Linux2 CTF Writeup

Challenge Overview

The challenge presented a guard who claims to be both strong and smart. The goal was to provide two numbers for division and find a way to "distract" the guard to access the vault.

Analysis

Using Ghidra to decompile the `linux2` binary, the function `gate()` contained the main logic:

- The guard prompts for two numbers.
- If the input is valid, the program attempts to divide the first number by the second.
- The guard checks if the result of the division is within a certain range.
- If the result is too large, the guard is "distracted" and the `vault()` function is called, revealing the flag.

Key Code Analysis

```
local_18 = local_28 / local_20;
if ((double)((ulong)local_18 & 0x7fffffffffffffff) <= 1.797693134862316e+308) {
    printf("Guard: HAHA, I know this one! The result is: %lf\n",local_18);
}
else {
    puts("Guard: Wait.. what... let me think... the guard is distracted and leaves
the door to the vault open.");
    vault();
}
```

The check compares the division result (`local_18`) against `1.797693134862316e+308`, which is the maximum value for a double in C. The goal was to cause the result to exceed this threshold.

Exploitation and Payload Strategy

1. Input a very large number as the first input.
2. Input a very small number (e.g., 1) as the second input to create a large division result.
3. The guard is "distracted" by the overflow, triggering the `vault()` function.

Result and Solution

```
./linux2
1st number: 99999999999999999999999999999999999999 ... (a lot of 9's)
2nd number: 1
```

The guard fails to process the large output, and the `vault()` function is called, displaying the flag:

```
ThisIsTheFlag
```

Conclusion and Lessons

The challenge involved understanding how to manipulate the floating-point operation to exceed the maximum double value, triggering the vulnerability in the program's logic to gain access to the vault and retrieve the flag.

References

- alt text
- alt text