Stack Smashing Attacks

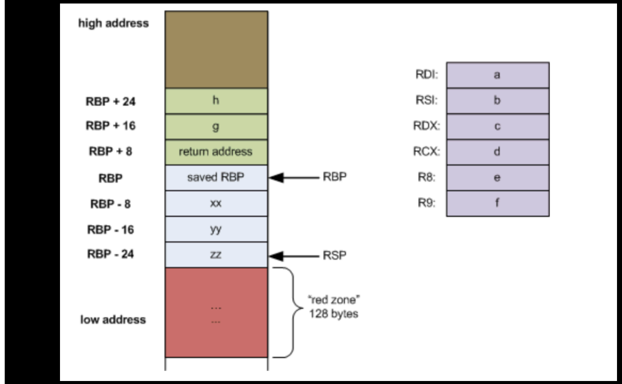
Buffer Overflows, Format Strings and Stack Canaries

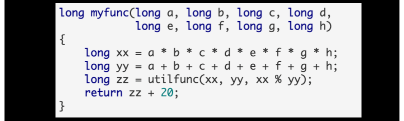
Rathi Kashi, 13th April 2023

Today’s presentation will cover

- Recap of Buffer overflows
- Intro to pwntools
- Intro to Format Strings
- Exploiting the format string vulnerability
- Intro to Stack Canaries
- Designing an attack to bypass the Canary

Stack and Registers: x86-64





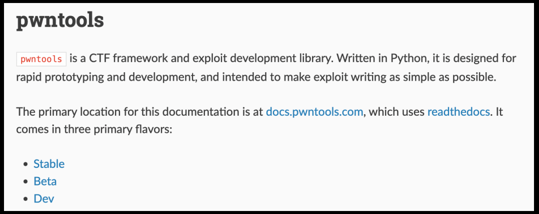
```
long myfunc(long a, long b, long c, long d,
            long e, long f, long g, long h)
{
    long xx = a * b * c * d * e * f * g * h;
    long yy = a + b + c + d + e + f + g + h;
    long zz = utilfunc(xx, yy, xx % yy);
    return zz + 20;
}
```

Buffer Overflow

- When a system writes more data to a buffer than it can hold, a buffer overflow or buffer overrun occurs
- Why? No bounds checking in C/C++ without the overhead of additional processing time
- Common types of stack overflows involve, overwriting local variables or the return address

Pwntools

Links!



pwntools

`pwntools` is a CTF framework and exploit development library. Written in Python, it is designed for rapid prototyping and development, and intended to make exploit writing as simple as possible.

The primary location for this documentation is at docs.pwntools.com, which uses readthedocs. It comes in three primary flavors:

- Stable
- Beta
- Dev

- https://ir0nstone.gitbook.io/notes/other/pwntools/introduction
- https://docs.pwntools.com/en/latest/

Format String Vulnerabilities

Why printf can be deadly

- Format Function: printf/fprintf
- Format String: printf(“The result is: %d\n”, 25) (text + parameters)
- Format String Specifiers: %x, %s (type of conversion)
- When format string specifiers are passed without enough arguments: allows for reading/writing from memory!

```
printf("Hello, my name is %s.", name);
```

```
printf("A is the number %d, reading stack data: %x", A);
```

```
printf("%x%x%x%x%x%x%x%x%x%x%x%x%x%x%x%x%x%x%x%x");\
```

```
printf("%10$x");\
```

Format String Vulnerabilities

Why printf can be deadly

```
printf("\xef\xbe\xad\xde%x%x%x%s", A, B, C);
```

```
int num_char;
printf("11111%n", &num_char);
```

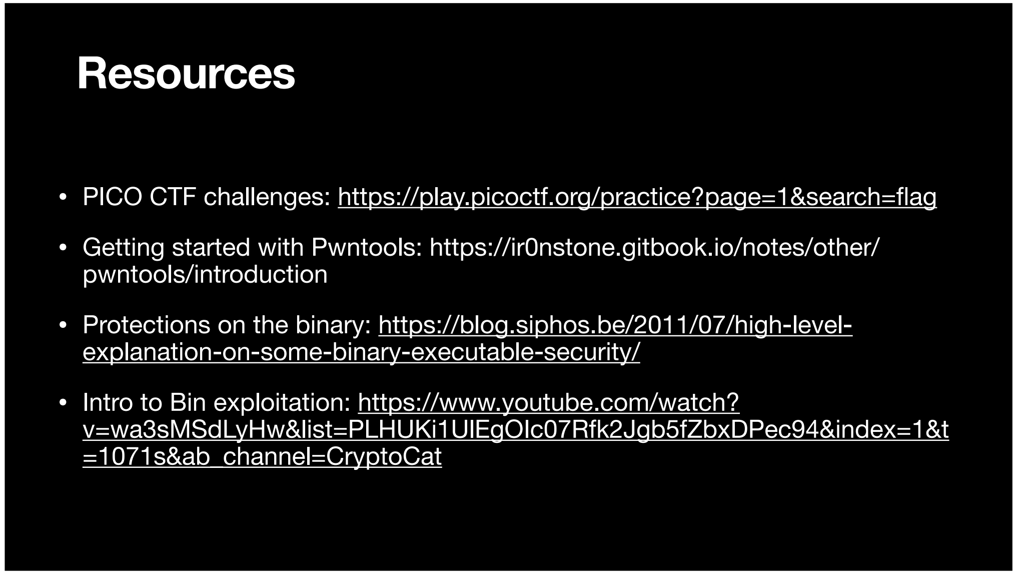
- When %s is used as the format specifier, the function will treat the data on the stack as an address to go fetch a string from. This is called pass by reference.
- Attacker can place an address on a stack and dereference it.

https://vickieli.dev/binary%20exploitation/format-string-vulnerabilities/

Stack Canaries

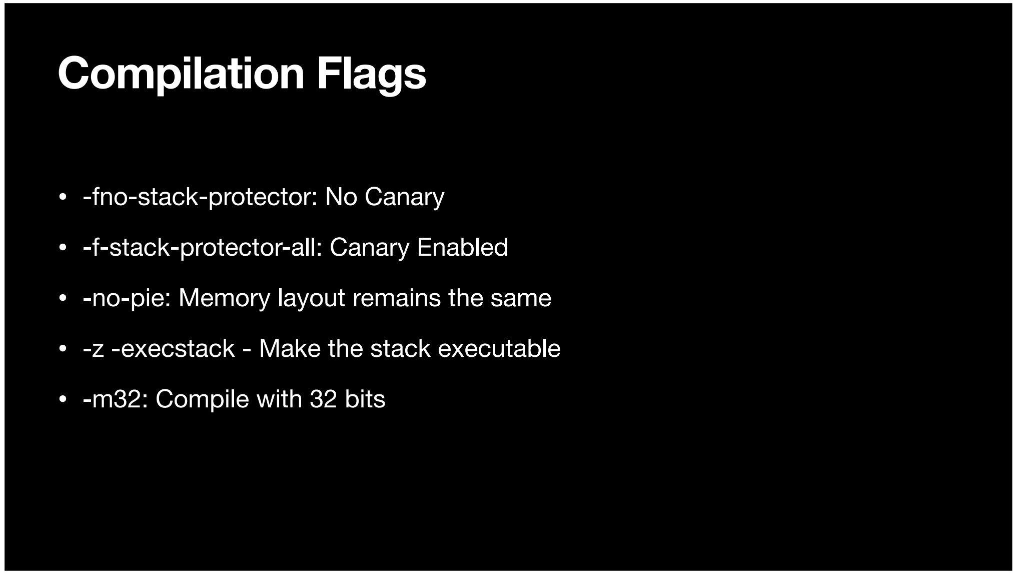
Preventing buffer overflows

- Stack Canaries are a warning to the computer that there has been a buffer overflow attack
- This a random number that lies between a buffer and the function’s return address.
- Before the function is returned, the canary is checked to ensure that it matches the value at the beginning of execution. If it does, program executes. If not, the program crashes, “stack smashing detected!”
- Fact: Canaries were used as carbon monoxide detectors in coal mines



Resources

- PICO CTF challenges: https://play.picoctf.org/practice?page=1&search=flag
- Getting started with Pwntools: https://ir0nstone.gitbook.io/notes/other/pwntools/introduction
- Protections on the binary: https://blog.siphos.be/2011/07/high-level-explanation-on-some-binary-executable-security/
- Intro to Bin exploitation: https://www.youtube.com/watch?v=wa3sMSdLyHw&list=PLHUKi1UlEgOIc07Rfk2Jgb5fZbxDPec94&index=1&t=1071s&ab_channel=CryptoCat



Compilation Flags

- -fno-stack-protector: No Canary
- -f-stack-protector-all: Canary Enabled
- -no-pie: Memory layout remains the same
- -z -execstack - Make the stack executable
- -m32: Compile with 32 bits