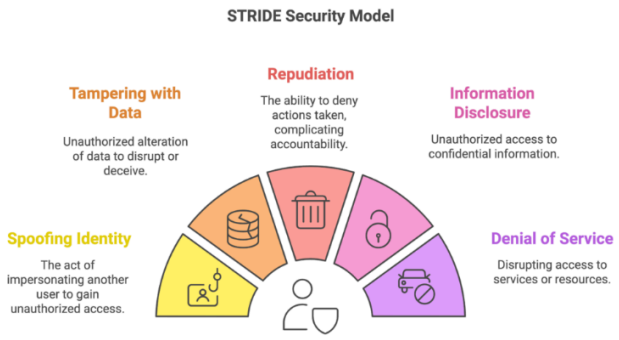
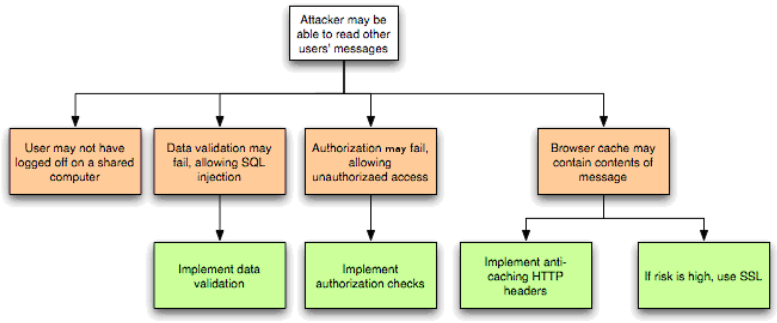
1. **Threat Modeling**
    a. **Classify Attacker Knowledge and Access Levels:**
        i. Black-box: Attackers have no knowledge of the system's internal workings.
        ii. Gray-box: Attackers have partial knowledge of the system, such as some understanding of its architecture.
        iii. White-box: Attackers have full access to and knowledge of the system's internals.
    b. **Identify Critical Assets:** Determine essential components such as:
        i. Models: The algorithms and structures used for processing and decision-making.
        ii. Data: The datasets that the models rely on.
    c. **Assess Failure Modes and Trust Boundaries:**
        i. Analyze how and where failures might occur within the system and identify the boundaries that define trust within these interactions.
    d. **Systematize Attack Vectors:** Categorize potential methods of attack, including:
        i. Adversarial Examples:
            1. Inputs designed to deceive the model.
    e. **Assess Impact, Exploitability, and Detectability:** Evaluate factors based on:
        i. Attacker Capability:
            1. The skills and resources available to the attacker.
        ii. Defender Mitigations:
            1. The security measures and protocols that the defenders have implemented.
2. **Popular Frameworks & Resources**
    a. STRIDE
        i. Source: https://learn.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20)?redirectedfrom=MSDN
        ii. Supplement: https://www.practical-devsecops.com/what-is-stride-threat-model/

Stride full form:

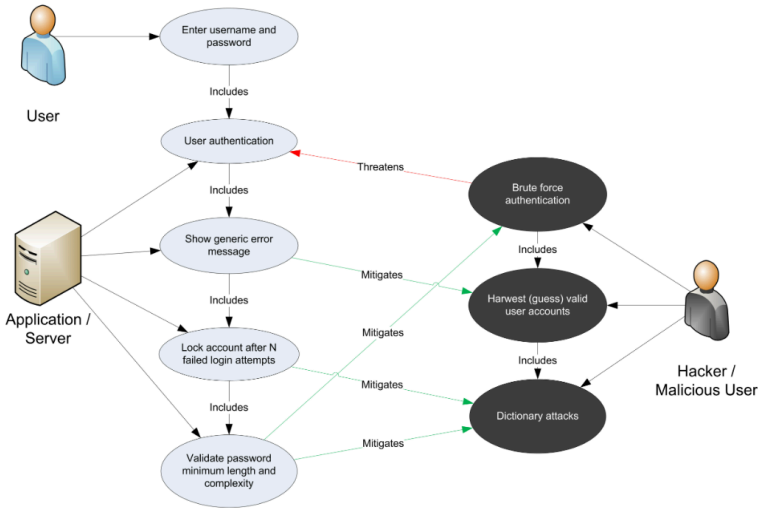
The full form of "STRIDE" in the context of security is a mnemonic representing a model used to identify computer security threats. It stands for:



b. OWASP
   i. Source: https://owasp.org/www-community/Threat_Modeling_Process
   ii. Supplement (Machine Learning Security Top10) https://owasp.org/www-project-machine-learning-security-top-10/docs/ML01_2023-Input_Manipulation_Attack.html

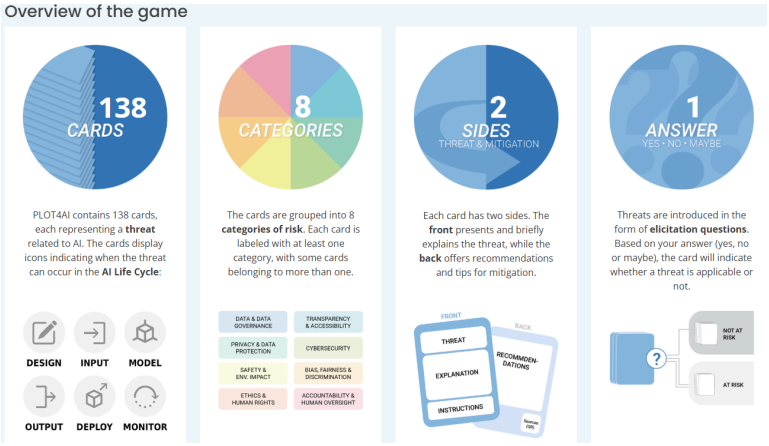


1. *Figure 3: Threat Tree Diagram.*



2. *Figure 4: Use and Misuse Cases*

c. PLOT4AI
   i. Source: https://plot4.ai/about
   ii. Supplement: https://plot4.ai/how-does-it-work



3. **Example Attacker Capabilities and Constraints Table**
    a.

Threat Model	Attacker Knowledge Level	Capabilities	Constraints
Black-box	No internal knowledge	Have limited code samples of own work and others	No model/data access
Gray-box	Partial knowledge	Expanded data access. Insights into defender approach. Can produce a surrogate model	Partial data/model access only
White-box	Full internal knowledge	Can modify model, data, tailor attacks optimally	Insider access required

4. **Adversarial Stylometry Ideas**
    a. Obfuscations
        i. **WHY**?
            1. Transformations applied to source code must mislead distinctive stylistic features without altering the intended functional output (semantic clones)
                a. (**Easy**) Evade verification (binary) detection
                b. (**Easy**) Evade attribution (multi-class) detection
                c. (**Moderate**) Evade verification detection & Evade obfuscation event detection
                d. (**Moderate**) Evade attribution detection & Evade obfuscation event detection
                e. (**Hard**) Evade attribution detection & Evade obfuscation event detection & Mislead detection to predict a target label
        ii. **HOW**?
            1. **Black box** (manually applied or via AI tools, etc.)
                a. Layout
                    i. Modify whitespace, indentation, bracket position, comment format, or line breaks.
                b. Lexical
                    i. Rename variables, functions, and classes in characteristic or randomized ways; alter literal values (e.g., representing booleans as 1/0, introducing redundant code)
                c. Syntactic

            i. Rearranging control structures (e.g., converting for-loops to while-loops), splitting/merging statements, or inserting function wrappers for simple logic.
        d. Semantic
            i. Insert semantically redundant or dead code, alter order of computations, or modularize code to disrupt pattern recognition while preserving output.
        e. Others
    2. **Gray box** (previous plus)
        a. Background on the potential approach taken by the defender
            i. Model: Random Forest, Neural Network
            ii. Data: Certain people or repositories, vectorization, etc
            iii. Evaluation: performance metric, optimizations, etc.
        b. Surrogate model creation and/or analysis
            i. Offline research to isolate unique stylistic markers likely associated with each author we can analyze
        c. Perform calibrated obfuscations based on results
            i. Minimize our stylistic markers, maximize targets, avoid excessive and/or obvious changes (less is more), etc.
    3. **White box** (previous plus)
        a. Full access to all information that could be helpful
b. Stylistic Suppression Concepts
    i. Can be done at any access level
    ii. Access level improves the likelihood of obfuscation event detection
c. Mimicry Concepts
    i. Need sample access (improves with access level)
d. AVOID: Data Poisoning Attack Concepts
    i. Access to the model training (train model on bad samples, etc.) out of scope for this project.

5. Experimental Setup

a. Following the Threat model, design and create set of experiments to conduct
b. Pipeline for efficient experimental execution (modular code, etc)

6. Experiments

a. Conduct and refine experiments to create the best adversarial samples possible (per the objective of each experiment)

PHASE 2: (Use adversarial samples for defender training, re-evaluate, etc.)