**full classroom exercise guide** for using **OWASP Threat Dragon** with a complete example for **Online Banking Login System** using the **STRIDE methodology**, including:

* Tool introduction
* Adding contributors/reviewers
* Creating and naming the threat model and diagram
* Building the diagram
* Applying STRIDE with multiple threats per element
* Sample threat details with scores, context, and descriptions
* Export/report guidance

**🧑‍🏫 Classroom Exercise Guide**

**🌐 Threat Modeling with OWASP Threat Dragon (STRIDE)**

**🧩 OBJECTIVES**

* Learn to use OWASP Threat Dragon for threat modeling
* Understand STRIDE categories
* Create and model a real-world system
* Identify threats across all STRIDE categories
* Assign scores and document threat details
* Collaborate with contributors and reviewers

**✅ PART 1: Getting Started**

**Step 1: Access the Tool**

1. Visit: <https://threatdragon.org>
2. Choose **Web Application (hosted)**.
3. Sign in with GitHub or use **Local Storage** for temporary modeling.
4. Click **“New Model”** to begin.

**✅ PART 2: Fill Project Info & Add Diagram**

**Step 2: Fill Project Metadata**

| **Field** | **Input** |
| --- | --- |
| **Model Name** | Online Banking Login System |
| **Model Description** | Threat model for the login system of a typical online banking application. Uses STRIDE to identify threats to authentication, data transmission, and user session management. |
| **Contributors** | Usama Wahab Khan, [Student Names] |
| **Reviewers** | [Instructor Name], [Peer Names] |

✅ Click **Create Model**

**Step 3: Add a Diagram**

Once the model is created:

1. Click on **➕ Add Diagram**
2. Fill in the following:

| **Field** | **Input** |
| --- | --- |
| **Diagram Name** | Online Banking STRIDE Diagram |
| **Diagram Description** | Visual threat model of a secure online banking login system, showing user interaction, credential flow, backend processing, and trust boundaries. |

✅ Click **Create Diagram**  
You’ll now be taken to the **diagram editor**.

**✅ PART 3: Diagram Construction**

**Step 4: Build the Threat Model**

Use the left sidebar to drag elements onto the canvas.

**Add the following components:**

| **Type** | **Label** | **Notes** |
| --- | --- | --- |
| External Entity | User | Outside trust boundary |
| Process | Login Page | Accepts user credentials |
| Data Flow | User Credentials | User → Login Page |
| Trust Boundary | Internet Boundary | Around user and login page |
| Process | Auth Service | Validates credentials, issues token |
| Data Store | User Database | Stores usernames, passwords, roles |
| Data Flow | Token | Auth Service → Browser |
| Data Store | Session Store | Stores issued tokens |

📌 Connect the elements logically using arrows. Make sure to run lines across trust boundaries to simulate real-world security scopes.

**✅ PART 4: Apply STRIDE Threat Modeling**

**STRIDE Categories**

| **Category** | **Description** |
| --- | --- |
| **S** | Spoofing (identity) |
| **T** | Tampering with data |
| **R** | Repudiation (no traceability) |
| **I** | Information Disclosure |
| **D** | Denial of Service |
| **E** | Elevation of Privilege |

**Step 5: Add Threats to Each Element**

**Instructions:**

1. Right-click an element → **Edit Threats**
2. Click **“Add Threat”**
3. Fill in threat details for each STRIDE category.

**✅ PART 5: Threat Examples by Element**

**🟦 Element: Login Page (Process)**

| **STRIDE** | **Title** | **Description** | **Score** |
| --- | --- | --- | --- |
| S | Credential Spoofing | Attacker reuses stolen credentials | High |
| T | Client-Side Tampering | JS injection alters logic | Medium |
| R | Failed Login Not Logged | No logging for failed attempts | Medium |
| I | Autofill Info Leak | Browser saves usernames insecurely | Low |
| D | Brute Force Login | Multiple login attempts crash server | High |
| E | Login Bypass Flaw | Parameter tampering bypasses login | Critical |

**🟧 Element: User Credentials (Data Flow)**

| **STRIDE** | **Title** | **Description** | **Score** |
| --- | --- | --- | --- |
| T | MITM Tampering | Data altered during transit | High |
| I | Plaintext Transmission | Credentials sent without HTTPS | Critical |
| D | Request Flooding | Excess POST requests overwhelm endpoint | Medium |

**🟨 Element: User Database (Data Store)**

| **STRIDE** | **Title** | **Description** | **Score** |
| --- | --- | --- | --- |
| S | Account Enumeration | Timing attack reveals usernames | Medium |
| T | SQL Injection | Input manipulates DB queries | Critical |
| I | Database Dump Leak | Backup file exposes PII | Critical |
| D | Table Lock Attack | Repeated writes block access | High |

**🟩 Element: Trust Boundary**

| **STRIDE** | **Title** | **Description** | **Score** |
| --- | --- | --- | --- |
| I | Unvalidated Input | User input reaches trusted backend unchecked | High |
| T | Cross-site Script Attack | XSS travels across boundary | Medium |

**🟦 Element: Auth Service (Process)**

| **STRIDE** | **Title** | **Description** | **Score** |
| --- | --- | --- | --- |
| S | Session Hijacking | Token reuse by attacker | High |
| T | Token Logic Flaw | Forged token bypasses auth | Medium |
| D | Token Replay | Captured token reused after expiry | High |
| E | Privilege Escalation via Role Injection | Normal user gets admin rights | Critical |

**✅ PART 6: Add Collaborators**

**Step 6: Add Contributors & Reviewers**

1. Click **Model Overview**
2. In metadata, enter:
   * **Contributors**: List all team members
   * **Reviewers**: Enter instructor or peer names

💡 This ensures credit and collaborative accountability.

**✅ PART 7: Review and Export**

**Step 7: Generate Reports**

1. Click **Report**
2. Choose:
   * **PDF** – for printable documentation
   * **Markdown** – for GitHub/technical sharing

📝 Reports include:

* Model overview
* Full threat list with scores
* Diagram snapshots
* Contributors/reviewers

**🧠 Bonus: Contextual Threats (Real-World Layer)**

| **Context** | **Threat Description** | **STRIDE** | **Score** |
| --- | --- | --- | --- |
| Public Wi-Fi | MITM attack without SSL | I, T | High |
| Shared Devices | Cached data exposes login | I | Medium |
| Social Engineering | Phishing email tricks user | S | High |
| Browser Plugins | Malicious extension steals tokens | I, S | High |
| Cloud Misconfiguration | S3 or storage bucket leaks session data | I, T | High |