PASTA

(Process for attack simulation and threat analysis)

Risk centric framework for accessing security threats and Implementing counter measures.

It provides structured, attacker focussed approach to understand how an adversary would exploit a system and how to defend against it effectively.

Defend medival castle

① Define objective (Understand what we need to protect)

Identify the mission of the webapp — whether's it is handling payments, managing sensetive data, or providing public service.

② Define the Technical Scope (understanding the assets and attack surface)

what does castle consist of? — The castle walls
— The drawbridge
— The guard post

Identify components of the application such as web servers, databases, APIs and authentication mechanism

③ Decompose the application (Mapping the Inner structure)

How is castle is structured?

- How people can enter or exit?
- Where weak spots in the wall?

Breakdown the application's architecture - including data flows, user Interactio
and system dependencies to Identify potential weak points

④ Analyse Threats (understanding potential attacks)

Enemy soldiers might climb the walls

Sneak in disguised

Identify potential cyber threats such as SQL Injection, DDOS attack phishing & Insider threats.

⑤ Attack Modeling (Simulating how an attacker would act?)

* How would enemy attack?

Simulate real world attack scenarios such as brute force attack, social enginery or priviledge escalations.

⑥ Risk & Impact analysis (evaluating consequences)

* What happens if the caste is breached?

king is kidnapped

water suppy is poisned — —

Assess the business Impact - will a breach cause data leaks, financial loss or service downtime?

(7) Countermeasures and mitigation (Strengthening defenses)

* How we can ~~protect~~ protect our castle?

Implement Security controls such as:

1. Firewalls
2. Intrusion detection mechanism
3. MFA
4. Regular patch & code Reviews.

```
        self.vulnerabilities = vulnerabilities
        self.impact = impact
        self.countermeasures = countermeasures

    def stage_1_define_objectives(self):
        # Stage 1: Define objectives of the security process
        print(f"System: {self.system_info}")
        print(f"Objectives: {self.objectives}\n")

    def stage_2_threat_landscape_analysis(self):
        # Stage 2: Threat Landscape Analysis
        print("Identified Threats:")
        for threat in self.threats:
            print(f"- {threat}")
        print("\n")

    def stage_3_vulnerability_analysis(self):
        # Stage 3: Vulnerability Analysis
        print("Identified Vulnerabilities:")
        for vuln in self.vulnerabilities:
            print(f"- {vuln}")
        print("\n")

    def stage_4_attack_simulation(self):
        # Stage 4: Attack Simulation (Simple Risk Simulation based on impact and
likelihood)
```

→ Set the goal of security assessment

→ list all the potential threats

→ Discuss weak points in the system that can be exploited.

```
    def stage_4_attack_simulation(self):
        # Stage 4: Attack Simulation (Simple Risk Simulation based on impact and
likelihood)
        risk_scores = {}
        for threat in self.threats:
            risk = self.impact.get(threat, 0) * self.vulnerabilities.get(threat, 0)
            risk_scores[threat] = risk

        print("Risk Assessment based on Threats and Vulnerabilities:")
        for threat, risk in risk_scores.items():
            print(f"Threat: {threat}, Risk Score: {risk}")
        print("\n")
```

← Risk score = Impact x vulnerability score

SQL Inject = 9 x 8 = 72

More score means more security concern

```
    def stage_5_risk_impact_analysis(self):
        # Stage 5: Risk and Impact Analysis
        total_risk = sum([self.impact.get(threat, 0) * self.vulnerabilities.get(threat, 0) for
threat in self.threats])
        print(f"Total Risk Score: {total_risk}")

    def stage_6_countermeasure_recommendation(self):
        # Stage 6: Suggest countermeasures based on vulnerabilities
        print("Recommended Countermeasures:")
        for vuln in self.vulnerabilities:
            print(f"- {self.countermeasures.get(vuln, 'No countermeasure available')}")
        print("\n")

    def stage_7_monitoring_and_improvement(self):
        # Stage 7: Continuous Monitoring (Simulation)
        print("Implementing continuous monitoring...")
        # In real-world scenarios, this might involve monitoring logs, attacks, and
updates.
        print("Monitoring system for new threats and vulnerabilities.\n")


# Example of system data and PASTA parameters
def main():
    system_info = "Web Application Server"
    objectives = "Ensure confidentiality, integrity, and availability of user data."

    threats = ["SQL Injection", "Cross-Site Scripting (XSS)", "Denial of Service (DoS)"]
    vulnerabilities = {
        "SQL Injection": 8,
        "Cross-Site Scripting (XSS)": 6,
```

→ Calculate total risk score

→ Security controls for each vulnerability

ensures ongoing security improvement by monitoring logs & updating defenses.