CSE 127: Introduction to Security

Threat modeling continued

Nadia Heninger
UCSD

Winter 2021 Lecture 2

Some slides from Kirill Levchenko, Stefan Savage, and Deian Stefan
Continued from last time: Threat modeling
Exercise

How would you steal my email password?
Exercise

How would you steal an election?
Exercise

What security systems do you interact with?
Thinking like a Defender

- Security policy
  - What are we trying to protect?
  - What properties are we trying to enforce?

- Threat model
  - Who are the attackers? Capabilities? Motivation?
  - What kind of attack are we trying to prevent?

- Risk assessment
  - What are the weaknesses of the system?
  - What will successful attacks cost us?
  - How likely?

- Countermeasures
  - Costs vs. benefits?
  - Technical vs. nontechnical?
Security Policies

• What assets are we trying to protect?
  • Password (hashes)
  • Emails
  • Browsing history

• What properties are we trying to enforce?
  • Confidentiality
  • Integrity
  • Availability
  • Privacy
  • Authenticity
Threat Models

• Who are our adversaries?
  • Motives?
  • Capabilities?

• What kinds of attacks do we need to prevent? (Think like the attacker!)

• Limits: What kinds of attacks we should ignore?
Example of Threat Modeling

<table>
<thead>
<tr>
<th>Threat</th>
<th>Solution</th>
<th>Threat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ex-girlfriend/boyfriend breaking into your email account and publicly releasing your correspondence with the My Little Pony fan club</td>
<td>Strong passwords</td>
<td>The Mossad doing Mossad things with your email account</td>
</tr>
<tr>
<td>Organized criminals breaking into your email account and sending spam using your identity</td>
<td>Strong passwords + common sense (don’t click on unsolicited herbal Viagra ads that result in keyloggers and sorrow)</td>
<td>Magical amulets?</td>
</tr>
<tr>
<td>The Mossad doing Mossad things with your email account</td>
<td></td>
<td>Fake your own death, move into a submarine?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>YOU'RE STILL GONNA BE MOSSAD'ED UPON</td>
</tr>
</tbody>
</table>

James Mickens “This World of Ours”
Example of Threat Modeling

Hi John

Someone just used your password to try to sign in to your Google Account john.podesta@gmail.com.

Details:
- Saturday, 19 March, 8:34:30 UTC
- IP Address: 134.249.139.239
- Location: Ukraine

Google stopped this sign-in attempt. You should change your password immediately.

CHANGE PASSWORD

Best,
The Gmail Team
Who is John Podesta?
Assessing Risk

Remember: *Controlled paranoia*

- What would security breaches cost us?
  - Direct costs: Money, property, safety, …
  - Indirect costs: Reputation, future business, well being, …

- How likely are these costs?
  - Probability of attacks?
  - Probability of success?
Countermeasures

- Technical countermeasures

- Nontechnical countermeasures
  Law, policy (government, institutional), procedures, training, auditing, incentives, etc.
How do we protect classified satellites?
Security Costs

• No security mechanism is free
  • Direct costs:
    Design, implementation, enforcement, false positives
  • Indirect costs:
    Lost productivity, added complexity

• Challenge is to rationally weigh costs vs. risk
  • Human psychology makes reasoning about high cost/low probability events hard
Exercise

Should you lock your door?

• Assets?
• Adversaries?
• Risk assessment?
• Countermeasures?
• Costs/benefits?
Should you use automatic software updates?

- Assets?
- Adversaries?
- Risk assessment?
- Countermeasures?
- Costs/benefits?
Exercise

Should we protect the CSE bear?

- Assets?
- Adversaries?
- Risk assessment?
- Countermeasures?
- Costs/benefits?
Secure Design

• Common mistake:
  Convince yourself that the system is secure

• Better approach:
  Identify weaknesses of design, focus on correcting them
  Formally prove that design is secure (soon)

• Secure design is a process
  Must be practiced continuously
  Retrofitting security is super hard
Where to focus defenses

• *Trusted components*
  Parts that must function correctly for the system to be secure.

• *Attack surface*
  Parts of the system exposed to the attacker

Security Principles

• Simplicity, open design, and maintainability
• Privilege separation and least privilege
• Defense-in-depth and diversity
• Complete mediation and fail-safe
Exercise

Preventing cheating on an online exam?
Exercise

Preventing you from stealing my password?
Stack Buffer Overflows
When is a program secure?

- Formal approach: When it does exactly what it should
  - Not more
  - Not less

- But how do we know what it is supposed to do?
When is a program secure?

• Formal approach: When it does exactly what it should
  • Not more
  • Not less

• But how do we know what it is supposed to do?
  • Somebody tells us? (Do we trust them?)
  • We write the code ourselves? (What fraction of the software you use have you written?)
When is a program secure?

- Pragmatic approach: When it doesn’t do bad things
- Often easier to specify a list of “bad” things:
  - Delete or corrupt important files
  - Crash my system
  - Send my password over the internet
  - Send threatening email to the professor
When is a program secure?

What if the program doesn’t do bad things, but could?

Is it secure?
Weird machines

- Complex systems contain unintended functionality
- Attackers can trigger this unintended functionality
  - i.e. they are exploiting vulnerabilities

![Diagram showing expected, valid input leading to normal, intended functionality, and unexpected input leading to unintended functionality.](image-url)
What is a software vulnerability?

- A bug in a program that allows an unprivileged user capabilities that should be denied to them
- There are many types of vulnerabilities
- Today: bugs that violate "control / integrity"
- Why? This lets an attacker run code on your computer!
- Typically these involve violating assumptions of the programming language or its runtime
What is a software vulnerability?

• A bug in a program that allows an unprivileged user capabilities that should be denied to them
What is a software vulnerability?

• A bug in a program that allows an unprivileged user capabilities that should be denied to them

• There are many types of vulnerabilities

• Today: bugs that violate “control flow integrity”
  • Why? This lets an attacker run code on your computer!
What is a software vulnerability?

• A bug in a program that allows an unprivileged user capabilities that should be denied to them

• There are many types of vulnerabilities

• Today: bugs that violate “control flow integrity”
  • Why? This lets an attacker run code on your computer!

• Typically these involve violating *assumptions* of the programming language or its runtime
Exploiting vulnerabilities (the start)

• Dive into low-level details of how exploits work
  • How can a remote attacker get a victim program to execute their code?

• Threat model: Victim code is handling input that comes from across a security boundary
  • What are some examples of this?

• Security policy: Want to protect integrity of execution and confidentiality of data from being compromised by malicious and highly skilled users of our system.