CSci 4271W
Development of Secure Software Systems
Day 7: Threat Modeling 2

Stephen McCamant
University of Minnesota, Computer Science & Engineering

Outline

- More perspectives on threat modeling
- Announcements intermission
- Threat modeling: printer manager
- Return-oriented programming (ROP)

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
- Corrupt gov't official
- 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

Announcements intermission

Threat modeling: printer manager

Return-oriented programming (ROP)

Project 0.5 now available

- Code auditing and attacking against BCBASIC
- Audit and attacks in groups of up to 3, write reports individually
- More realistic code auditing than you've had to do before
- Due Friday, February 23rd

Outline

More perspectives on threat modeling

Announcements intermission

Threat modeling: printer manager

Return-oriented programming (ROP)

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

Assets and attackers

- What assets is the system protecting?
  - What negative consequences do we want to avoid?
- Who are the relevant attackers?
  - What goals motivate those attackers?
- Take 5 minutes to brainstorm with your neighbors

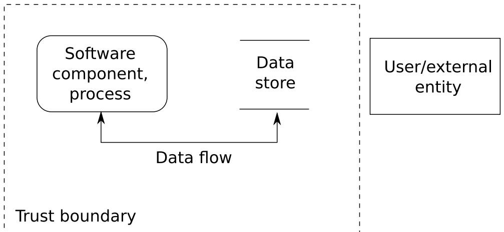
Assets and attackers

- Administrators:
  - Want to let students do printing needed for classes
  - While minimizing spending on paper, toner, and admins responding to problems
- Attackers:
  - Non-students might try to print
  - Students might try to print too much
  - Students might interfere with each other

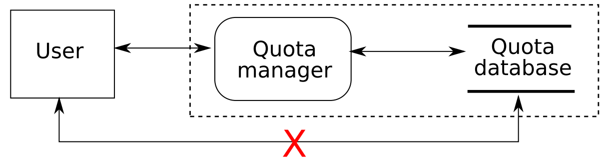
Data flow diagram

- Show structure of users, software/hardware components, data flows, and trust boundaries
- For this exercise, can mix software, OS, and network perspectives
- Include details relevant to security design decisions
- Take 15 minutes to draw with your neighbors

Data flow diagram: key

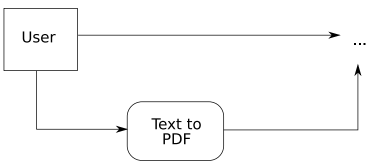


DFD #1: access control



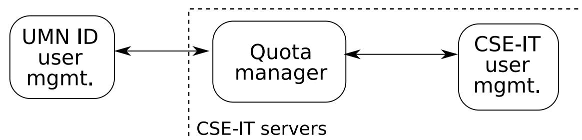
- The absence of data flow will need an implementation

DFD #2: optional processing



- Text-to-PDF can't add much risk here

DFD #3: a trust boundary



- Different risks from where authentication lies

STRIDE threat brainstorming

- Think about possible threats using the STRIDE classification
- Are all six types applicable in this example?
- Take 10 minutes to brainstorm with your neighbors

STRIDE examples

S: make your jobs look like a different student's
T: insert mistakes in another student's homework
R: claim you don't know why your quota is used up
I: read another student's homework
D: break printing before an assignment deadline
E: student performs administrator actions

Outline

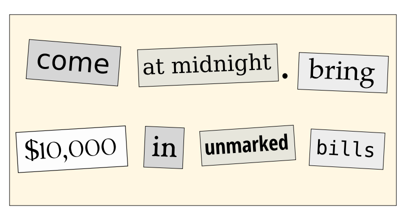
More perspectives on threat modeling

Announcements intermission

Threat modeling: printer manager

Return-oriented programming (ROP)

Pop culture analogy: ransom note trope



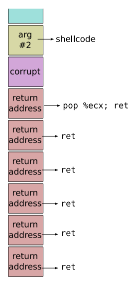
Basic new idea

- 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

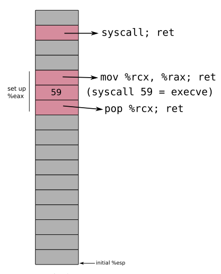
ret2pop (Nergal, Müller)



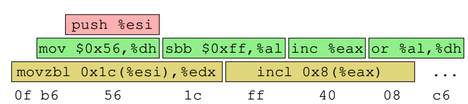
Gadgets

- 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



- 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