Threat Modelling

Image Generated by Dalle-E 2 from chat gpt

Prepared for

Kokub Sultan

Prepared by **Group 6**

Valentine Jingwa, Jean-Pierre Nde-Forgwang

Security System, Cohort E

School of Advance Digital Technology

SAIT

17 January 2024

Introduction

Overflow, also known as buffer overflow, is a specific type of vulnerability that occurs in software applications. This vulnerability is not just a theoretical concern but a practical issue that has been the root cause of numerous significant security breaches over the years. To understand buffer overflow, it is essential to grasp some fundamental concepts about how computer programs handle data and memory.

What is a Buffer

In computing, a buffer is essentially a temporary storage area in a program's memory. Buffers are crucial in managing the flow of data between processes or between a process and an input/output device. They ensure that data can be handled efficiently, even when the incoming rate is sporadic or faster than the process can manage [2].

Vulnerability Overview

Stack-based Buffer Overflows

A software vulnerability known as a stack-based buffer overflow occurs when a program writes more data into a buffer, which is a temporary data storage space, situated on the stack. The stack is a crucial portion of memory used by programs to manage function calls and local variables. The excess data overflows into neighboring memory locations on the stack when the buffer overflows. Return addresses and other crucial control data may be overwritten as a result. This overflow can have a variety of effects, such as making the software act erratically or crash or giving attackers the chance to run malicious code by editing the overwritten sections. A simple explanation is as follows; imagine you're playing with a set of 15 toy blocks, and you have a challenge to stack them as high as you can. Your stack of blocks is like a special shelf in a computer where it keeps important information temporarily. Now, your shelf can only hold exactly 15 blocks. If you try to add a 16th block to the top, there's no room, and the extra block might fall off and knock over a toy car (representing another part of the computer's memory) parked next to your stack. This is like a stack-based buffer overflow, where putting too much information in one place can cause problems in other areas. [3]

Example -

An actual instance of a stack-based buffer overflow happened in 1988 with the "Morris Worm" incident. This was among the earliest known instances of a buffer overflow being used as a hacking tool. [4]

Heap-based Buffer Overflows

These occur in the heap, a dynamically allocated memory area. Consider a sandbox where kids can put toys anywhere they like. If one child takes too much space with their toys, it might spill into another child's area. This is similar to a heap overflow where data goes beyond its designated space.

Example-

An example is the 2003 exploitation of Microsoft's Windows Server Service. Attackers exploited a heap-based buffer overflow to execute arbitrary code, leading to widespread system compromises.[7]

Format String Attack

This involves exploiting the format string parameters in functions like sprintf or printf. An attacker provides a format string that contains malicious code, leading to unauthorized data access. Think of a coloring book where you're supposed to color within the lines. But if you start drawing your own lines, you change the picture. This is like a format string attack, where changing the expected format changes what the program does.

Example

A notable example is the vulnerabilities in the WU-FTPD software, where format string bugs allowed attackers to execute arbitrary code on the server.[6]

Exploitation and Impact Analysis

Exploiters utilize buffer overflow vulnerabilities to inject malicious code into a system's memory. This unauthorized code, once executed, can compromise the system's integrity. It can lead to various security breaches, including unauthorized access to sensitive data, system control, and privilege escalation. The impacts range from data theft and service disruption to complete system takeover. The severity depends on the attacker's intent and the system's role, potentially leading to widespread security crises if critical infrastructure or sensitive data is involved.

Mitigation Strategies for Buffer Overflow

Code Analysis: Regularly review and analyze code for buffer overflow vulnerabilities.

Safe Programming Practices: Use programming techniques that inherently avoid buffer overflows, like bounds checking.

Compiler Security Features: Utilize compilers with built-in security features that detect and prevent buffer overflows.

Memory-safe Languages: Use languages that manage memory automatically, like Java or Python, reducing the risk of buffer overflows.

Regular Updates and Patches: Keep software updated to ensure that vulnerabilities, including buffer overflows, are patched.

References

1. Fortinet. "Buffer Overflow." Accessed January 17, 2024. https://www.fortinet.com/resources/cyberglossary/buffer-overflow (accessed Jan. 17, 2024)

2. Wikipedia contributors. "Buffer Overflow." Wikipedia, The Free Encyclopedia. Last modified January 12, 2024. https://en.wikipedia.org/wiki/Buffer_overflow (accessed Jan. 17, 2024)

3. Wikipedia contributors. "Stack Buffer Overflow." Wikipedia, The Free Encyclopedia. Last modified January 5, 2024. https://en.wikipedia.org/wiki/Stack_buffer_overflow (accessed Jan. 17, 2024)

4. Wikipedia contributors. "Morris Worm." Wikipedia, The Free Encyclopedia. Last modified December 15, 2023. https://en.wikipedia.org/wiki/Morris_worm(accessed Jan. 17, 2024).

5. CERT Coordination Center, "VU#639760 - WU-FTPD configured to use RFC 931 authentication running in debug mode contains format string vulnerability," https://www.kb.cert.org/vuls/id/639760/ (accessed Jan. 17, 2024).

6. CERT Coordination Center, "VU#483492 - Microsoft Windows RPCSS Service contains heap overflow in DCOM activation routines," https://www.kb.cert.org/vuls/id/483492 (accessed Jan. 17, 2024).