THREAT MODELING

WHAT IS IT

WHY DO YOU NEED IT

HOW TO WRITE IT

HOW TO USE IT

HTTPS://GITHUB.COM/ALEXSCHNEIDER/SAMPLE-THREAT-MODEL

HTTP://IS.GD/EY9SNA

EXAMPLE APPLICATION ("PAY TO POST")

- PII
  - Credit card data
  - Real Name
  - Address
- User Data
  - Login
  - Password
  - Public user info
    - About me
  - Authentication Tokens
- Anonymous and non-anonymous posts

- Users have pages that have public user information
- Users pay to comment on other user's pages
- Users can pay a little more for anonymous comments
- Spam mail is sent to the user's address for additional revenue stream
- Administrative portal to manage user data

WHAT IS A THREAT?

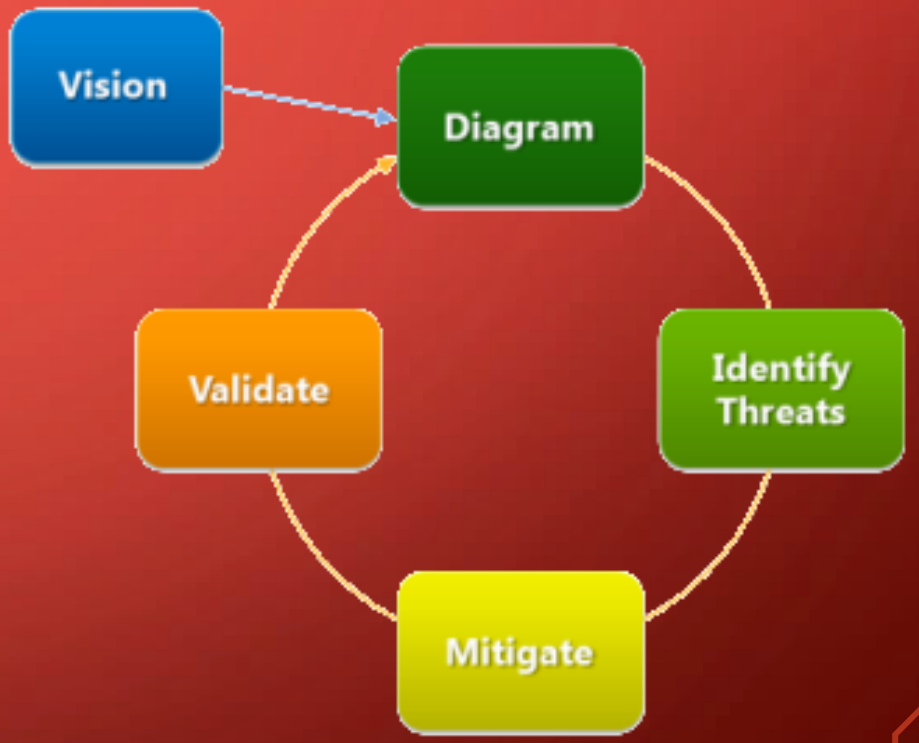
- Potential or actual event that impacts the security or integrity of the system
  - Example: DoS, physical damage, and disclosure, modification, or destruction of data
- Generally categorized in one or more of the following categories (known as STRIDE)
  - **S**poofing
  - **T**ampering
  - Non-**R**epudiation
  - **I**nformation Disclosure
  - **D**enial of Service
  - **E**levation of Privileges

WHAT IS A VULNERABILITY?

- A tangiable issue with an application that an attacker can use to cause harm
  - Can be a design or implementation issue
- Examples:
  - SQL Injection, Cross site scripting, Cross site request forgery
- Vulnerabilities can cause lasting damage to a company's reputation, user base, or infrastructure
- These need to be avoided at all costs!

WHAT IS THREAT MODELING

- From https://www.owasp.org/index.php/Category:Threat_Modeling
- Identify potential vulnerabilities
- Define countermeasures to protect against threats



WHY WRITE A THREAT MODEL?

- Ensure all the code written with a secure mindset
  - Can this code allow an attacker to attack my application? If yes, how do I stop him/her?
- This is a good thing to talk about at the presentations in December – threat modeling and security issues came across during the development of the application

WRITING A THREAT MODEL

- Identify assets and roles
- Identify existing countermeasures
- Identify possible attackers
- Identify threats and possible vulnerabilities
- Identify countermeasures that are needed
- Implement countermeasures

IDENTIFYING ASSETS AND ROLES

- Generally data that must be secret, public with an assurance of validity, or vital to the integrity of the system
- Examples
  - PII – Personally Identifiable Information
  - Authentication tokens
  - User data
- Which user groups should have access to which assets (always, sometimes, never)
- Examples
  - Anonymous user: Sometimes has access to user data (if it's public), never has access to anything else
  - Normal user: Sometimes has access to user data and PII (if it's public or their own), never has access to anything else
  - Admin user: Always has access to everything

IDENTIFY EXISTING COUNTERMEASURES

- Language/Framework
  - Example:
    - JavaScript – Buffer overflows are rare because the language doesn't provide direct memory access
    - Django – CSRF protection bundled into all forms by default
- Implemented in codebase already
  - Taking over a project or doing a threat model after codewriting has started

IDENTIFY POSSIBLE ATTACKERS

- Who will be attacking the application and why? What level of security will the application need and what are the consequences of attacks
- Examples:
  - Online games have hackers who want to gain an unfair advantage – Low
  - E-commerce sites have credit card data which criminals can use or sell – Medium
  - Encrypted messaging services have federal governments who want to read the contents of the messages sent – High

IDENTIFYING THREATS AND POSSIBLE VULNERABILITIES

- Every asset should have one or more of these
- Start out with a statement: An attacker can <x>
  - E.G. An attacker can read PII
- How? Explain what the attacker will do (there may be multiple ways).
  - **Hint**: most of these are going to come from the OWASP Top 10 list: https://www.owasp.org/index.php/Top_10_2013-Top_10 - try to anticipate vulnerabilities from as many of the top 10 as possible
  - Examples:
    - Because data is invalidated improperly on user input, an attacker can inject SQL commands to dump the database (A1-Injection)
    - The application doesn't properly authenticate users, so an attacker can log in as another user and access their data (A2-Broken Authentication and Session Management)
- One thing to keep in mind is how the users are broken up in roles and which assets can be CRUD by which roles (created, read, updated, or deleted)

IDENTIFY COUNTERMEASURES

- Every vulnerability must have a countermeasure – usually implemented in code, but not always (it can be a procedure or an operating system change)
- Some of these can be pretty vague and that's OK (though the more specific it is, the easier down the road)
- Countermeasures for the two attacks we had:
  - Because data is invalidated improperly on user input, an attacker can inject SQL commands to dump the database (A1-Injection)
    - Use an ORM and/or ensure all queries relying on user input are parameterized or prepared statements
  - The application doesn't properly authenticate users, so an attacker can log in as another user and access their data (A2-Broken Authentication and Session Management)
    - Ensure authentication controls exist and restrict access to other user's data

IMPLEMENT COUNTERMEASURES

- Finally, coding!
- Many frameworks or libraries will have optional or mandatory countermeasures – make sure they are used properly
  - However, make sure these frameworks and languages don't introduce additional vulnerabilities
- There may already be solutions available that can be plugged into the application
  - Middleware for a framework to make things more secure, proxies to reject requests hetaeristically, etc

IMPLEMENTING COUNTERMEASURES (CONT.)

- While coding the application
  - Refer back to threat model to make sure assets are protected
  - Update the threat model to reflect new attacks vectors, changes in assets or roles, or updating existing countermeasures you put in

RESOURCES

- OWASP (Open Web Application Security Project)
  - https://www.owasp.org/index.php/Threat_modeling
  - https://www.owasp.org/index.php/Application_Threat_Modeling
- Microsoft SDL (Security Development Lifecycle)
  - http://www.microsoft.com/en-us/sdl/default.aspx
- Security Synergy
  - http://security.howellsonline.ca/threat-model/

QUESTIONS?

- Feel free to contact me on Slack or in the lab with any questions
- Powerpoint will be on slack and I will write up the threat model outlined today and share it with everyone (feel free to copy the layout and structure)