CS6570 - Secure Systems Engineering: Assignment 4

Guidelines

- **Deadline**: 15th April, 2024
- We expect you to submit a gzipped tar archive., named as `<ROLL_NO>.tar.gz` (example: `CS20B123.tar.gz`)
- You should run `tar cvzf CS20B123.tar.gz CS20B123/`, where you have your submission files inside the directory `CS20B123/`
- The archive should contain a directory (`CS20B123/`), which should contain the following submission files:
  - Python scripts (to be run using `python3`) used to exploit both the challenges
    - To be named `q1.py` and `q2.py` for challenge 1 and challenge 2, respectively
  - Flags obtained for both the challenges
    - To be named `q1.txt` and `q2.txt` for challenge 1 and challenge 2, respectively
  - Report on the techniques used to exploit both the challenges (can either submit a PDF or a markdown/text file)

Files provided

- `sectok` : Statically linked executable, used in challenge 1
  - `sha256sum:2d5a715bc178525e2adc1d4e9f69fb18ee0641ce08fe633af0d997ae48ff6a49`
- `libc.so.6` : Shared library used for linking in both the challenges
  - `sha256sum:acf8ac6d219b657af09328abde95a6a52a3ce7019c6717cade77db5634498866`
  - GLIBC version `2.27`
- `sectok.c` : Source file used in challenge 1
  - Compiled on Ubuntu 18.04 64-bit and statically linked with GLIBC `2.27`
  - Command used: `gcc -static sectok.c -o sectok`
- `sectok_libc.c` Source file used in challenge 2
  - Compiled on Ubuntu 18.04 64-bit and dynamically linked with GLIBC `2.27`
  - Command used: `gcc sectok_libc.c -ldl -o sectok_libc`

Challenges

Challenge 1

- Remote server at IP `10.21.232.3` and port `10101`
- Obtain a remote shell on the server hosting the challenge, and obtain the flag
  - The flag is stored in the `flag` file, you can `cat` it on the server
- You are expected to use a double-free exploit

Challenge 2

- Remote server at IP `10.21.232.3` and port `20202`
- Obtain a remote shell on the server hosting the challenge, and obtain the flag
  - The flag is stored in the `flag` file, you can `cat` it on the server

- You are expected to use a double-free exploit

Testing

- We will be running your exploit script on the testing machine, with the provided `libc.so.6` in the same directory.
- We expect that after running the script, we will have a shell on the remote machine.
- Do not hardcode the flag and print it in the script. We want the script to exploit the remote server.

Notes

- You will only be able to access the servers within IITM network (i.e. iitmwifi or room LAN).
  - In case you want to desperately access these servers from outside IITM, you can reach out to the TAs.
  - If you are unable to access the servers, please let the TAs know.
- Make sure that you are testing the exploit locally with the correct libc version (the double free exploit taught in class works only when using libc version `2.27`).
- In this assignment, you will require communicating with a remote server. This is very common in CTFs and you will need to know how to do this in the final SSE CTF as well.
  - There is a famous Python library designed for it, known as `pwntools`.
  - Use the internet to learn how you can use it. It is extremely powerful.
- You can also check out `pwndbg`, which is a much more powerful debugger built on `gdb`. It has great heap inspections tools.
- If you get stuck, you can also have a look at `how2heap` which contains tutorials on a variety of heap exploits.
- Any sort of flag/exploit sharing or plagiarism will not be tolerated. There will be repercussions.