 150 to ADDRESS, then (150 + 120) % 256 to ADDRESS + 1
  - Effectively writing one byte at a time, thanks to little endian
- Hard part is getting addresses on the stack, since 64 bit addresses have lots of \x00 bytes. If ADDRESS has \x00 bytes, then place at end of printf call and hope your string input accepts \x00 bytes.

Exploiting arbitrary write

- Overwrite function pointers
  - Function pointers might be in some struct somewhere you leaked or in the GOT if writable

  - Point that to a win function, some shellcode, or a one gadget

Next Meetings

This Sunday:

- DiceCTF!

Next Thursday:

- Log4j with Minh