Threat Modeling

Threat Modeling

> Threat modeling is a **process by which potential threats**, such as structural vulnerabilities **can be identified, enumerated, and prioritized** – all from a hypothetical attacker's point of view.
>
> The purpose of threat modeling is to **provide defenders with a systematic analysis of the probable attacker's profile, the most likely attack vectors, and the assets most desired by an attacker**.
>
> Threat modeling answers questions like "Where are the high-value assets?", "Where am I most vulnerable to attack?", "What are the most relevant threats?", and "Is there an attack vector that might go unnoticed?". [^1]

Reasons to Threat Model

Find Security Bugs Early

- Help you find design issues even before you've written a line of code
- Once you've chosen, changes will be expensive

Shostack, A. (2014) Threat Modeling: Designing for Security, Wiley

Understand Your Security Requirements

> - Good threat models can help you ask "Is that really a requirement?"
> - Interplay between requirements, threats, and mitigations
>   - Some threats don't line up with your business requirements, and as such may not be worth addressing
>   - Your requirements may not be complete
>   - Other threats might be too complex or expensive to address

Shostack, A. (2014) Threat Modeling: Designing for Security, Wiley

Engineer and Deliver Better Products

> - Considering your requirements and design early in the process
> - Dramatically lower the odds that you'll be
>   - re-designing,
>   - re-factoring,
>   - or facing a constant stream of security bugs
> - Deliver a better product on a more predictable schedule

Shostack, A. (2014) Threat Modeling: Designing for Security, Wiley

Address Issues Other Techniques Won’t

> - Threat modeling will lead you to categories of issues that other tools won’t find
> - Models of what goes wrong, by abstracting away details, will help you see analogies and similarities to problems that have been discovered in other systems
> - Threat modeling should not focus on issues that your other safety and security engineering is likely to find

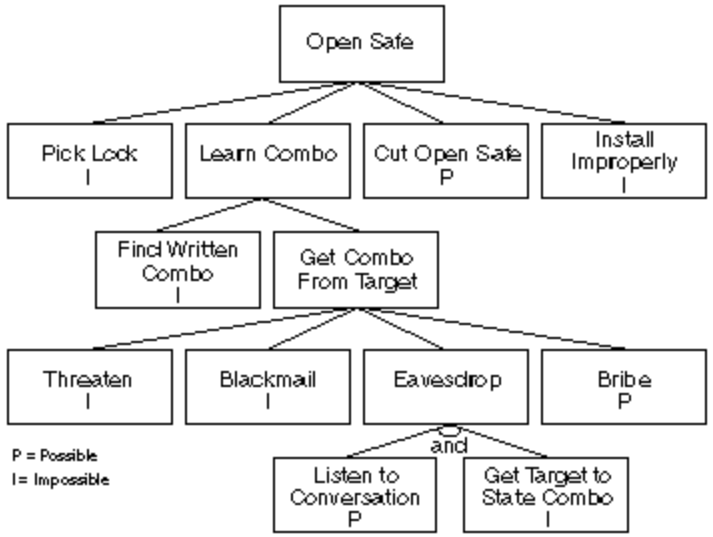
Shostack, A. (2014) Threat Modeling: Designing for Security, Wiley

Attack Trees

> Attack trees provide a formal, methodical way of describing the security of systems, based on varying attacks. Basically, you represent attacks against a system in a tree structure, with the goal as the root node and different ways of achieving that goal as leaf nodes.

Shostack, A. (2014) Threat Modeling: Designing for Security, Wiley

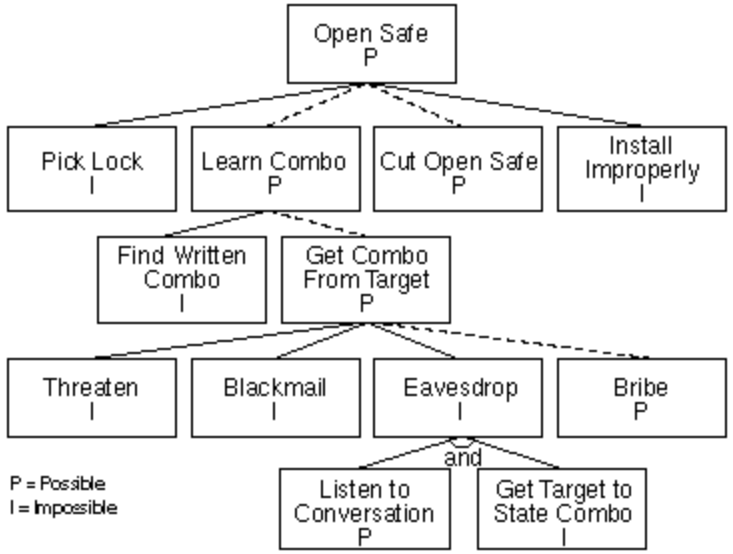
Attack Tree Example: Open Safe



- The goal: Opening the safe
- Each node becomes a subgoal, and children of that node are ways to achieve that subgoal. Parent nodes can be either `OR` or `AND` nodes
- **I** (impossible) or **P** (possible) have been assigned to each leaf node

Shostack, A. (2014) Threat Modeling: Designing for Security, Wiley

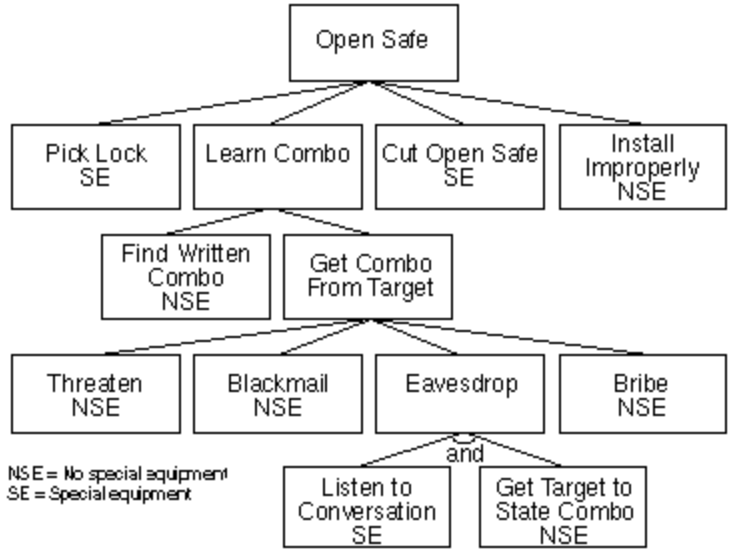
Possible vs. Impossible Attacks



- Dotted lines show possible attacks (=paths only over **P**-nodes)
  - Cutting open the safe
  - Learning the combination by bribing the owner of the safe

Shostack, A. (2014) Threat Modeling: Designing for Security, Wiley

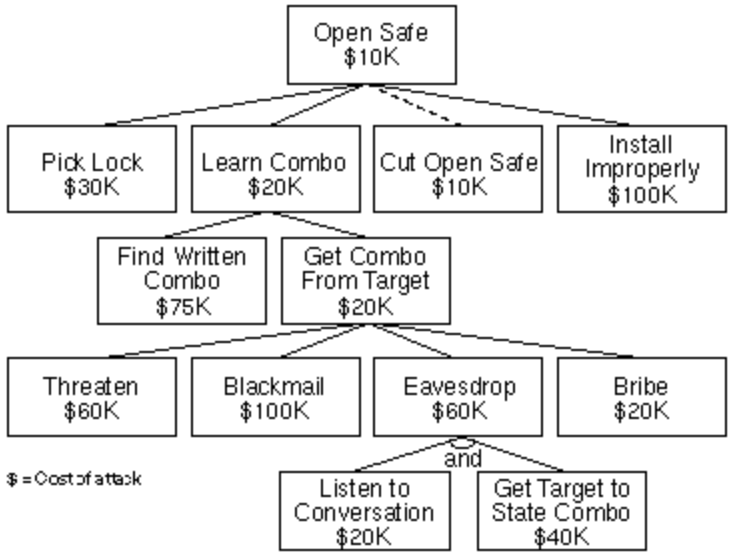
Assigning Boolean values to nodes



- Any Boolean value can be assigned to the leaf nodes and then propagated up the tree structure
- Example: Which attacks require **Special Equipment** (which is probably expensive to retrieve for the attacker)?

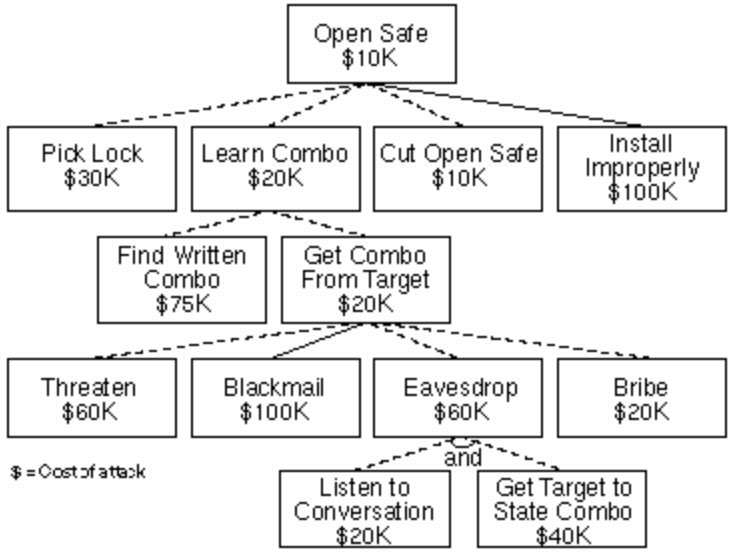
Shostack, A. (2014) Threat Modeling: Designing for Security, Wiley

Assigning continuous values to nodes



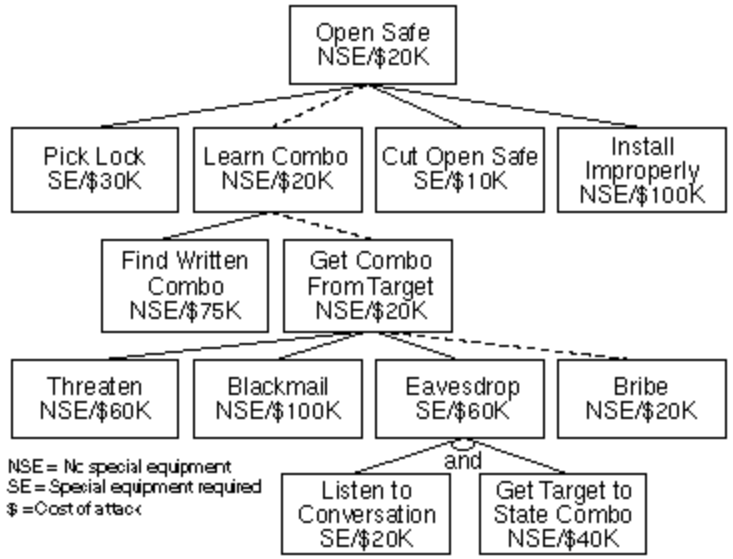
- Costs were assigned to the leaf nodes and propagated up the tree
  - `OR` nodes have the value of their cheapest child
  - `AND` nodes have the value of the sum of their children

Determine vulnerability of the system



- Dotted lines show attacks that cost less than $100,000
- Assumption: The contents of the safe are only worth $100,000
  - You should only concern yourself with attacks cheaper than that!

Assigning different values to nodes



- Dotted line = The cheapest attack requiring no special equipment
- Querying the attack tree about a certain characteristic of attack lets you learn more about the system's security

Exercise 7.1 (📌)

1. Create an attack tree for the goal **"Access Office Building"** (obviously assuming that you are not authorized to do so in the first place)
2. Assign values **H** (human interaction needed) or **N** (no human interaction needed) to each leaf node
3. Assign costs to each leaf node (based on realistic price assumptions)
4. Calculate cost and **H**/**N** values of each node and the goal
5. Make some statements about the building's security based on querying the attack tree in different ways

Four-Step Threat Modeling Framework

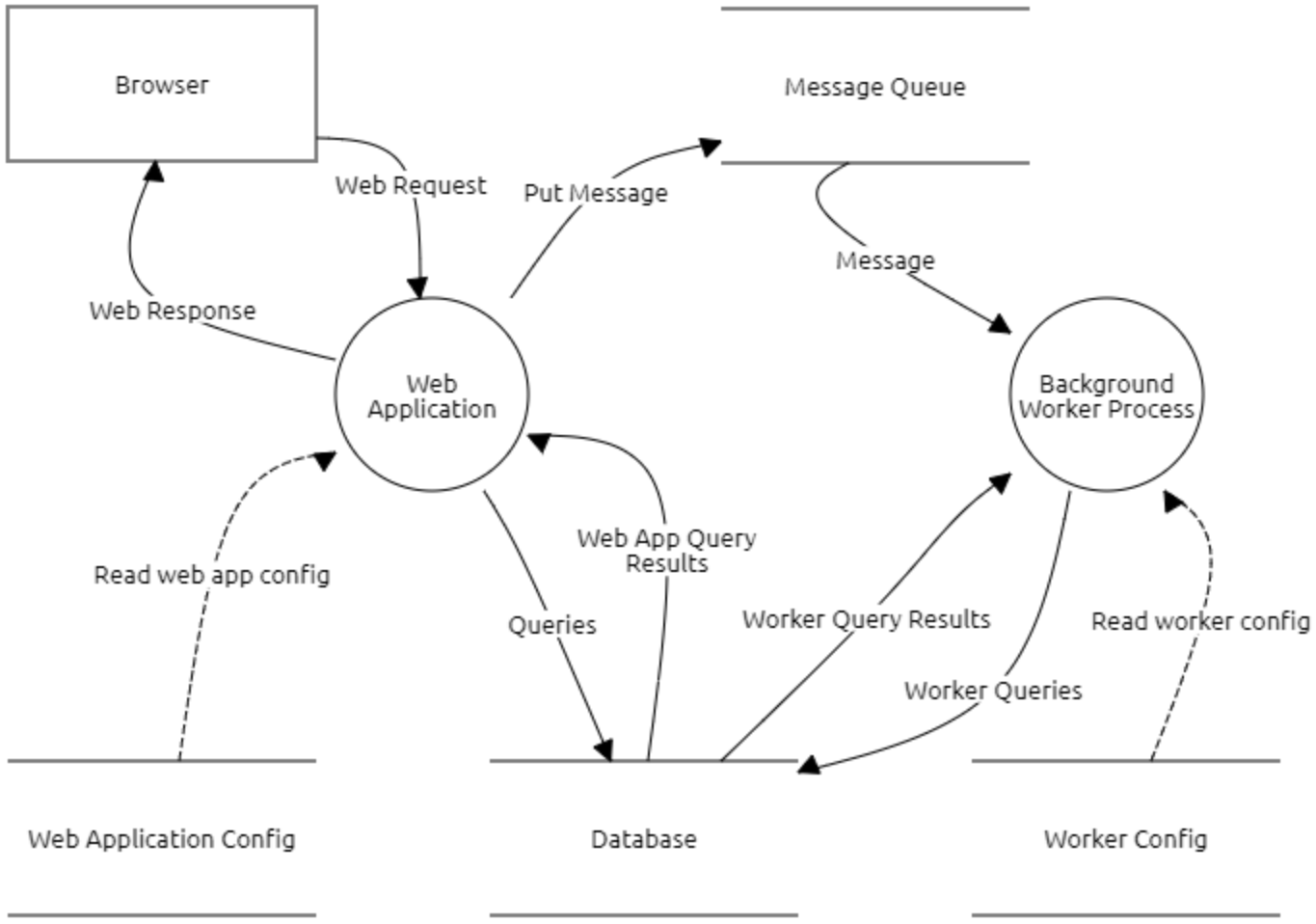
1. What are you building?
2. What can go wrong with it once it's built?
3. What should you do about those things that can go wrong?
4. Did you do a decent job of analysis?

Shostack, A. (2014) Threat Modeling: Designing for Security, Wiley

What are you building?

- Document your system in a practical and easy-to-grasp way, e.g. using a
  - Data Flow Diagram
  - System Context Diagram
  - High Level Architecture Diagram

Example Data Flow



Trust Boundaries

- Boundaries to show who controls what
- Threats that cross those boundaries are likely important ones
- Different people control different things

ℹ️ *Good examples include: Accounts, Network Interfaces, Different physical computers, virtual machines or organizational boundaries.*

Exercise 7.2 (🤝)

1. Mark all trust boundaries in the given Example Data Flow

STRIDE *(What Can Go Wrong?)*

Threat	Description
Spoofing	Pretending to be something or someone you're not
Tampering	Modifying something you're not supposed to modify. It can include packets on the wire (or wireless), bits on disk, or the bits in memory
Repudiation	Claiming you didn't do something (regardless of whether you did or not)
Information Disclosure	Exposing information to people who are not authorized to see it
Denial of Service	Attacks designed to prevent a system from providing service, including by crashing it, making it unusably slow, or filling all its storage
Elevation of Privilege	When a program or user is technically able to do things that they're not supposed to do

Threats vs. Security Goals/Principle

Threat	Security Goal/Principle
Spoofing	Authenticity
Tampering	Integrity
Repudiation	Non-repudiation
Information Disclosure	Confidentiality
Denial of Service	Availability
Elevation of Privilege	Authorization

OWASP Threat Dragon

> OWASP Threat Dragon is a tool used to create threat model diagrams and to record possible threats and decide on their mitigations.
>
> TD is both an online threat modelling web application and a desktop application. It includes system diagramming as well as a rule engine to auto-generate threats and their mitigations. The focus of TD is on great UX, a powerful rule engine and alignment with other development lifecycle tools.

Elevation of Privilege (EoP)

Threat Modeling Card Game

Shostack, A. (2014) Threat Modeling: Designing for Security, Wiley

Elevation of Privilege

1. Deal the deck (Shuffling is optional)
2. The person with the 3 of Tampering leads the first round
3. Play each round (⏭ slide)
4. When all the cards have been played, the game ends and the person with the most points wins
5. If you're threat modeling a system you're building, then you go file any bugs you find

Each round in EoP...

1. Each player plays one card, starting with the person leading the round, and then moving clockwise
2. To play a card, read it aloud, and try to determine if it affects the system you have diagrammed. If you can link it, write it down, and score yourself a point. Play continues clockwise with the next player
3. When each player has played a card, the player who has played the highest card wins the round. That player leads the next round

---

🖨️ EoP cards and accessories for printing (CC BY 3.0 US)

Exercise 7.3 (🤝)

1. Split into groups of 3-6 students
2. If not provided to you, 🖨️ a set of EoP cards and one Score Card
3. ✂️ out the EoP playing cards
4. Play a complete game of EoP for the sample application
5. Keep your *Score Card* in a safe place for Exercise 7.4 (📝)

You can alternatively play EoP remotely in your groups via https://github.com/dehydr8/elevation-of-privilege. Use the Example Data Flow `JSON` model with threat boundaries when setting up the game.

What should you do about those things?

> - **Mitigate**: Doing things to make it harder to take advantage of a threat
> - **Eliminate**: Almost always achieved by eliminating features
> - **Transfer**: Letting someone or something else handle the risk
> - **Accept**: Once you've accepted the risk, you shouldn't worry over it.
>   Sometimes worry is a sign that the risk hasn't been fully accepted, or that the risk acceptance was inappropriate

Shostack, A. (2014) Threat Modeling: Designing for Security, Wiley

Did you do a decent job?

Diagramming

1. Can we tell a story without changing the diagram?
2. Can we tell that story without using words such as “sometimes” or “also”?
3. Can we look at the diagram and see exactly where the software will make a security decision?
4. Does the diagram show all the trust boundaries, such as where different accounts interact? Do you cover all UIDs, all application roles, and all network interfaces?

Shostack, A. (2014) Threat Modeling: Designing for Security, Wiley

5. Does the diagram reflect the current or planned reality of the software?
6. Can we see where all the data goes and who uses it?
7. Do we see the processes that move data from one data store to another?

Threats

1. Have we looked for each of the STRIDE threats?
2. Have we looked at each element of the diagram?
3. Have we looked at each data flow in the diagram?

Validating Threats

1. Have we written down or filed a bug for each threat?
2. Is there a proposed/planned/implemented way to address each threat?
3. Do we have a test case per threat?
4. Has the software passed the test?

Exercise 7.4 (🤝/📝/🏠)

1. Meet with your original EoP-group and have the *Score Card* (or `JSON` model) ready
2. Address each threat identified from Exercise 7.2 and outline your choice of *Mitigate*, *Eliminate*, *Transfer*, or *Accept* with some corresponding measures
3. Go through the *Did you do a decent job?* list to verify your result is sufficiently detailed and up-to-date
4. Scan or take a photo of your Score Card and send it (or your `JSON` model) to bjoern.kimminich@nordakademie.de
5. *(Optional)* Use a PGP encrypted (🔐) and signed (🔏) email in step 4!