Private Challenge - Format String and GOT Written

> **note:** due to this being a private challenge, some details of the exploit process has been omitted.

Summary

Objective: Launch a shell
Areas of vulnerability: user input directly into `printf`, lack of canaries, no input validation
Recommendations:

- Validate the input (length, values, number of arguments)

Tools used:

- `gdb`

Findings:

1. Format string exploit
   Affects: `write()` function
   Found by disassembling the executable and tracing the `.asm` file, user has control over calls to `printf`
2. Stack-based overflow
   Affects: all functions without a canary
   Found by disassembling the executable and tracing the `.asm` file
   Allows for control flow redirection

Walkthrough

Stack

First, start off with dumping the code into `.asm` form and interacting with the executable to establish a baseline.
Environment restrictions:

- NX and ASLR, so no code injection onto the stack
- Gadget farm is sparse

We will use a GOT exploit to make a call to `system()` to execute a shell by overwriting one of the stored functions to be `system()` instead. With the format string, we can overwrite the value of the GOT table for `printf()`, changing it to the address for `system()`. The input given to the second `printf()` will be interpreted as path to an executable, and spawn shell as desired.