

A Broad Intro to Reverse Engineering

WiCyS Illinois

What is it?



The act of understanding how something works, usually without prior knowledge of how that thing works

Some examples involving software:

- You have a program but don't have source code/documentation
- You have source code, but it's unreadable
- You have an API and don't know what endpoints exist/what they do
- You have encrypted things and want to break their encryption

Even if security isn't your focus, you are highly likely to deal with undocumented code, APIs, protocols, etc. as a software engineer

If you want security engineering to be your focus, great for CTFs, finding vulnerabilities, analyzing malware, pretty much everything

Also, if you want to mod games or build emulators for old game consoles, some reverse engineering will be required



Why
should I
care?


What skills are needed?

A lot!

Some specifics that might help:

- Assembly
- C/C++
- Debugging tools like GDB
- Network analysis tools like Wireshark
- Decompiling tools
- Disassembly tools

Today's CTF problem – binary exploitation!



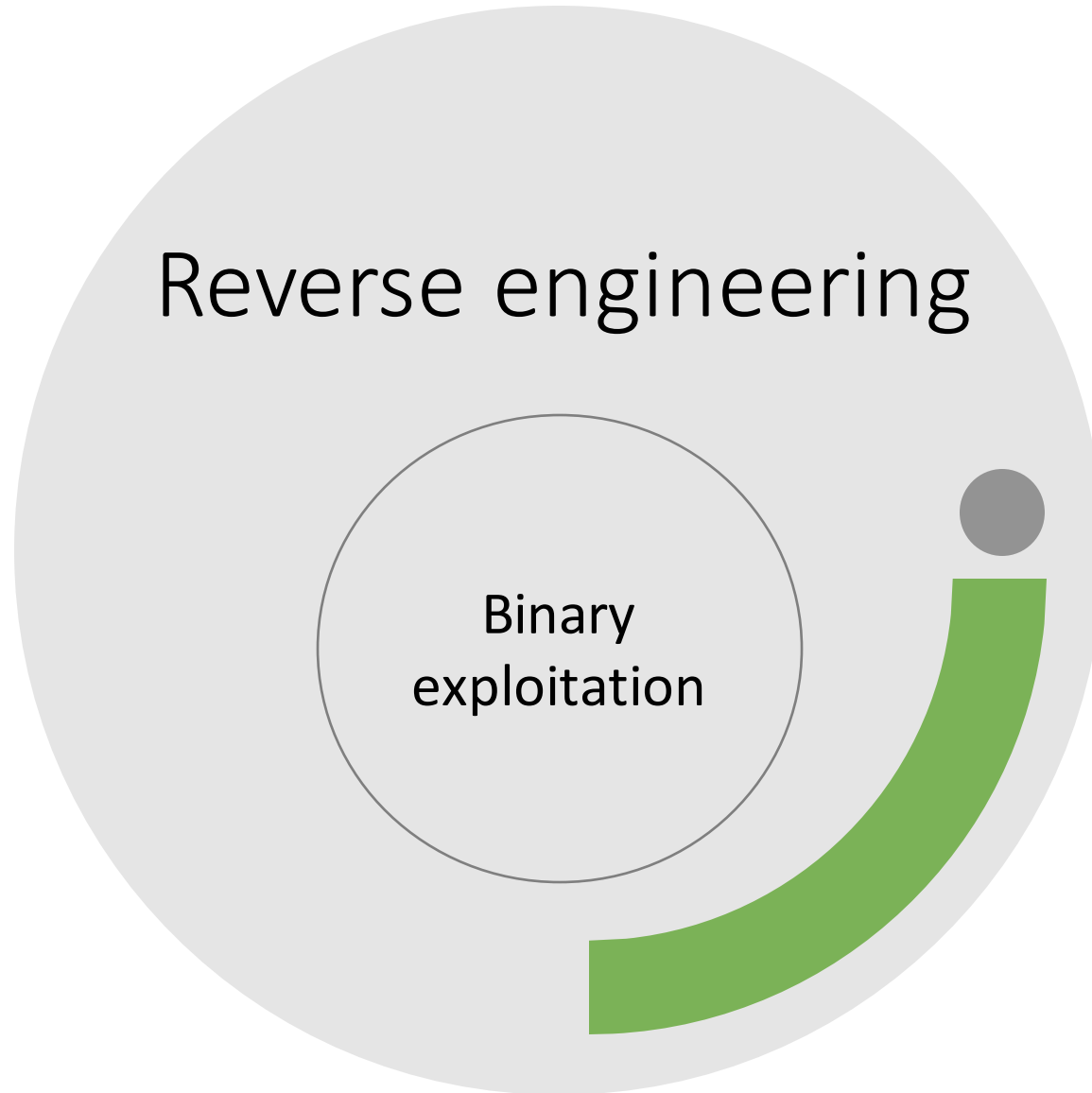
First problem in OverTheWire's Narnia games

Link: <https://overthewire.org/wargames/narnia/>

Access the problem with SSH

- **Host name:**
narnia.labs.overthewire.org
- **Port number:** 2226
- **Username:** narnia0
- **Password:** narnia0

(Feel free to use an SSH client like PuTTY, but it isn't necessary.)



Binary exploitation – tends to involve finding a vulnerability in an executable to change part of the program

Binary exploitation typically involves reverse engineering, but reverse engineering is a much broader topic

- C exists, and this problem involves some
- Program memory exists in stack form
- Handling strings in C is sometimes problematic
- Number systems can also get interesting

Background
knowledge
we'll try to
skim through

The problem content

This problem involves a little bit of C

The parts which are most interesting:

- char buf[20]
- scanf
- system("bin/sh")

```
long val=0x41414141;
```

```
char buf[20];
```

```
printf("Correct val's value from 0x41414141 -> 0xdeadbeef!\n");
```

```
printf("Here is your chance: ");
```

```
scanf("%24s",&buf);
```

```
printf("buf: %s\n",buf);
```

```
printf("val: 0x%08x\n",val);
```

```
if(val==0xdeadbeef){
```

```
    setreuid(geteuid(),geteuid());
```

```
    system("/bin/sh");
```

```
}
```

```
else {
```

```
    printf("WAY OFF!!!!\n");
```

```
    exit(1);
```

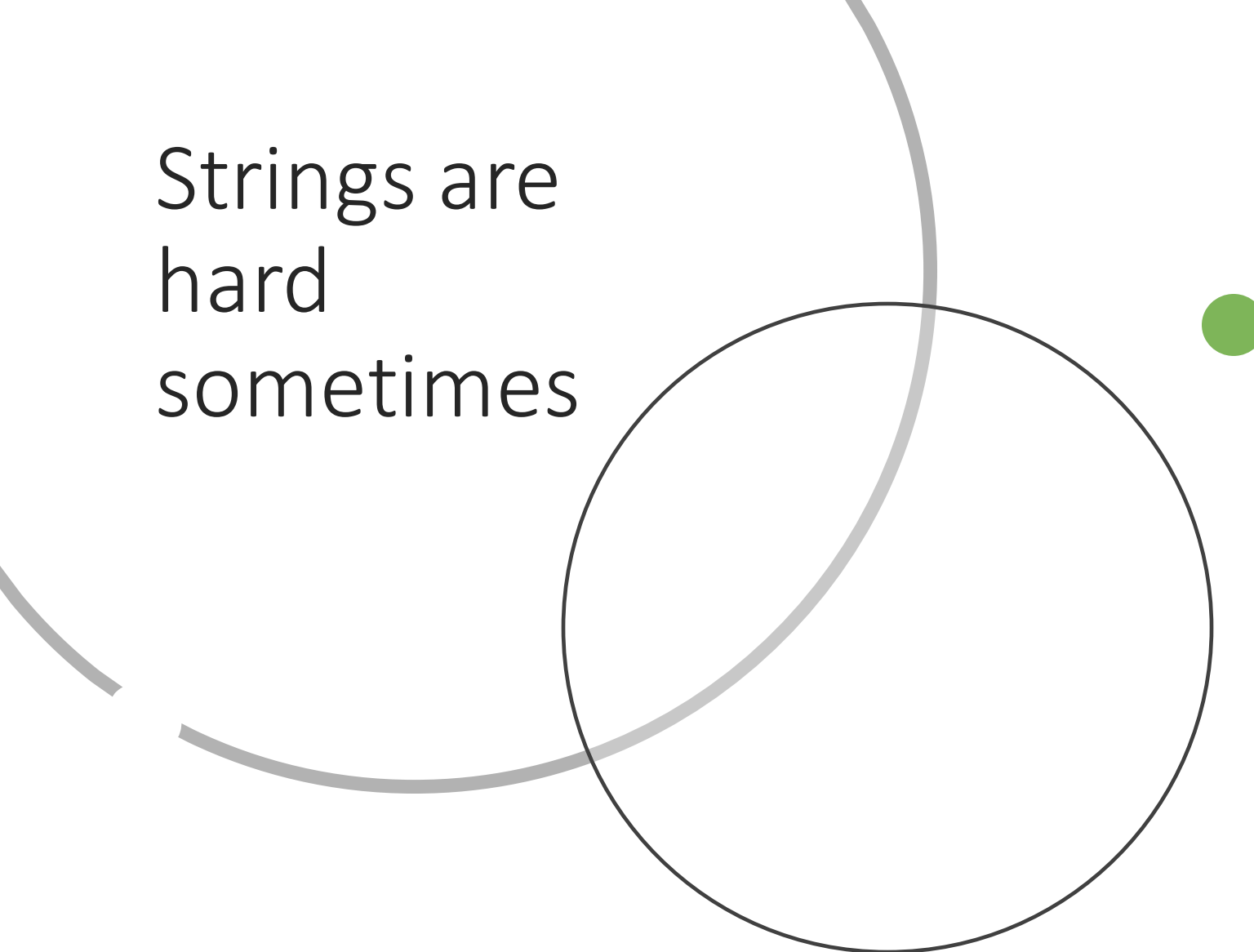
```
}
```


Program memory in C

- Data in C is stored within a stack
- Stack starts at high memory addresses, moves to lower ones over time
- Every variable, function, etc. ends up as a number on the stack (pretend the heap doesn't exist for now since it's not part of this problem)



Strings are
hard
sometimes

A decorative graphic consisting of two overlapping circles. The left circle is light gray and partially cut off by the left edge of the slide. The right circle is white with a thin black outline. A solid green dot is positioned to the right of the intersection of the two circles.

- For scanf, one must specify the size of the buffer in advance, even without knowing the user input
- The original buf variable was given 20 bytes, but the scanf line lists 24 bytes...
- This might be an issue.

Number systems and ASCII

- The variables in this code are assigned in hex values
- Even though we know the variable should be set to "deadbeef"
- Is the ASCII representation the same?

https://www.tutorialspoint.com/html/ascii_table_lookup.htm

So... how can we solve the problem?

1. Need to successfully modify 0xdeadbeef with the right user input
2. Once modified, need to find out how to move on to the next challenge

```
long val=0x41414141;
```

```
char buf[20];
```

```
printf("Correct val's value from 0x41414141 -> 0xdeadbeef!\n");
```

```
printf("Here is your chance: ");
```

```
scanf("%24s",&buf);
```

```
printf("buf: %s\n",buf);
```

```
printf("val: 0x%08x\n",val);
```

```
if(val==0xdeadbeef){
```

```
    setreuid(geteuid(),geteuid());
```

```
    system("/bin/sh");
```

```
}
```

```
else {
```

```
    printf("WAY OFF!!!!\n");
```

```
    exit(1);
```

```
}
```

<https://overthewire.org/wargames/narnia/>

<https://beginners.re/>

<https://opensecuritytraining.info/IntroductionToReverseEngineering.html>

<https://github.com/tylerha97/awesome-reversing>



Some
useful
links