CTF Write-up: SilentOracle (HTB)

Category: Reverse Engineering
Difficulty: Medium

1. Challenge Description

Beneath the temple sits a mute sage... Beware though, trying to lie to it will result in temporal banishment.

2. Initial Reconnaissance

We analyze the binary using file and observe its behavior.

```
$ file chall
chall: ELF 64-bit LSB pie executable, x86-64, stripped
```

Running the binary shows a delay on wrong input (~5 seconds). Using ltrace confirms sleep usage.

```
$ echo "AAAA" | ltrace ./chall
sleep(5) = 0
```

3. Static Analysis

Reverse engineering reveals a byte-by-byte comparison with a sleep penalty on mismatch.

```
bool check_flag(input, len) {
    for (int i = 0; i <= 20; i++) {
        if (input[i] != flag[i]) {
            sleep(5);
            return 0;
        }
    }
    return 1;
}
```

4. Vulnerability: Timing Side-Channel

Correct characters proceed instantly. Incorrect characters trigger sleep(5). This leaks information through response time, enabling a timing attack.

5. Exploitation Script

```
from pwn import *
```

```
import time, string

HOST = '83.136.252.32'
PORT = 48150

charset = string.ascii_letters + string.digits + "_!@?}"
flag = "HTB{"

while True:
    for c in charset:
        r = remote(HOST, PORT)
        r.sendline((flag + c).encode())

        start = time.time()
        try:
            r.recvall(timeout=1.5)
            duration = time.time() - start
        except:
            duration = 10

        r.close()

        if duration < 2:
            flag += c
            print(flag)
            if c == '}':
                exit()
```

6. Result

Flag: HTB{Tim1ng_z@_h0ll0w_t3ll5}