CSci 4271W
Development of Secure Software Systems
Day 7: ROP and More Threat Modeling
Stephen McCamant
University of Minnesota, Computer Science & Engineering

Outline
Return-oriented programming (ROP)
ROP shellcoding exercise
More perspectives on threat modeling
Threat modeling: printer manager, part 1

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

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

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

Another partial example
Overlapping x86 instructions

- `movl Dex(esi, edx)`
- `inc 0x8(%eax)`
- `or %al, %dh`
- `movzbl 0x1c(%esi), %edx`

Variable length instructions can start at any byte
Usually only one intended stream

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

- Return-oriented programming (ROP)
- ROP shellcoding exercise
- More perspectives on threat modeling
- Threat modeling: printer manager, part 1

Setup

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

Kinds of attackers (Intel TARA)

- Competitor
- Data miner
- Radical activist
- Cyber vandal
- Sensationalist
- Civil activist
- Terrorist
- Anarchist
- Irrational individual
- Gov’t cyber warrior
- Corrupt gov’t official
- Legal adversary

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

Outline

- Return-oriented programming (ROP)
- ROP shellcoding exercise
- More perspectives on threat modeling
- Threat modeling: printer manager, part 1
<table>
<thead>
<tr>
<th>Example functionality</th>
<th>Assets and attackers</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Queue of jobs waiting to print</td>
<td>- What assets is the system protecting?</td>
</tr>
<tr>
<td>- Can cancel own jobs, admins can</td>
<td>- What negative consequences do we want to avoid?</td>
</tr>
<tr>
<td>cancel any</td>
<td></td>
</tr>
<tr>
<td>- Automatically converting documents</td>
<td>- Who are the relevant attackers?</td>
</tr>
<tr>
<td>to format needed by printer</td>
<td>- What goals motivate those attackers?</td>
</tr>
<tr>
<td>- Quota of how much you can print</td>
<td>- Take 5 minutes to brainstorm with your neighbors</td>
</tr>
</tbody>
</table>