Assignment 0x03 - Memory Attacks

Start Assignment

Due Sunday by 23:59 **Points** 10 **Submitting** a file upload **File types** pdf
Available 31 Mar at 21:00 - 8 May at 23:59 about 1 month

For this assignment:

- Connect to the HacklabVM
- Get the secret message for each of q1 ~ q6

1. (1 point) Go to /home/q1/. Exploit the program to get the secret.

2. (2 point) Go to/ home/q2/. Exploit the program to get the secret.

3. (2 point) Go to/ home/q3/. Exploit the program to get the secret.

4. (2 point) Go to /home/q4/. Exploit the program to get the secret.

5. (2 point) Go to /home/q5/. Exploit the program to get the secret.

6. (2 point) Go to /home/q6/. Exploit the program to get the secret.

7. (1 point). Firewalls have the capability to block both ingress (inbound) and egress (outbound) traffic. Many organisations (and also true for my home NBN router) block ingress, but is pretty open when it comes to egress rules.

a) Why should organisations care about setting egress (outbound) firewall rules?

b) Lookup "C2 server" on the internet and explain why they can be successful even on firewalls that tightly restrict egress traffic to sanctioned ports like 53, 80 and 443.

8. (Bonuse 2 points) Go to /home/q7/. Exploit the program to get the secret. (You may not get the secret because of server problem, you can just provide process and description for this question and you will get the full mark.)

9. (Bonuse 3 points) Go to /home/q8/. Exploit the program to get the secret.

Hints:

- All programs compiled with -m32 -g -fno-pie -fno-stack-protector
- All pre-compiled programs are SETUID (chmod u+s) and runs as another user (user 'q1' for q1, 'q2' for q2, etc). You can check with with ls -l
  setuid
- Refer to source code for additional hints.
- Q8: Pretty much same as workshop, but you need to find out the address of target using gdb
- Q8 Bonus: combine the technique explained in Q4

Part II

10. (Bonuse 2 points) Return to Libc

- Go to /home/q9, and exploit the pre-compiled program q9 to get the secret. Source code is provided.
- You might need to read the source code to understand what's happening.
- **Hints:**
  - The program expects a filename for argv[1], so the payload needs to be.... in a <redacted>.
  - In performing Step 8 of the workshop, replace

    ```
    x/20s *((char **)environ + 30)
    ```

    with

    ```
    x/20s *((char **)environ)
    ```

    to look for your environmental variable (SH) as it's usually further up
  - If your exploit succeeds in gdb (it should) but fails outside of gdb (as per workshop) you need to adjust the last 4 bytes of the payload carefully... I have installed hexedit on the server (F1 for help).
  - Make sure to run with full path /home/q9/q9 /<full path to payload> outside of gdb to be consistent.
  - The findenv.c program would not work in this case, as the argv[0] length will be different.