**Lesson 01 Demo 02**

**Implementing PASTA Model**

**Objective:** To implement the PASTA (Process for Attack Simulation and Threat Analysis) model for assessing risks and recommending counter measures

**Tools required:** Python

**Prerequisites:** None

Steps to be followed:

1. Implement the PASTA model using a Python script

**Step 1: Implement the PASTA model using a Python script**

1. Open the **AWS Management Console** in your practice environment

A screenshot of a computer

Description automatically generated

1. Click on the cloud shell icon for the terminal  
     
   A screenshot of a computer

   Description automatically generated
2. Run the following command to verify if Python is installed:

**python3 --version**

A blue screen with white text

Description automatically generated

1. Execute the commands given below to create and navigate to the directory:

**mkdir python**

**cd python**  
A blue square with white lines

Description automatically generated with medium confidence

1. Run the following command to create a file:  
   **nano pasta\_model.py**

A blue background with white text

Description automatically generated

* 1. Once the file opens, add the code given below, save it, and then exit:

**class PASTA:**

**def \_\_init\_\_(self, system\_info, objectives, threats, vulnerabilities, impact, countermeasures):**

**# System and objectives definition**

**self.system\_info = system\_info**

**self.objectives = objectives**

**self.threats = threats**

**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)**

**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")**

**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,**

**"Denial of Service (DoS)": 7**

**}**

**impact = {**

**"SQL Injection": 9,**

**"Cross-Site Scripting (XSS)": 5,**

**"Denial of Service (DoS)": 7**

**}**

**countermeasures = {**

**"SQL Injection": "Use parameterized queries and ORM.",**

**"Cross-Site Scripting (XSS)": "Sanitize and validate user input.",**

**"Denial of Service (DoS)": "Use rate limiting and DDOS protection."**

**}**

**# Instantiate PASTA model**

**pasta\_model = PASTA(system\_info, objectives, threats, vulnerabilities, impact, countermeasures)**

**# Run through the stages of the PASTA model**

**pasta\_model.stage\_1\_define\_objectives()**

**pasta\_model.stage\_2\_threat\_landscape\_analysis()**

**pasta\_model.stage\_3\_vulnerability\_analysis()**

**pasta\_model.stage\_4\_attack\_simulation()**

**pasta\_model.stage\_5\_risk\_impact\_analysis()**

**pasta\_model.stage\_6\_countermeasure\_recommendation()**

**pasta\_model.stage\_7\_monitoring\_and\_improvement()**

**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

**A computer screen shot of a program code

Description automatically generated**

**Note**: To save and exit the script, use CTRL + O >> press enter >> CTRL + X

This script offers a practical implementation of the PASTA model. It simulates all seven stages, providing a hands-on approach to understanding and applying threat analysis frameworks in cybersecurity.

* 1. Once the Python script is created, run it using the following command:

**python3 pasta\_model.py**

A screen shot of a computer

Description automatically generated

By following these steps, you have successfully implemented the PASTA model to assess risks and recommend countermeasures.