Outline

Return-oriented programming (ROP), cont’d
ROP shellcoding exercise
More perspectives on threat modeling
Threat modeling: printer manager

Pop culture analogy: ransom note trope

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

Gadgets

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

Another partial example

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

Overlap x86 instructions

- Variable length instructions can start at any byte
- Usually only one intended stream
### 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
- Return-oriented programming (ROP), cont'd
- ROP shellcoding exercise
- More perspectives on threat modeling
- Threat modeling: printer manager

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

### Outline
- Return-oriented programming (ROP), cont'd
- ROP shellcoding exercise
- More perspectives on threat modeling
- Threat modeling: printer manager

### Software-oriented modeling
- This is what we've concentrated on until now
- And it will still be the biggest focus
- Think about attacks based on where they show up in the software
- Benefit: easy to connect to software-level mitigations and fixes

### Asset-oriented modeling
- Think about threats based on what assets are targeted / must be protected
- Useful from two perspectives:
  - Predict attacker behavior based on goals
  - Prioritize defense based on potential losses
- Can put other modeling in context, but doesn't directly give you threats
Kinds of assets

Three overlapping categories:
- Things attackers want for themselves
- Things you want to protect
- Stepping stones to the above

Attacker-oriented modeling

Think about threats based on the attacker carrying them out
- Predict attacker behavior based on characteristics
- Prioritize defense based on likelihood of attack
- Limitation: it can be hard to understand attacker motivations and strategies
- Be careful about negative claims

Kinds of attackers (Intel TARA)
- Competitor
- Data miner
- Radical activist
- Cyber vandal
- Sensationalist
- Civil activist
- Terrorist
- Anarchist
- Irrational individual
- Gov’t cyber warrior
- Legal adversary

Kinds of attackers (cont’d)
- Internal spy
- Government spy
- Chief
- Vendor
- Reckless employee
- Information partner
- Disgruntled employee

Outline

Return-oriented programming (ROP), cont’d
ROP shellcoding exercise
More perspectives on threat modeling
Threat modeling: printer manager

Setting: shared lab with printer

Imagine a scenario similar to CSE Labs
- Computer labs used by many people, with administrators
- Target for modeling: software system used to manage printing
  - Similar to real system, but use your imagination for unknown details

Example functionality

- Queue of jobs waiting to print
  - Can cancel own jobs, admins can cancel any
- Automatically converting documents to format needed by printer
- Quota of how much you can print

Things to model

- Draw architecture with data flows and trust boundaries
- List assets and attackers
- What are the threats a system must block?