**CY 5770 Software Vulnerabilities and Security**

**Midterm CTF-1 and Midterm CTF-2**

**Nov 15, 2024 and Nov 17, 2024**

**Total 160 Points + Bonus Points**

**Please use this file as the template for both CTFs.**

**Your full name:**

**Your CTF platform username:**

**You are allowed to use google, or refer to lecture slides, homework, code during the exam. But, you cannot communicate with anyone in or outside of the class. You cannot use LLM, e.g., ChatGPT, either.**

**For each challenge, you should clearly show (1) your exploit, (2) screenshot of successful exploitation, and (3) a very brief description of how you did it. You also need to submit the flag on the platform. Even if you fail to capture the flag, you may get some points for documenting your steps.**

**Some hints: Use “file” to check if the executable is 32-bit or 64-bit. Use “./checksec --file executable-name” to see if the stack is executable or if stack cookie is inserted. Use “strace” to see what system calls a program makes.**

**All four challenges are under the MidTerm24Fall category.**

**The first hacker to capture each flag receives a 10-point ‘first blood’ bonus.**

1. [40] Capture the flag of ***re\_2\_64***.

a. [5] Where does this program take input?

1. [40] Capture the flag of ***overflowret9\_32***.

a. [4] Is this a 32bit or 64-bit program?

b. [4] Is stack executable? Can you overwrite RET address on stack? Is there a canary to protect the stack? (You can use checksec.sh, objdump, etc., to find out)

c. [4] Where does this program take input?

d. [4] Describe your high-level idea on how to exploit this challenge.

1. [40] Capture the flag of ***overflow7\_32***.

a. [4] Is this a 32bit or 64-bit program?

b. [4] Is stack executable? Can you overwrite RET address on stack? Is there a canary to protect the stack? (You can use checksec.sh, objdump, etc., to find out)

c. [4] Where does this program take input?

d. [4] Describe your high-level idea on how to exploit this challenge.

1. [40] Capture the flag of ***overflowret7\_32***. The source code of this challenge is in /code

a. [4] Is this a 32bit or 64-bit program?

b. [4] Is stack executable? Can you overwrite RET address on stack? Is there a canary to protect the stack? (You can use checksec.sh, objdump, etc., to find out)

c. [4] Where does this program take input?

d. [4] Describe your high-level idea on how to exploit this challenge.