|  |
| --- |
| **EXPERIMENT NO: 03** |
| **AIM: Exercise on threat modeling**  **DOP: DOS:** |

**THEORY:**

Threat modeling is a structured process for identifying and mitigating potential security risks in systems, applications, or even business processes. It involves proactively analyzing potential threats and vulnerabilities, assessing their potential impact, and developing strategies to reduce or eliminate those risks. Essentially, it's a way to think like an attacker to better protect your assets.

**Why is it important?**

* **Proactive Security:**

It allows you to identify and address security issues before they are exploited, reducing the risk of security breaches.

* **Informed Decision Making:**

Threat modeling provides valuable insights into potential risks, helping organizations make informed decisions about security investments and resource allocation.

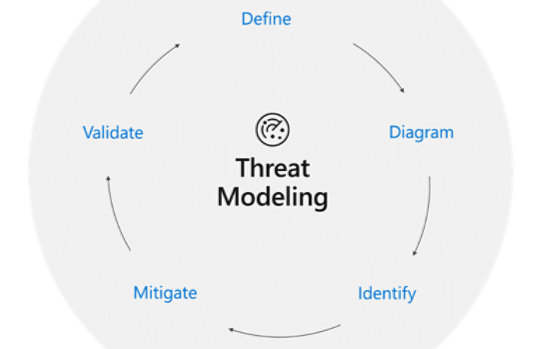
* **Improved Security Posture:**

By identifying and mitigating risks early on, threat modeling helps improve the overall security posture of an organization.

* **Cost-Effective Security:**

Addressing security issues during the design and development phases is often more cost-effective than fixing them after deployment.

Threat modeling is a valuable technique for any organization that wants to improve its security posture and protect its valuable assets.



**1. Define**

**What it means:** Decide *what* you’re protecting, *why*, and *how big* the project is.

* **Identify assets**: What’s valuable? (e.g., personal data, money, system uptime)
* **Define boundaries and entry points:** Where does data enter and leave the system? Where are the borders between “trusted” and “untrusted” zones?
* **Clarify objectives:** What are your goals? For example, “Ensure only authorized users can transfer funds.”  
  This first step is crucial—so you don’t waste time modeling irrelevant parts.

**2. Diagram**

**What it means:** Draw a visual map (typically a Data Flow Diagram or DFD) of your system’s parts and how data moves around.

* Show components like web servers, databases, APIs
* Mark data stores, flows, and users/services
* Add trust boundaries—lines that mark when data passes from safe to unsafe zones (e.g., internet → backend)  
  This diagram creates a “map” you can use to examine security gaps**.**

**3. Identify**

**What it means**: Look at your diagram and ask: “What could go wrong here?”

* **Use structured methods like STRIDE:**
  + **Spoofing** – impersonation
  + **Tampering –** data gets changed
  + **Repudiation –** denial of actions
  + **Information Disclosure –** data leaks
  + **Denial of Service –** system becomes unavailable
  + **Elevation of Privilege –** unauthorized power gains
* For each flow or component, run through these categories. Example: On a login page, could an attacker spoof credentials or tamper with data?
* Optionally, explore other frameworks like PASTA, Trike, or use abuse-case/misuse-case thinking

**4. Mitigate**

**What it means:** Decide how to reduce or remove each identified threat.

* **Options include:**
  + **Mitigate:** apply protections (e.g., encryption, filters, MFA)
  + **Eliminate:** remove risky features altogether
  + **Transfer:** share or outsource risk (e.g., using a secured third-party payment processor)
  + **Accept:** acknowledge low-risk threats and do nothing specific
* Each threat must have a clear action plan. For example, to block spoofing, require signed tokens and multi-factor authentication.

**5. Validate**

**What it means:** Check whether your solutions actually work and make your system safer.

* **Perform tests:** penetration testing, fuzzing, code reviews.
* **Check configurations:** are systems set up correctly?
* **Do regression testing:** ensure fixes don’t break anything else.
* **Iterate:** As your system changes or new vulnerabilities arise, update your diagram, re-identify threats, and repeat these steps.

**Threat Modeling Exercise (Web Application)**

**1. Define Scope & System Model**

* **Scenario:** A fintech startup launching an online payments web app.
* **Diagram:** Draw a Data Flow Diagram (DFD):
  + **Components**: frontend, auth service, database, third-party payment APIs, user.
  + **Trust boundaries**: public internet ↔ internal services.
  + Label data flows, data stores, processes.

**2. Threat Identification Using STRIDE**

Walk through each DFD element or interaction and ask “What could go wrong?” under each STRIDE category:

* **Spoofing:** e.g., stolen session tokens, account takeovers
* **Tampering:** e.g., transaction manipulation in transit
* **Repudiation:** e.g., missing audit logs to prove actions
* **Information Disclosure:** e.g., database leaks personal/financial data
* **Denial of Service:** e.g., brute-force auth causing outages
* **Elevation of Privilege:** e.g., exploiting APIs to grant admin access**.**

**3. Prioritize Threats (DREAD or PASTA)**

**Choose a scoring model:**

* **DREAD (Damage, Reproducibility, Exploitability, Affected users, Discoverability):**Score each threat 1–10 to calculate average risk.
* **PASTA:**
  1. Map business goals
  2. Define technical scope & DFD
  3. Analyze threats & vulnerabilities
  4. Simulate attacks
  5. Assess risk & impact
  6. Recommend mitigations

**4. Define Mitigations**

**For each high-priority threat, decide on controls:**

* Mitigate (e.g., enforce TLS, use MFA, validation, input sanitization)
* Eliminate (e.g., remove risky features)
* Transfer (e.g., outsourcing payment processing)

**5. Review & Iterate**

* Validate that mitigations cover threats.
* Update DFDs and threat list as system evolves.
* Schedule regular reviews each release cycle.

**Sample Threat: SQL Injection**

1. **Identify**
   * **Threat:** Tampering → SQL injection modifying financial records.
2. **Score (DREAD)**
   * **Damage =** 9, Reproducibility = 8, Exploitability = 7, Affected
   * **Users** = 8, Discoverability = 9 → Avg ≈ 8.2
3. **Mitigation**
   * Input validation, use parametrized queries, WAF in front.
4. **Review**
   * Retest after patches, ensure logging/auditing in place.

**CONCLUSION:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |