Threat Modeling
$ whoami

Jonas Becker
OWASP Frankfurt Co-Lead

- Security Engineer @ Certus Cybersecurity
- Focused on application and cloud security
- Involved in penetration tests, threat modeling, virtual CISO engagements, etc.
- Topics to talk about: Scrubs, Beer & Gin, biking, and all the geeky stuff
Background of Threat Modeling
Basics of Threat Modeling

Penetration Testing
Usually done in the testing & integration phase, often in the maintenance.

Source Code Review
Should be done during implementation.

Threat Modeling
Should be done in the design phase.
Basics of Threat Modeling

Fixing vulnerabilities **costs 100x more** if app is in production\(^{(1)}\)

- **Reduces the attack surface** of the architecture in scope
- **Helps the developers** to think about attack vectors
- Identifies **single point of failures** and bottlenecks

Basics of Threat Modeling

- Light weight process
- Shift security left within the SDLC
- Developers should be included
- The results should feed
  - Risk assessments
  - Source code review
  - Penetration testing
  - ...

- Three different kind of threat models (base, differential, blueprint)

**Base TM**
- Starting from scratch
- Should be done for new or existing solutions

**Differential TM**
- Builds up on base TM
- Done for a new function or design change

**Blueprint TM**
- Done for reoccurring design patterns
- Harder to scope and execute
Framework & Standards

**PASTA**
Process for Attack Simulation and Threat Analysis

**OCTAVE**
Operationally Critical Threat, Asset, and Vulnerability Evaluation

**DREAD**
Damage Potential, Reproducibility, Exploitability, Affected User, Discoverability

**VAST**
Visual, Agile, and Simple Threat Modeling

**STRIDE / LM**
Spoofing, Tampering, Non-Repudiation, Information Disclosure, Denial of Service, Elevation of Privileges, (Lateral Movement)
Threat Modeling Process
Basics of Threat Modeling

**Phase 1**
- Understand the solution
- Research technologies
- Scope the exercise

**Phase 2**
- Identify trust zones
- Depict the architecture
- Document data connections

**Phase 3**
- Spot potential threat objects
- Illustrate possible attack vectors
- Define mitigations
Scoping

✓ Identify your stakeholders
  ○ Business Owner
  ○ Engineers / Architects
  ○ Security Contacts

✓ Understand the solution
  ○ Business context
  ○ Involved components
  ○ Data streams and classification
  ○ Dev/deployment processes

✓ Scope things out
  ○ Identify boundaries / authority limitations
  ○ Specifically outscope such systems out of their control
Architecture Depiction - Interviews

Conduct interviews with relevant stakeholders, covering at least
- Engineers / Architects
- Business Owner
- Security contacts (if available)

During the calls, make sure to
- Ask questions multiple times in a different way
- Ask open ended questions
- Never work with assumptions

Result
- Understanding all the components involved and their roles
- Have a clear picture of the controls in place
Architecture Depiction - Diagram

✓ Create the architecture, which should answer
  ○ Trust Zones
  ○ Components
  ○ Upstream & downstream components / environments
  ○ Any traffic flows
  ○ Users & their devices

✓ System decomposition (https://c4model.com/#CoreDiagrams)

✓ Discuss architecture with the project team
Threat Identification

01 Threat Objects
Identify an adversary’s potential motivations and outline threat objects

02 STRIDE / LM
Walkthrough the diagram (each component and data stream) to identify threats related to:
- Spoofing
- Tampering
- Non-Repudiation
- Information Disclosure
- Denial of Service
- Elevation of Privileges
- Lateral Movement

03 Controls
Review existing controls to highlight any resolved threats as such
Threat Identification - Reporting

Executive Summary
Including a description of the solution, high-level overview of the threats, any assumptions that the threat model was executed with, and the scope.

Threat Model Diagram
Including all details outlined with the team and a description of in-scope and out-of-scope components.

Threat Scenarios
Including a title, detailed description of the impact, attack vectors and any missing controls / mitigation strategies.
Conclusion Thoughts
Limitations & Quality Gates

Limitations
● Timing, scoping and prioritization issues
● The quality of the results are dependent on the design’s quality
● Security engineer may not have enough understanding or wrong level of details
● It does not replace a penetration test or source code review

Quality Gates
● Contains remediated threat scenarios
● Not overloaded with unnecessary details
● Threat scenarios are summarized
● A good threat scenario should at least answer the following questions
  ○ Who could execute the attack?
  ○ How is it executed?
  ○ What are the exploited components?
  ○ Why is it an issue? / What is the impact?