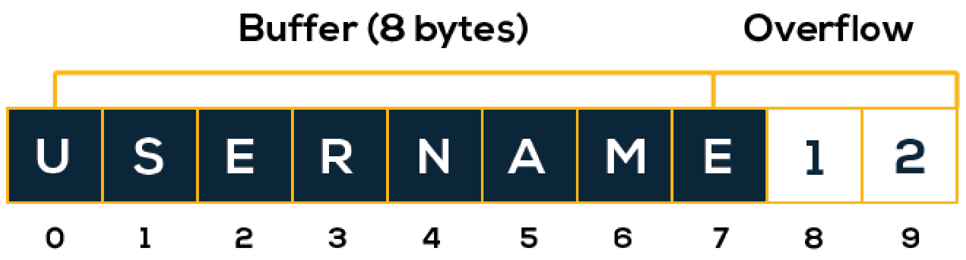
Buffer Overflow Attack

A **buffer** is a temporary area for data storage. When more data (than was originally allocated to be stored) gets placed by a program or system process, the extra data overflows. It causes some of that data to leak out into other buffers, which can corrupt or overwrite whatever data they were holding.

In a **buffer-overflow attack,** the extra data sometimes holds specific instructions for actions intended by a hacker or malicious user; for example, the data could trigger a response that damages files, changes data, or unveils private information.



For example, a buffer for log-in credentials may be designed to expect username and password inputs of 8 bytes, so if a transaction involves an input of 10 bytes (that is, 2 bytes more than expected), the program may write the excess data past the buffer boundary.

An attacker would use a buffer-overflow exploit to take advantage of a program that is waiting on a user's input. There are two types of buffer overflows: stack-based and heap-based. Heap-based, which are difficult to execute and the least common of the two, attack an application by flooding the memory space reserved for a program. Stack-based buffer overflows, which are more common among attackers, exploit applications and programs by using what is known as a stack memory space used to store user input.

How Does Buffer Overflow Attack Work?

A buffer overflow attack occurs when a program tries to fill a memory section with more data than the buffer capacity. Attackers can force the application to run arbitrary code by sending perfectly crafted user input to a vulnerable application. This arbitrary code execution can crash the system or take control of the machine. The attacker knowingly overwrites memory sections identified to hold executable code, modifies it, and significantly changes the way a program works.

Buffer overflow attacks can take place in web application servers providing static and dynamic web structures. Users of these products or services are considered to be at high risk because of extensive knowledge of buffer overflows. Moreover, the use of archives in web applications, such as graphics, for generating images can potentially increase the risks of buffers overflowing.

Types of buffer overflow attacks

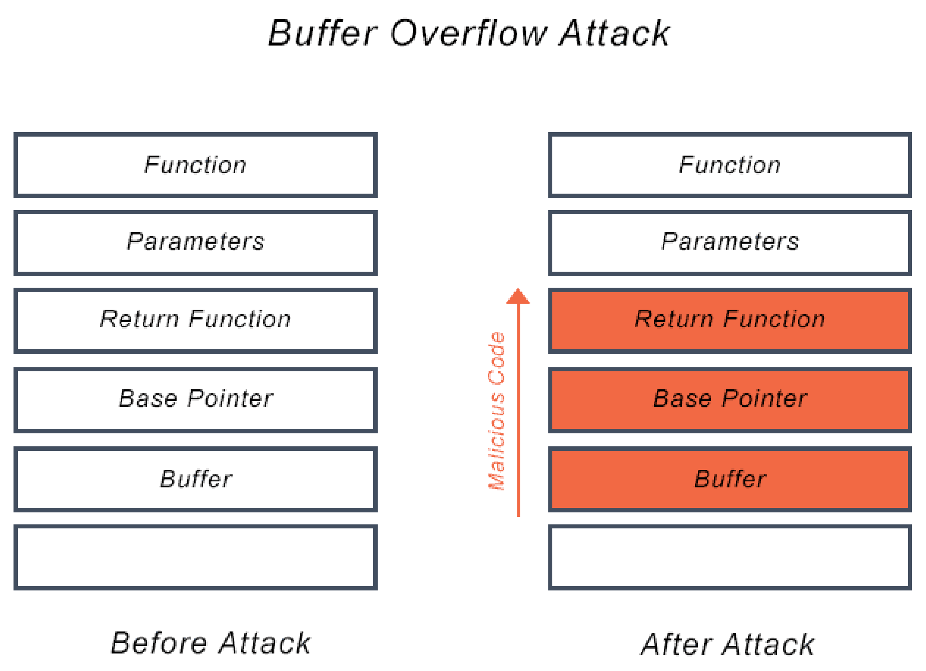
Techniques to exploit buffer overflow vulnerabilities vary based on the operating system (OS) and programming language.

Buffer overflows are categorized according to the location of the buffer in the process memory. They are mostly stack-based overflows or heap-based overflows. Both reside in a device's random-access memory.

Some types of buffer overflow attacks include the following.

1. Stack-based buffer overflow attack

The stack holds data in a last-in, first-out structure. It is a continuous space in memory used to organize data associated with function calls, including function parameters, function local variables and management information, such as frame and instruction pointers. Normally, the stack is empty until the targeted program requires user input, like a username or password. At that point, the program writes a return memory address to the stack, and then the user's input is placed on top of it. When the stack is processed, the user's input gets sent to the return address specified by the program.



However, a stack has a finite size. The programmer who develops the code must reserve a specific amount of space for the stack. If the user's input is longer than the amount of space reserved for it within the stack and the program does not verify that the input will fit, then the stack will overflow. This isn’t a huge problem, but it becomes a huge security hole when it is combined with malicious input.

2. Heap-based buffer overflow attack

The heap is a memory structure used to manage dynamic memory. Programmers often use the heap to allocate memory whose size is not known at compile time, where the amount of memory required is too large to fit on the stack or the memory is intended to be used across function calls. Heap-based attacks flood the memory space reserved for a program or process. Heap-based vulnerabilities, like the zero-day bug discovered in Google Chrome earlier this year, are difficult to exploit, so they are rarer than stack attacks.

Impact Of Buffer Overflow Attack

After a buffer overflow exploitation, your application can become unstable or return with error information. In some cases, data outside the buffer section may contain malicious commands bringing a multitude of problems to the application's security. Let's see what happens when a buffer overflow occurs.

- An attacker takes control of the application by executing commands, known as arbitrary code execution, and happens when code executed in the buffer is executed.
- When the application is executing on an operating system with system privileges, the arbitrary code execution vulnerability could be exploited by a hacker to perform an action called elevation of privileges.
- The hackers can leverage a Denial of Service attack through Buffer Overflow. It can be an incredible source of equipment to execute other attacks, where millions of vulnerable tools are used to perform DDoS attacks.

How to mitigate Buffer Overflow Attack?

- DDoS Protection—maintain uptime in all situations. Prevent any type of DDoS attack, of any size, from preventing access to your website and network infrastructure.
- Web Application Firewall—permit legitimate traffic and prevent bad traffic. Safeguard your applications on-premises and at the edge with an enterprise-class cloud WAF.
- Bot Management– get full visibility and control over human, good bot, and bad bot traffic to your website and API.
- Account Takeover Protection—uses an intent-based detection process to identify and defend against attempts to take over users' accounts for malicious purposes.
- API Security—protects APIs by ensuring only desired traffic can access your API endpoint, as well as detecting and blocking exploits of vulnerabilities.

- RASP—keep your applications safe from within against known and zero-day attacks. Fast and accurate protection with no signature or learning mode.
- Attack Analytics—mitigate and respond to real security threats efficiently and accurately with actionable intelligence across all your layers of defense.

References

https://owasp.org/www-community/vulnerabilities/Buffer_Overflow

https://www.cloudflare.com/learning/security/threats/buffer-overflow/

https://www.veracode.com/security/buffer-overflow