Don't Forget The Lyrics

Problem Type

Variable Stack Overflow

Checksec

I highly recommend running `checksec` when starting on any binary exploitation challenge. `checksec` will allow you to understand what security features are enabled and how you can approach the challenge.

Running `checksec` on this `lyrics` binary, we find that NX and PIE are enabled, while stack canaries are disabled.

```
Arch:     amd64-64-little
RELRO:    Partial RELRO
Stack:    No canary found
NX:       NX enabled
PIE:      PIE enabled
```

Solution

Vulnerability

```
    printf("=========================================\n");
    printf("Never gonna give you up!\n");
    printf("Never gonna let you down!\n");
    printf("Never gonna run around and ______ you!\n");
    printf("> ");

    scanf("%s", locals.buf);
    printf("=========================================\n");
```

`scanf("%s", buf);` is vulnerable to buffer (stack) overflow.

There is no restriction on the length of the string that is read and passed into the buffer. Hence, players can pass in payloads of length larger than that of buffer, and successfully overwrite values on the stack.

> A safer way of using `scanf` to prevent buffer overflow would be to specify the size of the string the program will read in: For example, `scanf("%16s", buf);` will read in 16 characters. One can also use `fgets` instead for program input.

Investigation

In the source code, we note that to obtain the flag, we need to change the `lyric` variable to *'desert'*.

```
if (strcmp(locals.lyric,"desert") == 0){
    printf("Congratulations!\n");
    char flag[512];
    // GET FLAG
    printf("The flag is: %s", flag);
}
```

To achieve this, we need to craft a payload of the correct length. The variables are initalised as such at the top of `main()`:

```
struct {
    char buf[16];
    char lyric[16];
  } locals;
```

This ensures that `lyric` is pushed first onto the stack, followed by `buf`. This implies that overflowing the `buf` variable will result in us having direct control of the value of `lyric` on the stack.

Payload

1. Our payload will first require 16 filler characters to fill up the buffer `buf` - We can use `16 * "A"` (AAAAAAAAAAAAAAAA)
2. Next, we want to overwrite `lyric` to 'desert'.

Final Payload: `16 * "A" + 'desert'` OR `AAAAAAAAAAAAAAAAdesert`

Flag

`CZ4067{b0f_w1ll_n3v3r_l3t_y0u_d0wn}`