START

Buffer Overflow Attack

FEBRUARY 26, 2024

Press Release: Future Software Should Be Memory Safe

Leaders in Industry Support White House Call to Address Root Cause of Many of the Worst Cyber Attacks

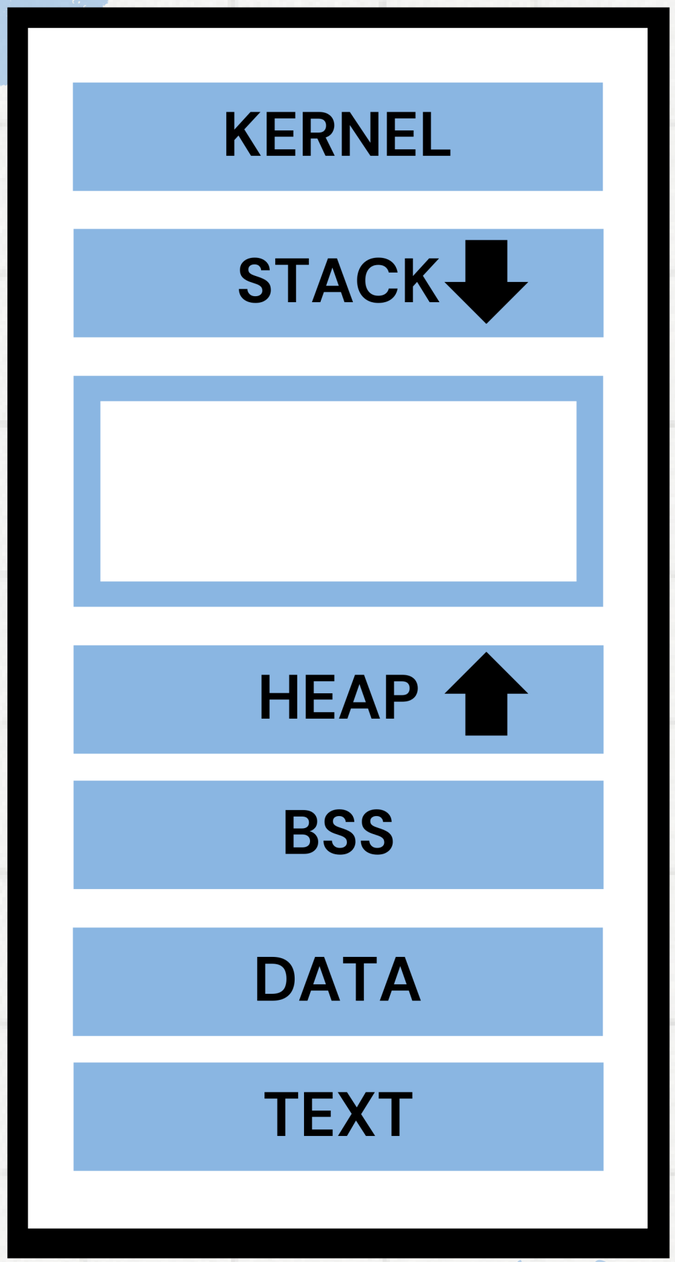
Read the full report here

WASHINGTON – Today, the White House Office of the National Cyber Director (ONCD) released a report calling on the technical community to proactively reduce the attack surface in cyberspace. ONCD makes the case that technology manufacturers can prevent entire classes of vulnerabilities from entering the digital ecosystem by adopting memory safe programming languages. ONCD is also encouraging the research community to address the problem of software measurability to enable the development of better diagnostics that measure cybersecurity quality.

What is a Buffer?

A buffer is a region of computer memory used to temporarily store data as it moves between two different locations or processes.

0xFFFFFFFF



0x00000000

MEMORY LAYOUT

What is a Buffer Overflow?

A buffer overflow, also known as a buffer overrun, is a memory corruption vulnerability that occurs when a program attempts to write data to a **fixed-length buffer** beyond its allocated memory boundaries.

Consequences of Buffer Overflow

- **Data corruption:** Overwritten data in adjacent memory locations can lead to unexpected program behavior, incorrect results, or crashes.

- **Control flow hijacking:** In a severe scenario, buffer overflows can be exploited by attackers. By carefully crafting an overflow payload containing malicious code, attackers can potentially overwrite program instructions and hijack the program's control flow. This can lead to unauthorized code execution, system compromise, or data theft.

Causes of Buffer Overflow

- **1. Low-level programming languages**

Languages like C and C++ provide great control but don't have built-in memory safety mechanisms. They require programmers to manually manage memory allocation and track buffer sizes.

- **2. Unsafe libraries**

Some older standard libraries contain functions (like strcpy, gets in C) that don't perform any bounds checking when copying or manipulating strings.

Causes of Buffer Overflow

- **3. Poor coding practices**

Programmers might neglect input validation, make assumptions about input lengths, or use unsafe functions without proper precautions.

- **4. Lack of secure code review and analysis**

Vulnerabilities like buffer overflows may not be caught during development if there aren't rigorous code reviews or if automated analysis tools aren't used.

Causes of Buffer Overflow

- **5. Lack of penetration testing**

Even well-written code might have exploitable flaws. Without penetration testing (where ethical hackers try to break into the system), latent buffer overflows might remain undetected.

- **6. No specific buffer overflow protection in place**

Without utilizing compiler-level safeguards (e.g., stack canaries) or system-level protections (e.g., ASLR - Address Space Layout Randomization), it's easier for attackers to exploit successful overflows.

Specific protection mechanisms

1. OS Mechanisms

1.1 Address Space Layout Randomization (ASLR)

Randomizes the positions of key memory structures like the stack, heap, and program libraries.

1.2 Data Execution Prevention (DEP)

Marks certain memory regions as non-executable (e.g., the stack).

Specific protection mechanisms

2. Compiler Mechanisms

2.1 StackShield

Inserts a "canary" value (a random value) on the stack adjacent to the return address. Before a function returns, the canary is checked. If it's been modified, the program terminates, indicating a likely overflow attempt.

2.2 StackGuard

A refinement of the canary concept. Also places canaries, but protects more function pointers on the stack, not just the return address.

Specific protection mechanisms

2.3 MemGuard

Maintains separate shadow memory storing the valid bounds of buffers allocated on the heap. Runtime checks against this shadow memory prevent out-of-bounds heap accesses.

2.4 Other GCC patches and compiler options

-fstack-protector: Enables basic stack protection mechanisms like canaries.
-fstack-protector-all: More comprehensive stack protection for all functions.
-D_FORTIFY_SOURCE=2: Activates enhanced bounds checking on common string and memory functions, providing runtime detection for some overflow attempts.

Types of buffer overflow

1. Based on Area of Memory

- **1.1 Stack-based buffer overflow**
  - **Target:** The program's call stack, a region used to store function parameters, local variables, and the return address (where to continue execution after a function ends).
  - **Mechanism:** Overflowing a buffer on the stack can overwrite the return address, potentially allowing the attacker to redirect program execution to malicious code.
  - **Prevalence:** Most common type of buffer overflow due to the stack's predictable layout.emory management mechanisms.

Types of buffer overflow

- **1.2 Heap-based buffer overflow**
    - **Target:** The heap, a region of dynamic memory allocation where programs request space during runtime.
    - **Mechanism:** Overflows on the heap corrupt internal data structures used for memory management. Sophisticated attacks manipulate these structures to gain code execution.
    - **Exploitation:** More complex than stack-based overflows, often requiring deep knowledge of memory management mechanisms.

2. Based on Exploit Technique

- **2.1 Off-by-one overflow**
  - **Nature:** A very specific type of overflow where data is written one byte beyond the buffer's boundary.
  - **Exploitation:** Can sometimes be enough to overwrite a neighboring variable that holds control data or security checks, subtly altering program behavior.
- **2.2 Format string overflow**
  - **Vulnerability:** Occurs due to incorrect use of format string functions (like printf) in C. These functions use format specifiers (%s, %d, etc.) to interpret arguments.
  - **Exploitation:** Attackers inject malicious format specifiers that cause the program to read or write to arbitrary memory locations, potentially leaking sensitive data or executing attacker-controlled code.

- **2.3 Integer overflow**
  - **Concept:** When an arithmetic operation attempts to create a numeric value too large or too small to be represented within the available data type. This can lead to a wrap-around effect, giving an unexpected result.
  - **Relation to buffer overflows:** Integer overflows can sometimes be manipulated to calculate an incorrect buffer size or memory offset, leading indirectly to a buffer overflow condition.