

CSci 4271W
Development of Secure Software Systems
Day 7: More Threat Modeling, ROP

Stephen McCamant
University of Minnesota, Computer Science & Engineering

Outline

- More perspectives on threat modeling
- Threat modeling: printer manager
- Return-oriented programming (ROP)
- Attacks and shellcode lab followup

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

Kinds of attackers (cont'd)

- Internal spy
- Government spy
- Thief
- Vendor
- Reckless employee
- Information partner
- Disgruntled employee

Outline

More perspectives on threat modeling

Threat modeling: printer manager

Return-oriented programming (ROP)

Attacks and shellcode lab followup

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?

Outline

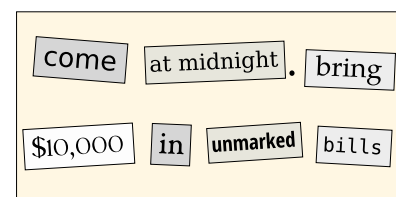
More perspectives on threat modeling

Threat modeling: printer manager

Return-oriented programming (ROP)

Attacks and shellcode lab followup

Pop culture analogy: ransom note trope



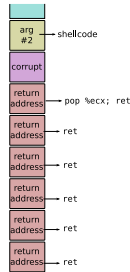
Basic new idea

- 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

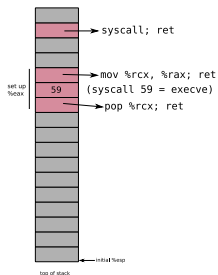
ret2pop (Nergal, Müller)



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

```

push %esi
mov $0x56,%dh|sbb $0xff,%al|inc %eax|or %al,%dh
movzbl 0x1c(%esi),%edx|incl 0x8(%eax) ...
0f b6 56 1c ff 40 08 c6
    
```

- 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

More perspectives on threat modeling

Threat modeling: printer manager

Return-oriented programming (ROP)

Attacks and shellcode lab followup

Reminder: what is shellcode

- Machine code that does the attacker's desired behavior
- Just a few instructions, not a complete program
- Usually represented as sequence of bytes in hex

Reminder: basic attack sequence

- Make the program do an unsafe memory operation
- Use control to manipulate control-flow choice
 - E.g.: return address, function pointer
- Make the target of control be shellcode

Overflow example hands-on

- Steps of overflow-from-file example

Side-effects example

- A second example with a new wrinkle