

# Lab 4

# Main Goals

---

- **Analyze** potential format string vulnerabilities in source code
- **Exploit** format string vulnerabilities in different scenarios
- **Gain** a deeper understanding of (some) format string options/modifiers

# Task 1: Inspect the Program

---

- Analyze the provided source code
- Determine the potential format string vulnerability
- Understand the stack layout during a function call

## Tasks 2—3

---

- Task 2: crash the process
- Task 3: read from the heap

# Task 4: Modify a Variable Value

---

Three subtasks

- Write a specific value
- Write a large value
- Write another large value

**How can you write values?**

# Task 4: Modify a Variable Value

---

Challenges of writing **large** values such as `0xff990000`

- The simple approach of using `%n` → You need to print 4,288,217,088 bytes on the screen!
  - Time consuming and inefficient; not allowed for this lab
- Other ideas?

## Task 4: Modify a Variable Value

---

Assume that the variable address is **0x08a0a0a0**

- Divide 0xff990000 into two-byte values, and write:

- 0xff99 to the higher memory address
- 0x0000 to the lower memory address

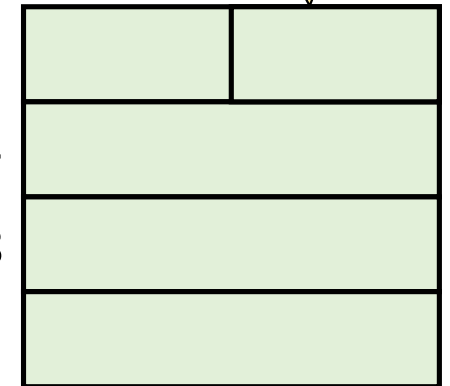
**0x08a0a0a0**

0x08a0a0a4

0x08a0a0a8

0x08a0a0ac

Address is...?



*How can we control the **size** of written values?*

## Task 4: Modify a Variable Value

---

Assume that the variable address is `0x08a0a0a0`

- How can you write `0x0000`?
  - Problem: `printf` has already written a number of bytes



# Questions?

---