CSC 583 Spring 2025
Lab 2: Analyzing Buffer Overflow Vulnerability in NETAPI32.DLL

Dr. Si Chen

1 Introduction

In the previous analysis, we gained some understanding of the CVE-2006-3439 vulnerability in the NETAPI32.DLL file. After discovering this vulnerability, Microsoft modified the code and released a patch to address the issue. However, the question remains: *does patching the vulnerability necessarily mean that the affected function is now completely secure?*

In this lab, we will delve deeper into the patched version of NETAPI32.DLL and examine the changes made by Microsoft. By comparing the patched version with the original vulnerable code and employing reverse engineering techniques, we aim to determine whether the patch effectively eliminates all potential security risks associated with the function in question.

Students will use IDA Free, a powerful disassembler and debugger, to study the assembly code of the patched NETAPI32.DLL file. Through this analysis, they will gain insights into the modifications made by Microsoft and assess the effectiveness of the patch in mitigating the buffer overflow vulnerability.

Furthermore, students will explore the possibility of discovering new vulnerabilities or exploitation methods that may still exist in the patched version of the code. By understanding the root cause of the original vulnerability and applying their knowledge of buffer overflow techniques, students will develop the skills necessary to identify and analyze potential security weaknesses in software.

This lab exercise serves as an excellent opportunity for students to enhance their reverse engineering skills, deepen their understanding of buffer overflow vulnerabilities, and develop a critical mindset when evaluating the security of software patches. By the end of this lab, students will have gained valuable experience in analyzing complex assembly code and assessing the effectiveness of security patches.

2 Objectives

- Understand the concept of buffer overflow vulnerabilities.
- Analyze assembly code using IDA Free.
- Identify the root cause of the vulnerability in NETAPI32.DLL.
- Explore potential exploitation techniques.

3 Tools Required

- IDA Free
- OllyDBG (optional)

4 Part 1: Setting Up the Environment

1. Download and install IDA Free on your system.
2. Obtain the vulnerable NETAPI32.DLL file from our website.
3. Read this report (part II) [Link]

5 Part 2: Analyzing the Vulnerable Code

5.1 Loading NETAPI32.DLL in IDA Free

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

Task 1: Locate the `NetpwPathCanonicalize()` function in the disassembled code. Describe the purpose of this function based on your understanding of the code.

5.2 Identifying the Vulnerability

Refer to the learning materials and focus on the code snippet that performs string concatenation using the `wcscat()` function.

Task 2: Explain why using the `wcscat()` function for string concatenation can lead to buffer overflow vulnerabilities.

5.3 Analyzing the Execution Flow

Study the code flow and the conditions under which the vulnerability can be triggered.

Task 3: Describe a scenario where the `NetpwPathCanonicalize()` function can be called twice consecutively, potentially leading to buffer overflow. What specific conditions need to be met for this to occur?

6 Part 3: Exploiting the Vulnerability

6.1 Crafting an Exploit

Based on your understanding of the vulnerability, devise a method to exploit it and achieve arbitrary code execution.

Task 4: Outline the steps required to exploit the buffer overflow vulnerability in NETAPI32.DLL. Explain how you would manipulate the input to the `NetpwPathCanonicalize()` function to trigger the vulnerability and gain control of the execution flow.

6.2 Mitigating the Vulnerability

Discuss potential mitigation strategies to prevent the exploitation of this vulnerability.

Task 5: Propose two methods to mitigate the risk of buffer overflow in the `NetpwPathCanonicalize()` function. Justify your recommendations.

7 Conclusion

Summarize your findings and the key takeaways from this lab exercise. Reflect on the importance of secure coding practices and the potential impact of buffer overflow vulnerabilities.

8 Submission

Submit a detailed report containing your answers to the tasks, along with relevant code snippets and screenshots on D2L. Ensure that your report is well-structured and properly formatted.

Note: Follow academic integrity guidelines and provide proper attributions for any external resources used.