CSci 4271W
Development of Secure Software Systems
Day 6: Memory safety defenses and counter-attacks
Stephen McCamant
University of Minnesota, Computer Science & Engineering

Outline
- W ⊕ X (DEP)
- Return-oriented programming (ROP)
- Announcements break
- ROP shellcoding exercise

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: (W ^ X)

Non-writable code, X → ¬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 → ¬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 ⊕ X
- Page protection implemented by CPU
  - Some architectures (e.g. SPARC) long supported W ⊕ 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

Outline

- W X (DEP)
- Return-oriented programming (ROP)
- Announcements break
- ROP shellcoding exercise

Pop culture analogy: ransom note trope

Gadgets

- Basic code unit in ROP
- Any existing instruction sequence that ends in a return
- Found by (possibly automated) search

Basic new idea

- Treat the stack like a new instruction set
- "Opcodes" are pointers to existing code
- Generalizes return-to-libc with more programmability
- Academic introduction and source of name: Hovav Shacham, ACM CCS 2007

ret2pop (Nergal, Müller)

- Take advantage of shellcode pointer already present on stack
- Rewrite intervening stack to treat the shellcode pointer like a return address
  - A long sequence of chained returns, one pop
Another partial example

Overlapping x86 instructions

Where gadgets come from

Possibilities:
- Entirely intended instructions
- Entirely unaligned bytes
- Fall through from unaligned to intended
- Standard x86 return is only one byte, 0xc3

Building instructions

String together gadgets into manageable units of functionality

Examples:
- Loads and stores
- Arithmetic
- Unconditional jumps
- Must work around limitations of available gadgets

Hardest case: conditional branch

Existing jCC instructions not useful
But carry flag CF is

Three steps:
1. Do operation that sets CF
2. Transfer CF to general-purpose register
3. Add variable amount to %esp

Further advances in ROP

Can also use other indirect jumps, overlapping not required
Automation in gadget finding and compilers
In practice: minimal ROP code to allow transfer to other shellcode

Outline

Office hours

Me: Mondays 1-2pm, TBA, or email for appointment
Saugata: Mondays 4-5pm, Thursdays 10-11am
Zoom links coming soon to Canvas page
Project 1 status

- Badly Coded developers implementing last few features
- Expect code release over weekend, full instructions by next lecture
- Initial due date (attacks and first report) will be 10/9

Important of attacks and shellcoding

- Constructing attacks will be important for the project
- Keeping looking at yesterday’s lab if you didn’t finish, we’ll come back to this next week

Outline

- W X (DEP)
- Return-oriented programming (ROP)
- Announcements break
- ROP shellcoding exercise

Setup

- Key motivation for ROP is to disable W X
- Can be done with a single syscall, similar to execve shellcode
- Your exercise for today: put together such shellcode from a limited gadget set
- Puzzle/planning aspect: order to avoid overwriting