Echo Chamber 1

Problem Type

- Format String Vulnerability

Checksec

```
Arch:     amd64-64-little
RELRO:    Partial RELRO
Stack:    Canary found
NX:       NX enabled
PIE:      PIE enabled
```

Solution

Vulnerability

The program takes in user input into the buffer, and prints it out with `print(buf)`.

```
printf("Shout something: ");
        fgets(buf,sizeof(buf),stdin);
        printf("\n The cave echoes back: ");
        printf(buf);
```

`printf(buf)` is susceptible to format string attacks.

Format String Attacks

`printf()` converts a 'primitive variable of the programming language into a human-readable string representation.'

`printf()` allows for the passing of arguments to be printed through format strings. An example would be `printf("I have %d apples, 41);`, which will print `I have 41 apples`. `%d` is known as a format parameter, and there are a few common format parameters used in `printf()`:

Format Parameter	Description	Type
%d	decimal	Integer
%u	unsigned decimal	Unsigned Integer
%x	hexadecimal	Unsigned Integer
%s	string	Reference
%n	*writes the number of characters into a pointer	Reference

What happens if a format parameter is passed into printf, but there are no arguments?

Take for example, `printf("I have %d apples);`. The program requires one argument to be printed in decimal. As no argument is specified, the program will fetch a value from the top of the stack and leak it, printing it out in decimal. As such, we have an ability to leak values from the stack.

Investigation

In the program, after the flag is initialized, a character pointer `*flag_ptr` is created to point to the flag string.

```
char *flag_ptr = flag;
```

This flag pointer will be somewhere on the stack when `printf(buf)` is called. With reference to our Format Parameter Table above, we see that if `%s` is used, the program will take the top value of the stack as a reference pointer and print out the string it is pointing to.

However, just inputting `%s` will not provide us the flag.

```
This cave is rather echo-y
Shout something: %s

zsh: segmentation fault  ./echo_1_static
```

This is because in 64 bit architecture, the **parameters to any function is passed using registers** rather than by pushing it into stack. For linux, parameter passing is done in this order: **RDI, RSI, RDX, RCX, R8, R9, remaining from the stack**.

This implies that the first six values leaked with the format string attack are values from the registers, not the top value off the memory stack.

Hence, in order to obtain the `*flag_ptr` pointer (located at the top of the memory stack) and print out the string it is pointing to, we require to read the 7th 'parameter on the stack' and we can access it with the payload: `%7$s`

As this might be quite confusing, I recommend reading this article - (https://nixhacker.com/case-of-format-string-in-64-bit-is-it-still-critical/) to understand format string attacks on 64-bit binaries.

Payload

```
### Result
```

This cave is rather echo-y Shout something: %7$s

The cave echoes back: CZ4067{b3_c4r3ful_w17h_pr1ntf} "'

Flag

`CZ4067{b3_c4r3ful_w17h_pr1ntf}`