THREAT MODEL

Threat Modeling is the process of analyzing various business and technique requirements of a system, identifying the potential threats, and documenting, How vulnerable this threat makes the system or application.

Threat – Any instance where an unauthorized part access sensitive information, application, or network of an organization

Example

Identifying an encryption algorithm used is to store a user password in your application that is outdated.

Vulnerability – Outdated encryption algorithm MD5

Threat – Decrypting the hashed password using Brute force.

Impact – Trying to sell personal information online

Mitigation – Change an encryption algorithm to something more modern and robust SHA 3

Here are the main objectives of threat modeling:

1. Risk Identification: Threat modeling helps identify potential security risks and vulnerabilities that could be exploited by malicious actors or lead to accidental data breaches.
2. Early Detection of Weaknesses: Threat modeling is typically performed during the early stages of system design and development. By identifying security weaknesses early in the development process, organizations can address them at a lower cost and reduce the chances of deploying a vulnerable system.
3. Prioritization of Security Controls: Not all security risks are equally severe. Threat modeling allows organizations to assess the impact and likelihood of each threat and vulnerability. Based on this assessment, they can prioritize the implementation of security controls and focus on the most critical issues first.
4. Decision Support: Threat modeling provides valuable insights to support decision-making processes related to security investments, risk tolerance, and overall cybersecurity strategy. It enables stakeholders to make informed choices about where to invest resources to enhance security.
5. Continuous Improvement: Threat modeling is an iterative process. As new threats emerge, systems evolve, or new features are added, organizations should revisit and update their threat models to keep security measures current and effective.

The approach is in 3 Different ways

Asset-Centric:- Take stock of various assets and analyze the vulnerabilities of each.

Attacker-Centric:-- Think of all possible attackers, what assets each would want to attack & how.

Software-Centric:- Focus on system design, how the data flows between various layers, and how it is configured.

5 Key steps associated with any Threat Modeling.

1. – Set Objectives – (What do we want to accomplish ?)

Usually, goals are set – CIA

2. – Visualize (What are we building ?)

Clearly documented overview of your entire application

- **Decompose the application**
  Concerned with gaining an understating of the application and how it interacts with external entities.
- How the application is used
- Identifying entry points
- Identifying assets – (the area that the attacker would be interested in – Input fields, Upload, URL Parameters, etc..)
- Identifying trust level
- Data Flow Diagrams (DFDs): DFDs show how data moves between different components and external entities, helping identify potential points of attack and data exposure.
- Control Flow Diagrams: Use control flow diagrams to show how different security controls are implemented and how they interact with each other to mitigate specific threats.
- Swimlane Diagrams: Swimlane diagrams can be used to show the interactions between different entities in the system, such as users, components, and external systems
- Sequence Diagrams: Sequence diagrams can be used to show the sequence of events during an authentication or authorization process. This helps identify potential vulnerabilities in the authentication flow.

3. – Identify Threats and Rank (What can go wrong ?)

Various ways in which your assets can be compromised and who the potential attackers are.

Here are some common methods to identify threats in threat modeling:

- **STRIDE** in threat modeling, you'll focus on six main categories of threats: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege. The STRIDE model provides a systematic and comprehensive approach to identifying and categorizing potential security threats.

Here's how to identify threats using STRIDE:

1. Spoofing:

  - Identify scenarios where an attacker could impersonate a legitimate user or entity.
  - Consider potential weaknesses in authentication and authorization mechanisms that could be exploited for impersonation.
  - Think about scenarios where attackers could forge identities or manipulate data to gain unauthorized access.

Here are some specific threats that fall under the Spoofing category:

1. Username/Password Spoofing: Attackers attempt to impersonate a legitimate user by using stolen or forged credentials (e.g., username and password) to gain unauthorized access to a system or application.
2. IP Address Spoofing: Attackers falsify the source IP address in network packets to make it appear as if the communication is coming from a trusted source, allowing them to bypass access controls and security mechanisms.
3. Email Spoofing: Attackers forge the sender's email address to make an email appear as if it is coming from a trusted source, which can be used for phishing attacks or spreading malware.
4. Website Spoofing: Attackers create fake websites that closely resemble legitimate ones, aiming to trick users into providing sensitive information, such as login credentials or payment details.
5. Service Spoofing: Attackers create malicious services or servers that masquerade as legitimate services, tricking users or other systems into connecting to them and revealing sensitive information.
6. HTTPS/SSL/TLS Certificate Spoofing: Attackers use fake or fraudulently obtained SSL/TLS certificates to make their websites appear secure and trusted, even though they are not, potentially leading to man-in-the-middle attacks.
7. MAC Address Spoofing: Attackers modify the MAC address of their network interface to impersonate another device on the network, leading to unauthorized access or bypassing network restrictions.

2. Tampering:
   - Look for situations where data integrity could be compromised, such as during data transmission or storage.
   - Consider potential vulnerabilities that could allow attackers to modify or manipulate data in transit or at rest.
   - Examine potential weaknesses in data validation and integrity checking mechanisms.

Here are some specific threats that fall under the Tampering category:

1. Data Tampering: Attackers modify or alter data within a system, database, or storage to change information, delete records, or insert malicious content.
2. Code Injection: Attackers inject malicious code (e.g., SQL injection, JavaScript injection) into the system to manipulate its behavior or exploit vulnerabilities.
3. Firmware or Software Modification: Attackers modify the firmware or software running on a device or system to introduce malicious functionality or unauthorized changes.

4. File Tampering: Attackers alter or replace files on a system, such as configuration files or executables, to achieve their objectives.
5. Parameter Tampering: Attackers manipulate parameters in requests or URLs to modify application behavior or bypass security controls.
6. Man-in-the-Middle (MITM) Attack: Attacker’s intercept and alter communication between two parties, potentially modifying data in transit or manipulating commands.
7. Database Tampering: Attackers gain unauthorized access to a database and modify its contents, leading to data corruption or unauthorized access to sensitive information.
8. Session Hijacking: Attackers take control of a user's session and tamper with the data being transmitted during the session.
9. Certificate Manipulation: Attackers tamper with digital certificates to undermine trust and security in secure communications, leading to potential man-in-the-middle attacks.

3. Repudiation:
    - Focus on scenarios where attackers could deny their actions or transactions.
    - Consider situations where there is a lack of proper logging or auditing mechanisms that would allow attackers to cover their tracks.
    - Identify potential vulnerabilities that could enable attackers to modify logs or falsify records.

Here are some specific threats that fall under the Repudiation category:

1. Request/Action Forgery: An attacker forges requests or actions within the system, such as submitting forms, making purchases, or altering data, without leaving any clear evidence of their actions. This could lead to denial of involvement if the actions are discovered later.
2. Tampering with Logs: Attackers modify or delete log entries to hide their activities or to make it appear as if certain events never occurred. By tampering with logs, they can potentially repudiate their actions.
3. Replay Attacks: In a replay attack, an attacker records a sequence of legitimate actions or messages exchanged between components of the system and then later replays them to repeat the actions or create an impact. By doing so, the attacker can deny performing those actions.
4. Non-Repudiable Transactions: In systems that require non-repudiable transactions (e.g., digital signatures, financial transactions), an attacker may find vulnerabilities that allow them to perform actions while avoiding proper verification, leading to repudiation of the transaction.
5. Manipulating Digital Signatures: If a system relies on digital signatures for authentication or verification, an attacker may find ways to manipulate or forge digital signatures, leading to potential repudiation of legitimate actions.
6. Identity Spoofing: By impersonating another user or entity within the system, an attacker can perform actions on behalf of that user, and later claim that they were not responsible for those actions.
7. Data Manipulation without Trace: Attackers may manipulate data, such as changing order details, modifying transaction amounts, or altering documents, without leaving any evidence of the modifications, enabling them to deny involvement.

4. Information Disclosure:
    - Identify points in the system where sensitive information is stored, processed, or transmitted.
    - Consider potential vulnerabilities that could lead to unauthorized access to sensitive data.
    - Look for weaknesses in access controls, encryption, and data handling processes.

Here are some specific threats that fall under the Information Disclosure category:

1. Eavesdropping: Attacker's intercept and monitor communication channels, such as network traffic or unencrypted transmissions, to capture sensitive information.
2. Unsecured Storage: Sensitive data stored in an insecure manner, such as plain text or weakly encrypted format, can be easily accessed by attackers.
3. Directory Traversal: Attackers exploit directory traversal vulnerabilities to access files and directories outside of the intended scope, potentially exposing sensitive information.
4. SQL Injection: Attackers use SQL injection techniques to extract sensitive data from databases by manipulating input parameters.
5. Error Messages: Poorly crafted error messages or verbose responses may unintentionally reveal sensitive information about the system or its configuration.
6. Misconfigured Permissions: Improperly configured access permissions on files, directories, or databases can allow unauthorized users to view sensitive data.
7. Information Leakage in Headers: Sensitive information might be unintentionally leaked in HTTP headers or responses, providing attackers with valuable insights about the system.
8. Insecure APIs: Insecurely designed or exposed APIs may allow attackers to access sensitive data or perform unauthorized actions.
9. Data Leakage via Logs: Sensitive information might be logged without proper protection, exposing it to potential attackers with access to log files.

5. Denial of Service (DoS):
    - Consider scenarios where attackers could disrupt or degrade the availability of the system or its resources.
    - Look for potential vulnerabilities that could be exploited to overwhelm system resources or cause service interruptions.
    - Identify weak points in load balancing, resource management, and error handling.

Here are some specific threats that fall under the Denial-of-Service category:

1. Network Floods: Attackers flood the target network with a high volume of traffic, such as UDP, ICMP, or SYN packets, overwhelming the network infrastructure and causing it to become unresponsive.

2. Distributed Denial of Service (DDoS): Similar to network floods, DDoS attacks involve multiple compromised systems (botnets) coordinated to flood the target with massive amounts of traffic, amplifying the impact of the attack.
3. Application Layer DoS: Attackers exploit vulnerabilities in the application layer to consume excessive server resources, causing the application to become unresponsive or crash.
4. Zero-Day Exploits: Attackers leverage previously unknown vulnerabilities (zero-days) in the target system to cause crashes or instability, leading to a denial of service.
5. HTTP/S Request Flooding: Attackers flood a web server with a high volume of HTTP/S requests, overwhelming the server's capacity to respond to legitimate user requests.
6. DNS Amplification: Attackers exploit misconfigured DNS servers to send a large volume of DNS queries to the target, causing DNS resolution delays or failures.
7. Slowloris: Attackers exploit the way web servers handle simultaneous connections by sending partial HTTP requests at a slow rate, effectively tying up server resources and preventing new connections.

6. Elevation of Privilege:
    - Identify situations where attackers could gain higher privileges than they should have.
    - Consider potential weaknesses in privilege management, access controls, and user role assignments.
    - Think about scenarios where attackers could escalate their privileges through exploits.

Here are some specific threats that fall under the Elevation of Privilege category:

1. Privilege Escalation: Attackers exploit vulnerabilities in the system to elevate their privilege level from a lower-privileged user to a higher-privileged user, gaining access to administrative or superuser privileges.
2. Privilege Delegation: Attackers take advantage of misconfigured or poorly implemented privilege delegation mechanisms to obtain higher privileges than intended.
3. Impersonation: Attackers impersonate other users, services, or entities with higher privileges to gain unauthorized access to sensitive resources.
4. Directory Traversal: Attackers exploit directory traversal vulnerabilities to access files or directories outside of their intended scope, potentially obtaining higher privileges.
5. Exploiting Default Credentials: Attackers use default or weakly configured credentials to gain elevated access to the system.
6. Buffer Overflow: Attackers exploit buffer overflow vulnerabilities to execute arbitrary code and gain elevated privileges.
7. Injection Attacks: Attackers use injection techniques (e.g., SQL injection, command injection) to manipulate the system and obtain elevated privileges.
8. Insecure Access Control: Poorly implemented access control mechanisms may allow attackers to bypass restrictions and gain unauthorized access to privileged resources.
9. Backdoor Access: Attackers plant backdoors within the system, providing them with unauthorized access even after initial access has been removed.

• Security Checklists: Security checklists, based on industry best practices and security standards, can be used as a reference to identify common security issues and potential threats.

• Threat Libraries: Some organizations maintain threat libraries that include a collection of known threats and vulnerabilities, often curated from previous experiences or security research.

• Attack Surface Analysis: Evaluating the exposed attack surface of the system, including interfaces, APIs, and external dependencies, can help identify potential entry points for attackers.

• Attack Trees: Constructing attack trees, which depict various attack paths an attacker might take to achieve their objectives, can aid in the identification of threats and potential attack vectors.

Here are some common methods to rank threats in threat modeling:

- **DREAD** is a risk assessment framework used to rank and prioritize threats based on their potential impact and severity. The acronym DREAD stands for Damage, Reproducibility, Exploitability, Affected users, and Discoverability. Each aspect is assigned a score from 0 to 10, with 0 being the lowest impact or severity and 10 being the highest. The higher the overall DREAD score, the more critical the threat.

Here's how to use DREAD to rank threats in a threat model:

1. Damage (D): Assess the potential damage that the threat could cause if it is successfully exploited. Consider the impact on data confidentiality, integrity, and availability, as well as potential financial or reputational harm. Assign a score from 0 to 10, with 10 indicating the most severe damage.
2. Reproducibility (R): Evaluate how easily the threat can be reproduced or exploited repeatedly by an attacker. A threat with high reproducibility is more concerning as it can be exploited multiple times. Assign a score from 0 to 10, with 10 indicating high reproducibility.
3. Exploitability (E): Measure the difficulty or complexity for an attacker to exploit the vulnerability associated with the threat. A higher score indicates that the threat is easier to exploit. Assign a score from 0 to 10, with 10 indicating high exploitability.
4. Affected Users (A): Consider the number of users or entities impacted by the threat. A higher score is given if a large number of users are affected. Assign a score from 0 to 10, with 10 indicating a large number of affected users.
5. Discoverability (D): Evaluate the likelihood of the threat being discovered by an attacker. A higher score indicates that the threat is easier for attackers to find and exploit. Assign a score from 0 to 10, with 10 indicating high discoverability.

After assessing each aspect of DREAD and assigning the appropriate scores, calculate the overall DREAD score using the following formula:

Overall DREAD Score = (Damage + Reproducibility + Exploitability + Affected Users + Discoverability) / 5

• CVSS (Common Vulnerability Scoring System): CVSS is a standardized method for assessing and communicating the severity of vulnerabilities. It evaluates vulnerabilities based on factors like Access Complexity, Authentication requirements, and Impact (Confidentiality, Integrity, and Availability). The resulting CVSS score helps prioritize vulnerabilities and associated threats.

• OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation): OCTAVE is a risk assessment method that focuses on assets, threats, and vulnerabilities. It involves workshops and discussions with stakeholders to identify and rank threats based on their potential impact on critical assets.

4. – Mitigation (What are we going to do about it)

Implement the given recommendation.
Don’t Do Anything
Remove the features associated with it
Ture the feature off or reduces the functionality
Bring in code, Infrastructure or Design fix.

5. Validation (Did we do a good job!)

Here are some threat modeling tools:

• Microsoft **Threat Modeling Tool:** This is a widely used threat modeling tool that helps in creating threat models, identifying threats, and generating reports. It integrates well with Microsoft's secure development lifecycle (SDL) process.

• OWASP **Threat Dragon:** An open-source threat modeling tool developed by OWASP (Open Web Application Security Project). It allows you to create threat models using Data Flow Diagrams (DFDs) and provides a user-friendly interface.

• Astra**:** An open-source tool designed for threat modeling cloud-based applications and infrastructures.

• TMT **(Threat Modeling Toolkit):** A tool developed by OWASP that assists in creating and managing threat models.

• P.A.S.T.A. **(Process for Attack Simulation and Threat Analysis):** While not a specific tool, P.A.S.T.A. is a threat modeling methodology that guides organizations in identifying threats based on business impact.