CS 261: 2/3 scribe notes
Rachel Freedman

Memory Safety (spatial)

Spatial memory vulnerabilities are themselves not the most common type of vulnerability, but browser or file exploits are often memory-related. The prototypical example of a spatial memory exploit is when a program accesses memory and intentionally writes to an out-of-bounds pointer, which allows it to inject malicious code.

Common Defenses

- Address Space Layout Randomization (**ASLR**)
    - Allocates objects to random locations in the heap
    - Prevents attacker from predicting where (for example) return addresses that they can overwrite are stored
- Data Execution Prevention (**DEP**) or No-Execute (**NX**)
    - Requires that regions of memory that store data are marked non-executable
    - Prevents attacker from executing malicious code somehow inserted into the data segment of memory
- Stack **Canaries**
    - Inserts random values directly before return addresses
    - When the program makes a function call, it first checks that these canary values haven't changed
    - Prevents attacker from replacing the return address by overwriting this whole section of memory (including the canary)

Return-Oriented Programming

- Return-Oriented Programming (**ROP**) attack
    - Overwrites program return address with an address pointing to another piece of code in the same program (so validly executable)
    - **Return-into-libc attack** (baby ROP): find somewhere in code that does something useful for the attacker, overwrite return address to jump there
    - **ROP**: Piece tiny snippets of the original code (called **gadgets**) together to do something malicious (ex. run a shell)
    - NX alone doesn't defeat ROP attacks because the gadgets come from parts of the program (which are marked executable in memory)
    - However, NX and ASLR in combination can defeat ROP attacks because the attacker can't reliably find return addresses to overwrite and gadgets to combine
- Blind ROP (**B-ROP**) attack
    - Enables ROP attack even when attacker doesn't have access to the codebase
    - Attacker first finds gadgets that stop and crash the program, then bootstraps up to more useful gadgets, eventually combining enough to create a shell command

    - Because the program crashes when canaries are incorrect, the attacker can brute-force-guess canaries bit-by-bit

Other Schemes for bounds checking

- **PAriCheck**
    - Include a table that stores a unique ID for every object
    - Check whether first and last pointer corresponds to same ID
- **Fat pointers**
    - Pointer includes 3 pieces of information: address, “base” (start of object) and object size