Buffer Overflow Attacks

Indrajeet Patwardhan
Department of Computer Science
California State University, Fullerton
Fullerton, United States
Indrajeet2002@csu.fullerton.edu

Amritpaul Sidhu
Department of Computer Science
California State University, Fullerton
Fullerton, United States
asidhu2001@csu.fullerton.edu

Abstract—**Buffer overflow attacks occur when attackers manipulate the buffer overflow error to compromise systems by exposing their data or damaging their current files. The risk of buffer overflow varies for different programming languages, so certain operating systems are more at risk of getting attacked than others. Since there are different ways in which buffer overflow errors can occur, there are also different ways in which attacks can be used on systems. These attacks can cause serious damage to their targets and have occurred recently in well-known applications, so steps must be taken to ensure that the probability of the attacks happening and their consequences is minimized.**

Keywords—buffer, buffer overflow, memory, vulnerability, exploit, stack, heap

I. Introduction

Since the topic of this paper is about buffer overflow attacks, it automatically falls into the realm of cybersecurity. Cybersecurity and security in computers in general revolves a lot around fixing viruses and preventing attacks. It is the practice of protecting critical systems and sensitive information from digital offense. It is also known as information technology security, its measures are developed to defend from threats against networked systems and applications, whether they originate from inside or outside of an organization. It has layers of protection to defend against all sorts of cyber attacks, which include buffer overflow attacks.[10]

II. What Is a Buffer

A data buffer is a section of physical memory storage that temporarily stores data while it is in the midst of transportation from one place to another. Buffers usually live in RAM memory and are utilized by computers to improve overall performance. They are designed to contain a specific amount of data. Programs that use buffers will typically have built-in measures to discard any data when it becomes too large for the buffer so that it doesn't overwrite any neighboring data.[7]

III. What is Buffer Overflow

Buffer overflow is a software coding error or vulnerability that allows hackers to gain unauthorized access to targeted systems. It is an anomaly that occurs when software writing data to a buffer overflows its capacity resulting in adjacent memory locations being overwritten. In simpler terms, It is when too much information gets passed into a container that does not have enough space to hold all of it, and then ends up dumping the excess information into an adjacent container and replacing its data contents. While it is one of the best-known software security vulnerabilities, it is still fairly common since there are different ways buffer overflows can occur and the techniques used to prevent them are often prone to error. A buffer overflow vulnerability usually occurs when code depends on external data to control its behavior, depends on data variables that are enforced outside of its direct scope, or is so complex that its behavior cannot be accurately determined by programmers.[9]

IV. Buffer Overflow Exploits

The techniques that hackers use to attack buffer overflows highly depend on the architecture and operating system that is used by their target. However, sometimes the extra data that gets dished out to a program contains malicious code that allows the hacker to trigger new actions and instructions to the application. These techniques are called buffer overflow exploits. Buffer overflows are exploited by hackers with the goal of altering a computer's memory in order for them to be in charge of program execution. An example of a buffer overflow exploit is adding additional code into a program where it can send it new instructions that give an opening for the attacker to access all of the systems belonging to the organization.[9]

V. Languages Most At Risk

C and C++ are the most at risk to buffer overflow attacks because they lack safeguards built in to prevent accessing or overwriting data in memory. This puts Windows, Mac OSX and Linux all at risk for attacks since they are written in C and C++. On the other hand,

Identify applicable funding agency here. If none, delete this text box.

languages such as Java, Javascript, C# and PERL have safety mechanisms built in to make buffer overflow attacks less likely.[2] The following compares Python, which is type-safe, to C, which is not type-safe.

Python code that prints a string:

```
>>> mystring="This is my string"
>>> print mystring
This is my string
```

C code which performs the same function:

```
char mystring[20]="This is my string";
printf("%s", mystring);
```

In C the programmer is required to specify the size of the string variable, but that is not the case in Python. If the programmer accidentally attempts to store 40 bytes but has only allocated 20 bytes of memory, then this will cause a buffer overflow. This explains why C, C++ and Assembly are at higher risk of buffer overflow attacks than Python and other languages. [1]

VI. Consequences/Outcomes

Attackers will use buffer overflow to corrupt a web application's execution stack and execute arbitrary code that will take over a machine. On that note, flaws in buffer overflows can exist in both web servers and application servers, and more especially in web applications.[9]

- Data loss - Buffer overflows can be used to steal data by altering the behavior of applications and redirecting outputs.

- Service disruption - Applications can crash as a result of the overflow error, which can bring down critical services and make the system vulnerable to further attacks.

- Code execution - Buffer overflow can allow attackers to execute code using the target system for their gain at the expense of the system.

- Unauthorized access - Attackers can obtain unauthorized access to a system's computing architecture, which means they can compromise the system in numerous ways.

Overall, the common result of a buffer overflow attack is a system crash, access control loss, and causing issues in many aspects of security services. This would especially be harmful for major banking or data managing systems since sensitive financial information would fall into the wrong hands. [3]

VII. Types of Buffer Overflow Attacks

Like it was stated in the beginning, there are many ways that buffer overflow can occur meaning there are also a number of buffer overflow attacks that attackers use to exploit systems.

These types include:

- Stack based buffer overflows - Stack-based buffer overflows are the most common type of buffer overflow attack. It occurs when an attacker sends data containing malicious code to an application that will store the data in a stack buffer. This will overwrite the data on the stack, including the return pointer which then hands full control of transfers over to the attacker. [9]

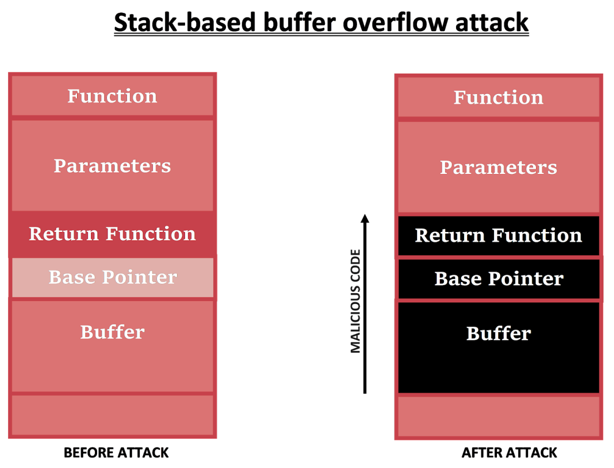
- Heap based buffer overflows - The Heap-based approach is more difficult to carry out than the stack based approach. This is because it involves the attack having to cause the program's memory space to surge beyond the memory it was using for its current runtime operations.[9]

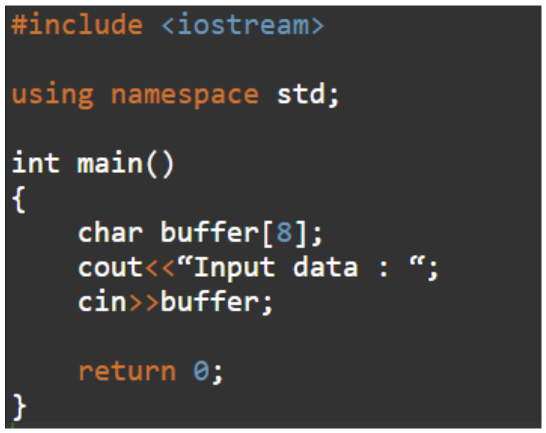
- Format string attacks - The format string attack is a format string exploit that takes place when an application processes input data as a command or does not validate input data efficiently. This will allow the attacker to execute code, read the stack, and or cause segmentation faults in the application. This could trigger new actions that harm the security of the system.[9]

- Integer overflow attacks - Integer overflow is an arithmetic overflow error in which the result of an integer operation does not rest in the allocated memory. It will usually result in an unexpected outcome. In many programming languages, integer values usually have a specific amount of bits in memory. For example, the space reserved for a 32 - bit integer data type may be an unsigned integer between 0 and 4,294,967,295, or a signed integer between −2,147,483,648 and 2,147,483,647. However, what happens when you calculate 4,294,967,295 + 1 and try to store a result that surpasses the maximum value for the data type? Since most languages and compilers make no mistakes and only perform a modulo, accept, or truncate operation, it will only crash and put you at risk of an attack. Almost all integer overflow conditions simply lead to erroneous program behavior and do not cause any vulnerabilities. However, in some instances, integer overflows may have dire consequences like changing financial calculation.[7]

- Unicode overflow - Unicode strings were created to ensure that every language from all countries could be used without transcription issues. For example, Arabic characters are different from English characters. It was realized that such characters could not be converted according to the ASCII codes that we were accustomed to. Therefore, any character can be used with Unicode strings. Due to this, the Unicode schema enables the user to take advantage of the program by typing Unicode characters in an input that expects ASCII characters. It would simply provide an input that surpasses the maximum limit to make a buffer overflow with uncertain characters of Unicode where the program is expecting an ASCII input. [7]

VIII. Stack Vs Heap Attacks

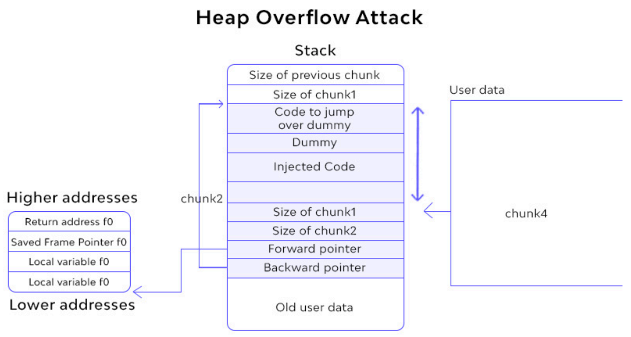




```
#include <iostream>

using namespace std;

int main()
{
    char buffer[8];
    cout<<"Input data : ";
    cin>>buffer;

    return 0;
}
```

[4]

The code above is a simple C++ program that demonstrates stack based overflow. It uses a char type variable and creates an array that will store up to 8 bytes of data. The program will wait for a user input when it is executed. Once the input is provided, it stores the value in the allocated buffer of 8 bytes. If the data provided is greater than 8 bytes then it overwrites the neighboring memory locations resulting in a termination of the process. To test it out, this program can be executed in any C++ compiler, including online compilers. A sample run with an input of 8 or less bytes, lets say "computer," will run the program and exit with exit code 0. Exit code 0 is how the program will exit normally and refers to a successful execution of the process. However, if we ran the program again and provided an input greater than 8 bytes, "many computers," then it would output a message that says stack smashing detected and exit with exit code 134. Exit code 134 means that the program was aborted, maybe due to a failed assertion. That stack smashing detected error is a defense mechanism generated by the compile to fend against buffer overflows. The buffer was designed to hold only 8 bytes of memory but was supplied 13 instead. Those extra 5 bytes went over the buffer boundary and overwrote adjacent memory locations that were on the stack. This resulted in a segmentation fault with a stack smashing error.[4]



The heap based overflow attack occurs when a piece of memory is allocated to the heap and the data being written to that memory is not being checked. This will more than likely result in critical structures on the heap, such as heap-header or any heap based data like dynamic pointers. These can then overwrite the virtual function table that is pictured above.[7]

IX. Recent Occurrences

• NVIDIA SHIELD TV was open to be attacked, thanks to two vulnerabilities, one of which is a buffer overflow bug in devices running software versions prior to 8.0.1. NVIDIA launched an update and asked users to patch their systems ASAP since their last software update. Issues addressed included information disclosure, denial of service, code execution, or escalation of privileges. The buffer overflow bug is designated as CVE-2019-5699. According to a security news bulletin by NVIDIA themselves, "NVIDIA Tegra bootloader contains a vulnerability where the software performs an incorrect bounds check, which may lead to buffer overflow resulting in escalation of privileges and code execution."[5]

• macOS Catalina 10.15 fixed a number of vulnerabilities including a buffer overflow bug. According to ZDNet, the code execution flaw that was detected and resolved was identified as CVE-2019-8745, a buffer overflow that was traced back to the UIFoundation component which could be triggered through malicious text files.[5]

• UIFoundation is a part of UIKit, the graphical interface framework that provides the necessary framework for iOS or tvOS apps. This would mean that it is a core part of the operating system which in turn would mean that a bug here poses huge risks to macOS users.[5]

• WhatsApp, probably the most popular instant messenger, also had some of its own vulnerabilities. Though the end-to-end encryption attracts more users day by day, WhatsApp wasn't as safe and secure as it's made out to be. A critical bug was found during May, 2019 in WhatsApp VoIP, this feature was responsible for audio and video calls. The vulnerability reported was a buffer overflow bug, which allowed an attacker to take over a mobile device. An attacker can call a target and within minutes of the call, it does not matter whether the call was answered or not. The phone would start revealing its encrypted content, displayed to a computer screen halfway across the world. It then would transmit back the most sensitive details such as private messages, location, could even turn on the camera, and microphone to live-stream meetings.[5]

• Exim, a mail transfer agent developed at the University of Cambridge ,gives life to more than half of the world's public mail servers. In September, a severe buffer overflow bug was discovered and identified as CVE-2019-16928, a heap-based buffer overflow bug in its string_vformat (string.c). The vulnerability was extremely critical because it could enable threat actors to perform denial-of-service and remote code execution attacks. TrendLabs Security Intelligence Blog had stated that It was through EHLO strings that a threat actor could exploit CVE-2019-16928 to trigger malicious attacks, such as terminating the Exim process that resulted in denial-of-service via crash. Furthermore, a backdoor command used as an input for EHLO could lead to remote code execution.[5]

X. Preventing/Detecting Buffer Overflows

In addition to built-in protection in programming languages, modern operating systems have runtime protection against buffer overflow vulnerabilities. Data execution prevention, address space randomization (ASLR), and structured exception handler overwrite protection (SEHOP) are three common runtime protections.[8]

• Data execution prevention - certain areas of memory are flagged as either executable or non-executable, stopping the attack from running code in a region that is non-executable.

• Address space randomization - address space locations in data regions are moved around randomly. This makes buffer overflow attacks almost impossible since they need to know the general location in which the executable code is located.

• Structured exception handler overwrite protection - Structured Exception Handling (SEH), a system built-in to manage hardware and software exceptions, is saved from attacks by malicious code. This means the attacker is prevented from using the SEH overwrite exploitation technique.

If using C or C++, it is best to avoid standard library functions such as scanf, gets, and strcpy because they are not bounds-checked. Additionally, it is recommended that systems be checked regularly to detect and fix buffer overflows. Enforcing bounds-checking at run-time is another solution, which automatically checks that data is written to a buffer that is in accepted boundaries.[8]

Even with all these prevention measures, buffer overflows can still occur and put a user's system at risk of attack. This is where it is useful to be able to detect a buffer overflow. Many times, random values called canaries are created by compilers running a program and placed on the stack following each buffer. These canaries' function is to flag danger, so the user can compare the canary's current value against its original value to determine if a buffer overflow has occurred. If the value has been altered, the user can terminate the program or allow it to go into an error state instead of continuing to the potentially altered return address.[6]

References

[1] "A buffer overflow in a "Type safe" Language? - SANS Internet Storm Center", *SANS Internet Storm Center*, 2022. [Online]. Available: https://isc.sans.edu/forums/diary/A+buffer+overflow+in+a+Type+safe+Language/17749/. [Accessed: 19- May- 2022].

[2] "Buffer overflow vulnerabilities, exploits & attacks," *Veracode*. [Online]. Available: https://www.veracode.com/security/buffer-overflow. [Accessed: 18-May-2022].

[3] "Identify, mitigate & prevent buffer overflow attacks on your systems - TuxCare", *TuxCare*, 2022. [Online]. Available: https://tuxcare.com/identify-mitigate-prevent-buffer-overflow-attacks-on-your-systems/#What-are-the-risks-of-a-buffer-overflow-attack. [Accessed: 19-May- 2022].

[4] R. Chandel, W. K. says: and A. S. says: "A beginner's Guide to Buffer Overflow," *Hacking Articles*, 05-May-2021. [Online]. Available: https://www.hackingarticles.in/a-beginners-guide-to-buffer-overflow/. [Accessed: 18-May-2022].

[5] Rofman, O., 2022. *5 Buffer Overflow Vulnerabilities in Popular Apps*. [online] Security Boulevard. Available: https://securityboulevard.com/2019/11/5-buffer-overflow-vulnerabilities-in-popular-apps/ [Accessed 18 May 2022].

[6] S. Team, J. Rabon, C. Purandare, S. Team and C. Freeman, "How to detect, prevent, and

mitigate buffer overflow attacks | Synopsys", *Application Security Blog*, 2022. [Online]. Available: https://www.synopsys.com/blogs/software-security/detect-prevent-and-mitigate-buffer-overflow-attacks/#:~:text=Checking%20the%20value%20of%20the,the%20potentially%20modified%20return%20address. [Accessed: 19- May- 2022].

[7] Wallarm, "What is a buffer overflow attack ❓ types, how hackers use it: Wallarm," *RSS*. [Online]. Available: https://www.wallarm.com/what/buffer-overflow-attack-definition-types-use-by-hackers-part-1. [Accessed: 18-May-2022].

[8] "What is a buffer overflow: Attack types and prevention methods: Imperva," *Learning Center*, 29-Dec-2019. [Online]. Available: https://www.imperva.com/learn/application-security/buffer-overflow/. [Accessed: 18-May-2022].

[9] "What is buffer overflow? attacks, types & vulnerabilities," *Fortinet*. [Online]. Available: https://www.fortinet.com/resources/cyberglossary/buffer-overflow#:~:text=A%20buffer%20overflow%20attack%20typically,composition%20or%20size%20of%20data. [Accessed: 18-May-2022].

[10] "What is cybersecurity?," IBM. [Online]. Available: https://www.ibm.com/topics/cybersecurity. [Accessed: 19-May-2022].