CSC 600 Spring 2025 Lab 3: Analyzing Buffer Overflow Vulnerability CVE-2008-4250 in NETAPI32.DLL

Dr. Si Chen

1 Introduction

In the previous lab, we explored the CVE-2006-3439 vulnerability in the NETAPI32.DLL file and examined the effectiveness of Microsoft's patch. In this lab, we will focus on another vulnerability, CVE-2008-4250, which also affects the NETAPI32.DLL file. This vulnerability is caused by a developer's negligence and lack of rigor in string movement operations, failing to strictly check for out-of-bounds situations.

Through static analysis of the assembly code, we will identify the root cause of the vulnerability in the `NetpwPathCanonicalize()` function and understand how improper bounds checking can lead to buffer overflow. Students will use IDA Free to examine the disassembled code and trace the execution flow to pinpoint the vulnerable code segment.

By the end of this lab, students will gain a deep understanding of the CVE-2008-4250 vulnerability, the importance of proper bounds checking, and the potential consequences of buffer overflow vulnerabilities. They will also develop skills in static code analysis and vulnerability assessment.

2 Objectives

- Understand the CVE-2008-4250 vulnerability in NETAPI32.DLL.
- Analyze the disassembled code using IDA Free to identify the vulnerable function.
- Trace the execution flow and pinpoint the root cause of the vulnerability.
- Understand the importance of proper bounds checking in preventing buffer overflow.

3 Tools Required

- IDA Free

4 Part 1: Setting Up the Environment

1. Ensure you have IDA Free installed on your system.
2. Obtain the vulnerable NETAPI32.DLL file affected by CVE-2008-4250.
3. Read the static analysis report on CVE-2008-4250 provided by the instructor.

5 Part 2: Analyzing the Vulnerable Code

5.1 Loading NETAPI32.DLL in IDA Free

Open the vulnerable NETAPI32.DLL file in IDA Free and wait for the initial analysis to complete.

Task 1: Locate the `NetpwPathCanonicalize()` function in the disassembled code. Identify the call to the vulnerable `sub_5FDDA180` function.

5.2 Identifying the Vulnerability

Analyze the `sub_5FDDA180` function and focus on the string concatenation operation using the `wcscat()` function.

Task 2: Explain how the lack of proper bounds checking in the `sub_5FDDA26B` function can lead to buffer overflow when simplifying dot-dot ".." sequences in the path.

5.3 Tracing the Execution Flow

Study the code flow and understand the conditions required to trigger the vulnerability.

Task 3: Describe the exploitation conditions of the CVE-2008-4250 vulnerability. What specific path simplification scenarios are needed to activate the vulnerability and overwrite the return address?

6 Part 3: Implications and Mitigation

6.1 Consequences of the Vulnerability

Discuss the potential impact of the CVE-2008-4250 vulnerability if exploited by an attacker.

Task 4: Explain how an attacker could leverage the buffer overflow vulnerability to execute arbitrary code or gain unauthorized access to the system.

6.2 Mitigation Strategies

Propose measures to prevent similar vulnerabilities in software development.

Task 5: Discuss the importance of proper bounds checking and suggest secure coding practices to mitigate the risk of buffer overflow vulnerabilities.

7 Conclusion

Summarize your findings and the key lessons learned from analyzing the CVE-2008-4250 vulnerability. Emphasize the significance of thorough code analysis and secure coding practices in preventing buffer overflow vulnerabilities.

8 Submission

Submit a detailed report containing your answers to the tasks, along with relevant screenshots and explanations. Ensure that your report is well-structured and properly formatted.