Assignment 3 – Format String

Hritesh Sonawane

Reference(s):  
- <https://owasp.org/www-community/attacks/Format_string_attack>

- <https://cwe.mitre.org/data/definitions/134.html>

- <https://ctf101.org/binary-exploitation/what-is-a-format-string-vulnerability>

- <https://stackoverflow.com/questions/7459630/how-can-a-format-string-vulnerability-be-exploited>

After establishing a connection to the server, I sent the format string “%p” to leak an address from the stack. This address was captured in the server's response and decoded from hexadecimal format, revealing a memory location within the application’s stack.

To ensure the shellcode could execute correctly, I adjusted the leaked address by adding an offset of “0x26C0” to point to the specific buffer location where my shellcode would be placed. The shellcode was crafted to invoke a shell using the “execve” system call, specifically moving the address of the string /bin//sh (represented as “0x68732f2f6e69622f”) into the rdi register. This address was pushed onto the stack to set up the command for execution.

Finally, I constructed the payload by combining the shellcode, NOP padding for alignment, and the adjusted buffer address. Upon sending this payload, I successfully gained an interactive shell to rpint the flag.

Used the GCP web shell as I have a Mac and binary analysis was not possible, even on a VM as we discussed in class too!

A screenshot of a computer

Description automatically generated

Exploit code: (took help from GPT and Faaiza as I couldn’t get the padding right)

A computer screen shot of a program

Description automatically generated

---

For part 2: return to libc:

A computer screen shot of a program

Description automatically generated

I’ll continue working on it, but this is what I understood until now:

To perform a "Return to libc" exploit, I started by identifying the offsets for the system function and the string /bin/sh in libc.so.6 using readelf and strings commands. For example, I found system at 0x0000000000052290 and /bin/sh at 0x1b45bd. However, I'm still working on confirming the correct address for /bin/sh.

Next, I plan to leak an address from the process to determine the base address of libc.

---

**libc\_base = leaked\_address - offset\_to\_known\_function**

Once I have the libc\_base:

**system\_addr = libc\_base + offset\_of\_system**  
**bin\_sh\_addr = libc\_base + offset\_of\_bin\_sh**

The final payload will involve padding to reach the return address and appending the addresses of system and /bin/sh???:

**payload = [padding] + [system\_addr] + [bin\_sh\_addr]**

--- Thank you!