Project Title: Adversarial AI Attack Generator Against Machine-Learning Firewalls

---

Core Idea:

- Build a system where **AI models generate adversarial network traffic** designed specifically to trick and bypass **ML-based firewalls or anomaly detection systems**.

Component	Role
Firewall Behavior Profiler	Models how ML-based firewalls classify traffic
Adversarial Traffic Generator	Crafts minimal changes that evade ML detection
Reinforcement Learning Trainer	Evolves better evasion strategies over time
Traffic Emitter/Tester	Sends crafted traffic and observes firewall reactions
Success Scorer Module	Measures how stealthy/adversarial samples perform

---

Component Details:

1. **Firewall Behavior Profiler**:
    - Passive/active techniques:
        - Probe small changes and measure responses (accept/reject/alert).
    - Builds approximate ML decision boundary models.
2. **Adversarial Traffic Generator**:
    - Crafting techniques:
        - **Feature Space Perturbations**:
            - Alter statistical features (packet sizes, flow durations, etc) without breaking protocol compliance.
        - **Gradient Attack Simulation**:
            - Estimate firewall's decision gradient to create evasive samples.
        - Etc
3. **Reinforcement Learning Trainer**:
    - Reinforces traffic mutations that:
        - Pass detection,
        - Appear benign.
4. **Traffic Emitter/Tester**:
    - Sends adversarial samples.
    - Measures firewall verdicts.
5. **Success Scorer Module**:
    - Tracks:
        - Stealth success rate.
        - Volume of successful bypasses over time.

Overall System Flow:

- Input: ML firewall target
- Output: Evolving adversarial traffic able to bypass it
- Focus: **Adversarial machine learning applied to network traffic evasion**

Internal Functioning of Each Module:

1. Firewall Behavior Profiler

- **Probing**:
  - Send carefully mutated benign-looking traffic.
  - Measure:
    - Accept rates,
    - Drop rates,
    - Latency changes.
- **Model firewall's decision boundary**:
  - Use binary classification modeling:
    - "Accept" = Class 0,
    - "Reject" = Class 1.

---

2. Adversarial Traffic Generator

- **Generation strategies**:
  - **Feature perturbation**:
    - Slightly modify packet size distributions, flow timing, etc.
  - **Decision boundary attacks**:
    - Apply gradient descent-inspired methods:
      - Small changes that "flip" classification.

---

3. Reinforcement Learning (RL) Trainer

- **RL setup**:
  - States: Traffic features.
  - Actions: Mutation operations.
  - Rewards:
    - +1 if traffic passes undetected,
    - -1 if blocked.
- **Exploration vs exploitation**:
  - Epsilon-greedy exploration for novel evasion tactics.

---

4. Traffic Emitter/Tester

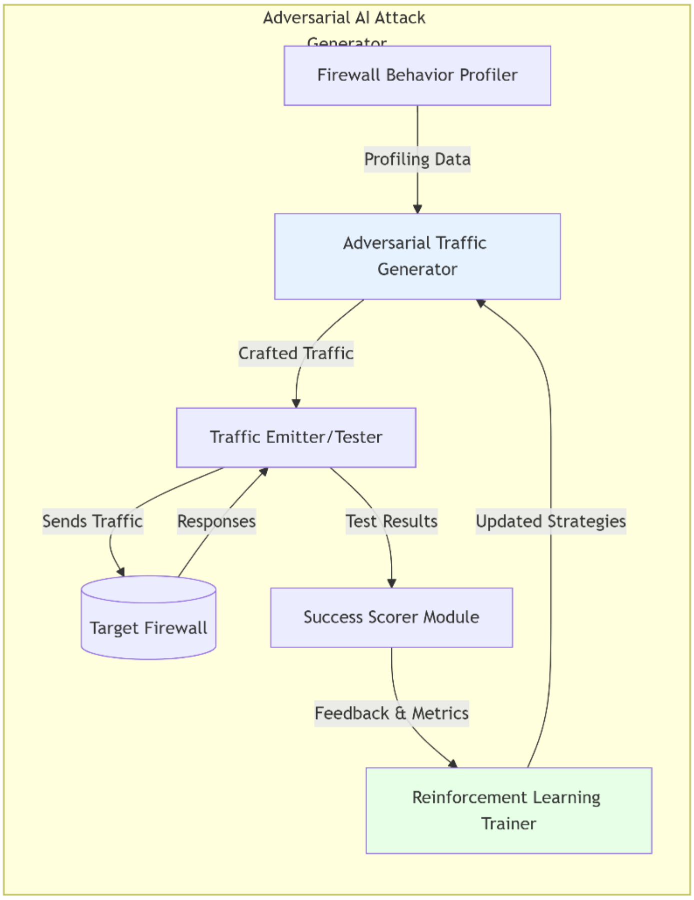
- **Test cycle**:
    - Send modified traffic,
    - Monitor firewall verdicts,
    - Feed results back into RL agent.

---

5. Success Scorer Module

- **Metrics**:
    - Stealth rate (percent undetected),
    - Transfer rates,
    - Learning curve slope.

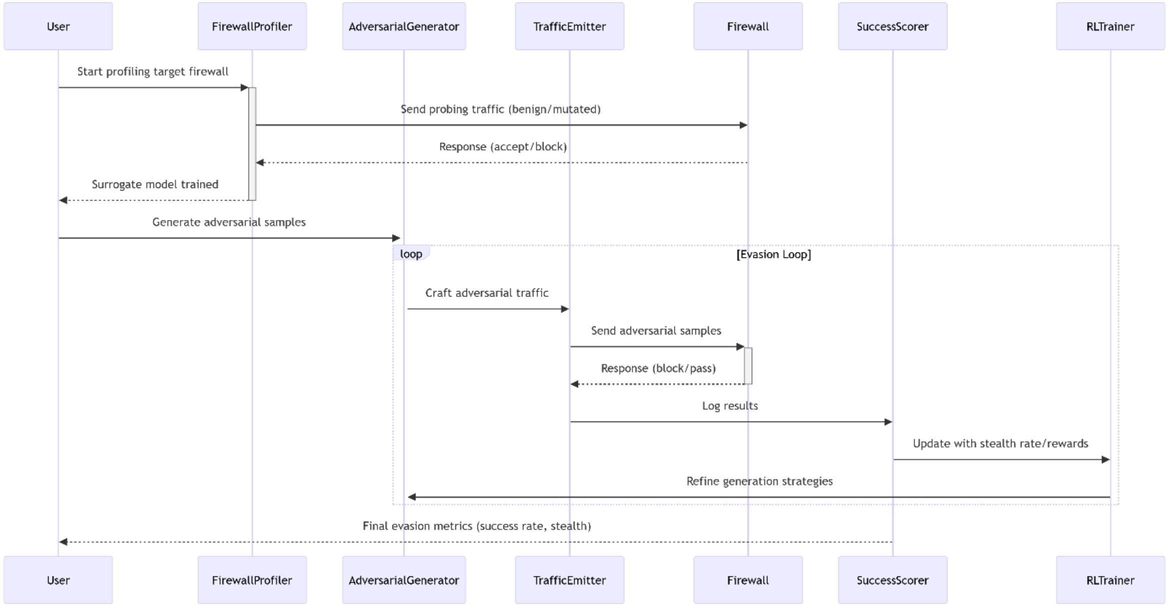
Component Diagram



Explanation:

1. **Components**:
   - **Firewall Behavior Profiler**: Probes the target firewall to model its ML decision boundaries (e.g., using surrogate models like SVM/Random Forest, etc).
   - **Adversarial Traffic Generator**: Crafts evasive traffic via feature perturbations (e.g., packet size tweaks, etc) or gradient-based attacks (e.g., FGSM, etc).
   - **Traffic Emitter/Tester**: Sends adversarial traffic to the firewall and logs responses (blocks/passes).
   - **Success Scorer Module**: Evaluates stealth rate and transferability of attacks.
   - **Reinforcement Learning Trainer**: Uses feedback (rewards/penalties) to refine evasion strategies (e.g., via PPO or Q-learning).
2. **Workflow**:
   - **Profiling**: The Firewall Profiler sends probing traffic to infer the firewall's ML model.
   - **Traffic Generation**: The Adversarial Generator crafts traffic to bypass learned decision boundaries.
   - **Testing**: The Emitter sends traffic to the firewall and records outcomes.
   - **Feedback Loop**: The Success Scorer sends metrics to the RL Trainer, which updates strategies for the Generator.
3. **Key Features**:
   - **Adaptive Evasion**: RL-driven iterative improvement of adversarial samples.
   - **Black-Box Attacks**: Surrogate models mimic target firewall behavior for gradient estimation.
   - **Protocol Compliance**: Validates traffic functionality (e.g., valid TCP handshakes, etc).

Sequence Diagram



Explanation:

1. **Firewall Profiling**:
    - The **User** starts the **Firewall Behavior Profiler**, which sends benign and mutated traffic to the target firewall.
    - The firewall's responses (accept/block) are used to train a surrogate model that approximates its decision boundaries.
2. **Adversarial Traffic Generation**:
    - The **Adversarial Traffic Generator** crafts samples using feature perturbations (e.g., altering packet sizes, etc) or gradient-based attacks (e.g., FGSM, etc).
3. **Testing & Feedback Loop**:
    - The **Traffic Emitter** sends adversarial traffic to the firewall and logs responses.
    - The **Success Scorer** evaluates evasion effectiveness (i.e. stealth rate) and feeds results to the **Reinforcement Learning Trainer**.
    - The **RL Trainer** updates strategies (e.g., PPO policies, etc) and refines the **Adversarial Generator**'s tactics.
4. **Iteration**:

- The loop continues until adversarial samples consistently bypass the firewall or meet success criteria.

Detailed Project Description: Adversarial AI Attack Generator Against ML Firewalls

This system generates adversarial network traffic to bypass machine learning (ML)-based firewalls. By probing firewall behavior, crafting evasive samples using reinforcement learning (RL), and iteratively refining attacks, this system identifies vulnerabilities in ML-driven security systems.

---

1. Core Components & Implementation Details

1.1 Firewall Behavior Profiler

- **Role**: Model the target firewall's decision boundaries.
- **Implementation (e.g.)**:
  - **Active Probing**:
    - Send benign and slightly perturbed traffic (e.g., altered packet sizes, flow durations) using Scapy.
    - Record firewall responses (accept/block) to infer classification thresholds.
  - **Decision Boundary Modeling**:
    - Train a surrogate model (e.g., Random Forest, SVM, etc) to mimic the firewall's behavior using probed data.
- **Tools (e.g.)**: Scapy, Scikit-learn, Wireshark (traffic analysis), etc.

1.2 Adversarial Traffic Generator

- **Role**: Craft traffic that evades detection.
- **Implementation (e.g.)**:
  - **Feature Perturbation**:
    - Modify statistical features (packet size variance, flow duration) while maintaining protocol compliance.
  - **Gradient-Based Attacks**:

        - Use adversarial ML libraries (e.g., ART, etc) to apply FGSM or PGD on traffic features.
    - **Functional Validity**:
        - Validate traffic with protocol conformance tests (e.g., TCP handshake completion, etc).
- **Tools (e.g.)**: ART (Adversarial Robustness Toolbox), Scapy, Netcat, etc.

1.3 Reinforcement Learning Trainer

- **Role**: Optimize adversarial strategies through trial and error.
- **Implementation (e.g.)**:
    - **RL Environment**:
        - **States**: Feature vectors (packet size, flow duration, etc.).
        - **Actions**: Perturbation operations (e.g., ±10% packet size).
        - **Rewards**: +1 for evasion, -1 for detection.
    - **Agent Training**:
        - Use Proximal Policy Optimization (PPO) or Q-learning with epsilon-greedy exploration.
- **Tools (e.g.)**: OpenAI Gym (custom environment), Stable Baselines3 (RL algorithms), etc.

1.4 Traffic Emitter/Tester

- **Role**: Test adversarial samples against the firewall.
- **Implementation (e.g.)**:
    - **Packet Injection**: Send crafted traffic using Scapy or NFQUEUE.
    - **Response Monitoring**: Detect blocks via TCP RST, ICMP errors, or timeouts.
    - **Feedback Loop**: Log results (success/failure) for RL training.
- **Tools (e.g.)**: Scapy, Python threading (parallel testing), etc.

1.5 Success Scorer Module

- **Role**: Quantify evasion effectiveness.
- **Implementation (e.g.)**:
    - **Metrics**:

        - **Stealth Rate**: Percentage of undetected adversarial samples.
        - **Transferability**: Success rate against updated firewall models.
    - **Visualization**: Plot learning curves and feature distributions.
- **Tools (e.g.)**: Matplotlib, TensorBoard, Pandas, etc.

---

2. System Workflow

1. **Probe Firewall**: Profile decision boundaries by sending test traffic.
2. **Train Surrogate Model**: Approximate the firewall's ML model.
3. **Generate Adversarial Traffic**: Perturb features or apply gradient attacks.
4. **Test & Score**: Send traffic, log results, and calculate stealth metrics.
5. **RL Training**: Update agent policies based on rewards/penalties.
6. **Iterate**: Repeat until evasion success stabilizes.

---

3. Evaluation Metrics

- **Evasion Success Rate**: % of adversarial samples bypassing the firewall.
- **Feature Drift**: KL divergence between adversarial and legitimate traffic.
- **Convergence Time**: Iterations needed to achieve 90% stealth rate.

---

4. Tools & Frameworks (e.g.)

- **Traffic Crafting**: Scapy, NetfilterQueue, etc.
- **Adversarial ML**: ART, CleverHans, etc.
- **Reinforcement Learning**: Stable Baselines3, OpenAI Gym, etc.
- **Analysis**: Wireshark, Scikit-learn, Matplotlib, etc.

---

5. Suggested Implementation Steps (e.g.)

1. **Setup Environment**:
    - Deploy an ML-based firewall (e.g., using Snort with ML plugins).
    - Install Python dependencies (Scapy, ART, Stable Baselines3, etc).
2. **Profile Firewall**:
    - Send benign/malicious traffic to train a surrogate model.
    - Identify critical features (e.g., packet size, flow duration, etc).
3. **Build Adversarial Generator**:
    - Implement feature perturbations and gradient-based attacks.
    - Validate traffic with protocol checks (e.g., valid HTTP headers).
4. **Integrate RL Training**:
    - Define custom Gym environment for traffic mutation.
    - Train RL agent with PPO, using evasion success as reward.
5. **Test & Optimize**:
    - Run iterative cycles: generate → test → score → refine.
    - Tune RL hyperparameters (learning rate, exploration rate, etc).

---

6. Challenges & Mitigations (optional)

- **Black-Box Firewall**: Use transfer learning to adapt surrogate models.
- **Traffic Validity**: Integrate protocol validators (e.g., PyShark).
- **Dynamic Firewalls**: Periodically retrain surrogate models.
