جامعة الأمير سلطان
PRINCE SULTAN
UNIVERSITY

CYS405 - Penetration Testing and Ethical Hacking

Buffer Overflow Vulnerability-lab

Department of Computer & Information Sciences

CYS405 Project deliverable

Feb 28th, 2024

Instructor: Dr. Anees

Students:

Tala Almayouf 218410276

Haya Alsaid 219410464

Sarah Aljurbua 220410528

PHASE 1 - Project Proposal

Table of Contents

Phase 1

1.1. Background

Buffer overflow vulnerability is a shortcoming in a program that is exploited by attempting to write more data to a buffer (temporary volatile storage that is used to hold data while being moved) than it is designed to hold. This excess data in the buffer can overwrite adjacent memory, leading to unpredictable behavior such as the execution of malicious codes, system crashes, or data corruption. The goal of buffer overflow is to uncover and exploit these vulnerabilities to demonstrate potential security breaches. This practice helps organizations identify and fix weaknesses, which in turn enhances their security posture. Buffer overflows can facilitate and lead to denial of service (DoS) attacks by making systems unresponsive, but their implications are much broader, such as potentially allowing attackers to gain unauthorized access or control.

1.2. Literature review

1.2.1. Buffer Overflows: Attacks and Defenses for the Vulnerability of the Decade

OGI-IEEE: The research paper "Buffer Overflows: Attacks and Defenses for the Vulnerability of the Decade" presents a comprehensive examination of buffer overflow vulnerabilities, a predominant security concern for the past decade. The authors categorize various buffer overflow attacks and defenses, including their novel StackGuard method, which has shown high effectiveness in mitigating these vulnerabilities without compromising system performance or compatibility. The StackGuard method is a defense mechanism designed to prevent buffer overflow attacks by placing a canary, which is a known value, at strategic points in the memory stack, typically just before the return address. The idea is that a buffer overflow that attempts to overwrite the return address will also overwrite the canary. By checking the integrity of the canary value before executing a return instruction, the system can detect an attempted buffer overflow. If the canary value has been altered, the system assumes a buffer overflow attack is being attempted and can take appropriate action, such as terminating the affected program, thus preventing the attacker from executing arbitrary code. According to Cowan et al., 1999, StackGuard is described as a "compiler technique for providing code pointer integrity checking to the return address in function activation records,". The paper highlights that buffer overflows have been central to remote network penetration attacks, where attackers aim to gain unauthorized control over a system. It discusses the mechanics of buffer overflow attacks, including code injection and control flow corruption, and evaluates different defensive strategies like writing secure code, employing non-executable buffers, and using array bounds and code pointer integrity checking. The research underscores the significance of a combined approach

employing StackGuard alongside other methods to significantly reduce buffer overflow threats. Additionally, it acknowledges the limitations of current defenses against sophisticated attacks that manipulate non-pointer variables or employ social engineering techniques.

1.2.2. Detecting Return-to-libc Buffer Overflow Attacks Using Network Intrusion Detection Systems

The article was entitled "Detecting Return-to-libc Buffer Overflow Attacks Using Network Intrusion Detection Systems". The authors investigate the use of network intrusion detection systems (NIDS) for the detection of return-to-libc buffer overflow attacks. The work deals with the detection of return-to-libc attacks, a particular kind of buffer overflow attack. These attacks redirect the program's execution to a different function (often from the C library) by overflowing a buffer and overwriting the return address on the stack without inserting any malicious code. By using this method, attackers can get beyond security measures that might be in place to stop code injection. The authors suggest using NIDS to find return-to-libc attacks. They make use of the fact that certain patterns of system library function calls are involved in these assaults, which differ from how programs are typically executed. The suggested method of detection centers on tracking the function calls that payloads in network traffic make and contrasting them with a pre-established list of known safe sequences.
The study draws attention to the difficulties in identifying return-to-libc attacks because of the variability of network payloads and the requirement for effective detection techniques that reduce false positives. Using a testbed, the authors assess their detection methodology and show that it can effectively detect return-to-libc attacks with a low false positive rate. Intrusion detection, buffer overflow, protection, computer networks, computer worms, payloads, computer hacking, functional programming, computer bugs, and the internet are some of the keywords linked to the paper. The main ideas and fields of study covered in the paper are reflected in these keywords. In conclusion, this study describes a technique for employing network intrusion detection systems to identify return-to-libc buffer overflow attacks. The suggested method focuses on finding harmful patterns by dissecting the network payloads' function call sequence. The evaluation's findings show how well the detection strategy detects these types of threats while reducing false positives.

1.2.3. Buffer-Overflow Protection: The Theory

The paper thoroughly explores the widespread threat posed by buffer overflow attacks in computer systems, emphasizing the urgent need for strong defenses. It introduces two new tools, TIED and LibsafePlus, which work together to provide a comprehensive solution for detecting and preventing such attacks at runtime. TIED extracts essential debugging information from program binaries and combines it with buffer size data to enable LibsafePlus to perform size checks before executing risky C library functions. This combined approach is compatible with existing C code and imposes minimal runtime overhead, making it practical for real-world use. Additionally, the paper discusses the implementation

details and performance analysis of the tools, demonstrating their effectiveness in enhancing system security. By placing this work within the context of existing research, the paper establishes TIED and LibsafePlus as innovative tools for bolstering computer system resilience against buffer overflow attacks. Throughout the project timeline, team collaboration spans various stages, including planning, development, testing, documentation, and feedback sessions, ensuring continuous progress and addressing challenges. In summary, the proposed solution offers a robust and efficient approach to addressing buffer overflow vulnerabilities, thereby significantly enhancing system security.

1.3. Problem definition

Buffer overflow vulnerability poses a significant cybersecurity threat by allowing attackers to execute arbitrary or malicious payload code, which ultimately leads to gain of unauthorized access, or causes system crashes. Despite the criticality of this vulnerability, detecting and mitigating them remains challenging due to the complexity of software systems and the sophistication of exploitation techniques. This project aims to enhance the effectiveness of penetration testing and ethical hacking in identifying and addressing buffer overflow vulnerability due to mitigation and detection being one of the primary challenges. By discussing possible tools used to mitigate or solve an already occurring buffer overflow attack. By advancing the knowledge and tools available for combating buffer overflow vulnerabilities, this project contributes to the development of more secure software systems and a safer cyberspace for users and businesses.

1.4. Tools to use

The tools we have mentioned below assist in buffer overflow vulnerability discovery, analysis, and understanding, allowing developers and security professionals to implement appropriate fixes and defenses:

1. GDB (GNU Debugger): GDB is a powerful debugger that allows you to analyze programs and examine their memory during runtime. It can be used to identify buffer overflow vulnerabilities, trace program execution, and investigate crashes.

2. Immunity Debugger: Immunity Debugger is a popular debugger for analyzing and exploiting software vulnerabilities. It provides advanced features like scriptable automation, exploit development capabilities, and a graphical interface for easily analyzing memory and registers.

3. WinDbg: WinDbg is a debugger provided by Microsoft for Windows applications. It is commonly used in the Windows environment for analyzing and debugging software vulnerabilities, including buffer overflows. WinDbg offers various commands and extensions to aid in vulnerability analysis.

4. Valgrind: Valgrind is a dynamic analysis tool that helps detect memory errors, including buffer overflows, in C and C++ programs. It can detect, report, and track illegal memory access and provide detailed information about the problematic code.

5. AFL (American Fuzzy Lop): AFL is a popular fuzzer that uses genetic algorithms to generate test cases that can trigger buffer overflow vulnerabilities. It is widely used for vulnerability discovery and can help identify potential buffer overflow points in a program.

6. Metasploit Framework: Metasploit is a powerful penetration testing framework that includes a wide range of exploits and payloads. It can be used to exploit buffer overflows in various software applications, making it a valuable tool for vulnerability testing and analysis.

1.5. Common Terminologies

- Buffer: A temporary storage area used to hold data while it is being transferred from one location to another.

-Stack Overflow: A specific type of buffer overflow that occurs in the call stack of a program.

- Overflow Payload: The data sent by an attacker to exploit a buffer overflow vulnerability, often containing malicious code to be executed.

- Payload: Part of malware that performs the intended malicious action after the exploit has successfully breached the security of the computer system.

- Shellcode: A small piece of code used as the payload in the exploitation of a software vulnerability, typically to open a shell on the target system.

- Stack Guard: A generic term for various protection mechanisms designed to prevent stack overflows, including canaries and other stack integrity checks.

- Exploit: A piece of software, a chunk of data, or a sequence of commands that takes advantage of a bug, glitch, or vulnerability to cause unintended or unanticipated behavior to occur on computer software or hardware.

- Vulnerability: A flaw or weakness in hardware or organizational processes that can be exploited by attackers to gain unauthorized access.

- Denial of Service (DoS): A type of cyber attack aimed at making a computer, network, or service unavailable to its intended users.

- Penetration testing: Also known as pen testing or ethical hacking, is a simulated cyber attack against a computer system, network, or web application to identify vulnerabilities and security weaknesses that a malicious attacker could exploit.

1.6. References

- *Buffer overflows: attacks and defenses for the vulnerability of the decade*. (2000). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/821514
- Tools for Runtime Buffer Overflow Protection— Technical Paper.” [Online]. Available: https://www.usenix.org/legacy/event/sec04/tech/full_papers/avijit/avijit_html/
- *Detecting Return-to-libc Buffer Overflow Attacks Using Network Intrusion Detection Systems*. (2010). Retrieved March 1, 2024, from https://ieeexplore.ieee.org/document/5432802

1.7. Work distribution

Names	Work Distribution
Sarah Aljurbua	Background, 1 literature review, problem definition, termonologies, references
Tala Almayouf	1 literature review, tools used, references
Haya Alsaid	1 literature review, tools used, references