

CSci 4271W
Development of Secure Software Systems
Day 6: Memory safety attacks 2

Stephen McCamant
University of Minnesota, Computer Science & Engineering

Outline

Shellcode techniques

Examples in GDB

Exploiting other vulnerabilities

W⊕X (DEP)

Basic definition

- Shellcode: attacker supplied instructions implementing malicious functionality
- Name comes from example of starting a shell
- Often requires attention to machine-language encoding

Classic `execve /bin/sh`

- `execve(fname, argv, envp)` system call
- Specialized syscall calling conventions
- Omit unneeded arguments
- Doable in under 25 bytes for Linux/x86

Avoiding zero bytes

- Common requirement for shellcode in C string
- Analogy: broken 0 key on keyboard
- May occur in other parts of encoding as well

Outline

Shellcode techniques

Examples in GDB

Exploiting other vulnerabilities

W⊕X (DEP)

Demo

- Overwriting the return address
- Jumping to shellcode

Outline

Shellcode techniques

Examples in GDB

Exploiting other vulnerabilities

W⊕X (DEP)

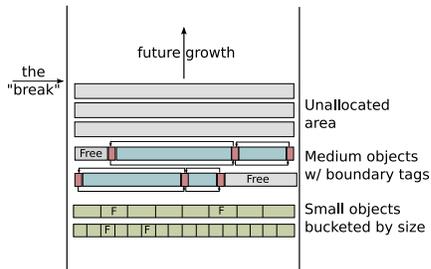
Non-control data overwrite

- Overwrite other security-sensitive data
- No change to program control flow
- Set user ID to 0, set permissions to all, etc.

Heap meta-data

- Boundary tags similar to doubly-linked list
- Overwritten on heap overflow
- Arbitrary write triggered on `free`
- Simple version stopped by sanity checks

Heap meta-data



Use after free

- Write to new object overwrites old, or vice-versa
- Key issue is what heap object is reused for
- Influence by controlling other heap operations

Integer overflows

- Easiest to use: overflow in small (8-, 16-bit) value, or only overflowed value used
- 2GB write in 100 byte buffer
 - Find some other way to make it stop
- Arbitrary single overwrite
 - Use math to figure out overflowing value

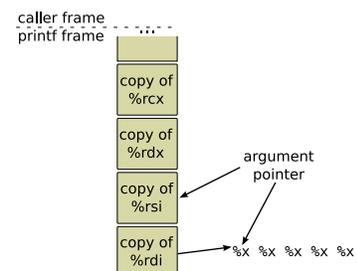
Null pointer dereference

- Add offset to make a predictable pointer
 - On Windows, interesting address start low
- Allocate data on the zero page
 - Most common in user-space to kernel attacks
 - Read more dangerous than a write

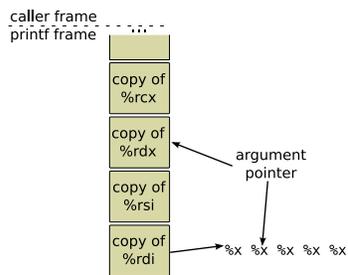
Format string attack

- Attacker-controlled format: little interpreter
- Step one: add extra integer specifiers, dump stack
 - Already useful for information disclosure

Format string attack layout



Format string attack layout



Format string attack: overwrite

- `%n` specifier: store number of chars written so far to pointer arg
- Advance format arg pointer to other attacker-controlled data
- Control number of chars written with padding
- On x86, can use unaligned stores to create pointer

Outline

Shellcode techniques

Examples in GDB

Exploiting other vulnerabilities

$W \oplus X$ (DEP)

Basic idea

- Traditional shellcode must go in a memory area that is
 - writable, so the shellcode can be inserted
 - executable, so the shellcode can be executed
- But benign code usually does not need this combination
- W xor X , really $\neg(W \wedge X)$

Non-writable code, $X \rightarrow \neg W$

- E.g., read-only `.text` section
- Has been standard for a while, especially on Unix
- Lets OS efficiently share code with multiple program instances

Non-executable data, $W \rightarrow \neg X$

- Prohibit execution of static data, stack, heap
- Not a problem for most programs
 - Incompatible with some GCC features no one uses
 - Non-executable stack opt-in on Linux, but now near-universal

Implementing $W \oplus X$

- Page protection implemented by CPU
 - Some architectures (e.g. SPARC) long supported $W \oplus X$
- x86 historically did not
 - One bit controls both read and execute
 - Partial stop-gap "code segment limit"
- Eventual obvious solution: add new bit
 - NX (AMD), XD (Intel), XN (ARM)

One important exception

- Remaining important use of self-modifying code: just-in-time (JIT) compilers
 - E.g., all modern JavaScript engines
- Allow code to re-enable execution per-block
 - `mprotect`, `VirtualProtect`
 - Now a favorite target of attackers

Counterattack: code reuse

- Attacker can't execute new code
- So, take advantage of instructions already in binary
- There are usually a lot of them
- And no need to obey original structure

Classic return-to-libc (1997)

- Overwrite stack with copies of:
 - Pointer to libc's `system` function
 - Pointer to `"/bin/sh"` string (also in libc)
- The `system` function is especially convenient
- Distinctive feature: return to entry point

Chained return-to-libc

- Shellcode often wants a sequence of actions, e.g.
 - Restore privileges
 - Allow execution of memory area
 - Overwrite system file, etc.
- Can put multiple fake frames on the stack
 - Basic idea present in 1997, further refinements

Pop culture analogy: ransom note trope

