CSC 583 Fall 2024 Lab 3

Dr. Si Chen

November 14, 2024

Introduction

Lab Objectives:

- Understand the principles and techniques of Return-oriented Programming (ROP) in ARM architecture.
- Develop skills to craft and deploy ROP chains using Python scripts.

Upon completing this lab, students should be able to demonstrate:

1. A clear understanding of ROP and its application in bypassing security mechanisms like non-executable stacks.
2. Proficiency in writing Python scripts to create and send ROP chains to vulnerable ARM-based applications.
3. The ability to analyze ARM binaries to identify useful gadgets and construct a working ROP exploit.

Students are expected to apply this knowledge practically to perform a successful ROP attack against a controlled ARM environment.

Course Webpage: https://www.cs.wcupa.edu/schen/advsec24/

Return-Oriented Programming Exercise

In this lab, you will be working with an ARM-based application (‘lab4‘) that is vulnerable to a buffer overflow. Unlike traditional buffer overflows, where shellcode is injected, this lab focuses on ROP, a more advanced technique that circumvents non-executable stack protections.

Steps:

1. Use your credentials to log into the BadgerCTF environment provided.

2. Fetch the vulnerable ARM binary from the course shared folder and save it in the directory '/workdir/'. To do this after logging in, run:

```
cp /workdir/advsec24/lab3/lab3 /workdir/
```

3. Analyze 'lab4.c' to understand its structure and identify potential buffer overflow vulnerabilities.
4. Write a Python script using the pwntools library to create a ROP chain. Your script should aim to execute a command (e.g., spawning a shell) by leveraging existing code (gadgets) within the binary or libraries.
5. Utilize tools like GDB-multiarch and objdump to assist in finding useful gadgets and constructing your ROP chain.
6. Test your exploit to ensure it successfully manipulates the program's flow to achieve the desired outcome.
7. Document your process, including how you identified gadgets, constructed the ROP chain, and overcame challenges.
8. Conduct this exercise with ethical responsibility.

Hints and Guidance:

- Start by finding the offset at which the buffer overflow occurs. This will be crucial for aligning your ROP gadgets correctly.
- Look for gadgets that allow you to control registers like `PC` (Program Counter) and `R0-R3`, which are often used for function arguments in ARM.
- Remember to align your stack correctly. ARM architecture often requires 8-byte alignment.
- Use `pwntools` for crafting payloads and interacting with the binary. Functions like `p32()` can be used to convert addresses to the correct format.

Deliverables: A comprehensive lab report is required. It should include the source code of your exploit (named 'rop_exploit.py'), a screenshot showing the successful execution of your exploit, and a detailed methodology describing how you identified gadgets, constructed the ROP chain, and any challenges you faced during the process.