PIE_TIME

PIE TIME

Easy Binary Exploitation picoCTF 2025 browser_webshell_solvable

AUTHOR: DARKRAICG492

Description

Can you try to get the flag? Beware we have PIE!

Connect to the program with netcat:

```
$ nc rescued-float.picoctf.net 52608
```

The program's source code can be downloaded here. The binary can be downloaded here.

This challenge launches an instance on demand.

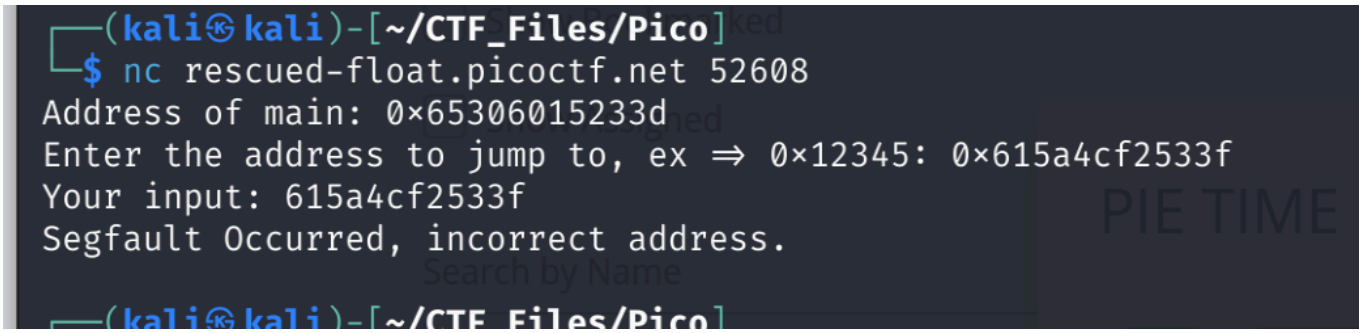
Its current status is: RUNNING

Instance Time Remaining: 10:42

Restart Instance

Hints

1



Played with the code with netcat for a bit and was completely lost....

Was confused about...

- Recalling what a binary ACTUALLY is.
- Whether `flag.txt` is inside the binary or on the server instance.
- What `PIE` and stack protections are.
- How to actually exploit this over the network.

1. Binary vs. Source

- The `.c` file is source code.
- The binary (`vuln`) is the **compiled machine code** that actually runs.
- Binaries don't "contain" flag.txt — they may **read from it at runtime** if told to.

2. Challenge Structure

- The binary runs **on a remote CTF server**.
- Idea is to:
  - Analyze it **locally**.
  - Find the address of a hidden `win()` function.
  - Send that address over `netcat` to the remote binary.
  - If successful, it calls `win()` and prints the flag.

3. The Key Vulnerability

- The binary reads a user-supplied address:

```
scanf("%lx", &val);
((void (*)())val)();
```

- This lets you **hijack execution** by entering the address of `win()` .

4. What Is PIE (Position-Independent Executable)?

- PIE means the binary loads at a **random memory address every run**.
- The function addresses (`main()` , `win()`) shift every time.
- Must calculate the base address at runtime.

5. Exploitation Math

- Get local offsets using `nm vuln` :

```
nm vuln | grep win    # → 00000000000012a7 T win
nm vuln | grep main   # → 000000000000133d T main
```

- On the remote server, you'll see:

```
Address of main: 0x5e28abc1233d
```

- Then do the math:

```
base = remote_main - local_main_offset
win_addr = base + win_offset
```

6. Sending the Exploit

- Once you calculate the `win()` address:

```
echo '0xCALCULATED_WIN_ADDRESS' | nc challenge.site 1234
```

✅ If done in the same session: it prints the flag!

8. Automation with Python (`pwntools`)

- Wrote a Python script to: Refer to Pico file for `vul.py` script)
  - Connect to the challenge.
  - Parse the remote `main()` address.
  - Do the math.
  - Send the exploit.
  - Print the flag.
- Had a problem installing `pwntools`, but learned to fix it via:
  - `python3 -m venv`
  - `pip install pwntools` inside the virtual env

Final Thoughts

- This was a classic **ret2win + PIE** CTF challenge.
- You now know how to:
  - Understand binary layout
  - Work with PIE binaries
  - Use `nm`, `netcat`, and `pwntools`
  - Install Python tools safely in Kali