Project Title: LLM-based Cyber Deception and Trap System (AI Honeypots)

Core Idea:

- Use LLMs to **design adaptive cyber-deception environments**.
- Fake systems (honeypots, honeytokens, honey APIs) that **change intelligently** based on attacker behavior.

Component	Role
Deception Surface Planner (LLM)	Designs fake systems tailored to environment threats
Adaptive Honeytoken Engine	Deploys fake credentials, fake databases, fake API keys, etc
Attacker Interaction Analyzer	Profiles attacker techniques and adapts deception
LLM Response Simulator	Creates realistic dynamic responses from fake systems
Deception Metrics Engine	Measures engagement, attack delays, attacker confusion

---

Component Details:

1. **Deception Surface Planner (LLM)**:
   - Reads live attack trends.
   - Creates believable fake services (e.g., fake Jenkins server, vulnerable web login, etc).
2. **Adaptive Honeytoken Engine**:
   - Deploys fake assets inside systems, repositories, emails.
3. **Attacker Interaction Analyzer**:
   - Captures attacker sessions.
   - Classifies attacker behavior:
     - Reconnaissance
     - Credential harvesting
     - Exploitation attempts
     - Etc
   - Etc
4. **LLM Response Simulator**:
   - Provides fake but believable system responses to attackers.
5. **Deception Metrics Engine**:
   - Calculates:
     - Average attacker engagement time.
     - Paths attackers take inside fake systems.
     - Etc.

Overall System Flow:

- Input: Environment intelligence
- Output: Active, adaptive deception surfaces
- Focus: **Waste attacker time, gather intelligence, reduce real-world breach risk**

Internal Functioning of Each Module:

1. Deception Surface Planner (LLM)

- **Inputs**:
    - Network map, technology stack, threat intelligence feeds.
- **Process**:
    - LLM plans:
        - Fake high-value targets based on environment (e.g., fake admin portals if web stack is strong, etc).
    - Chooses deception mix:
        - Fake databases
        - Fake file servers
        - Honey APIs (e.g., OAuth servers)
        - Etc

---

2. Adaptive Honeytoken Engine

- **Deployment**:
    - Injects fake credentials into codebases, Git repositories, databases, etc.
    - Honeytokens can trigger alerts if used.

---

3. Attacker Interaction Analyzer

- **Tracking**:
    - Captures:
        - Commands typed (e.g., via honeypot SSH)
        - Files accessed
        - API calls made
        - Etc
- **Classification**:
    - Categorizes attacker behavior:
        - Recon
        - Privilege escalation
        - Lateral movement

- Etc

---

4. LLM Response Simulator

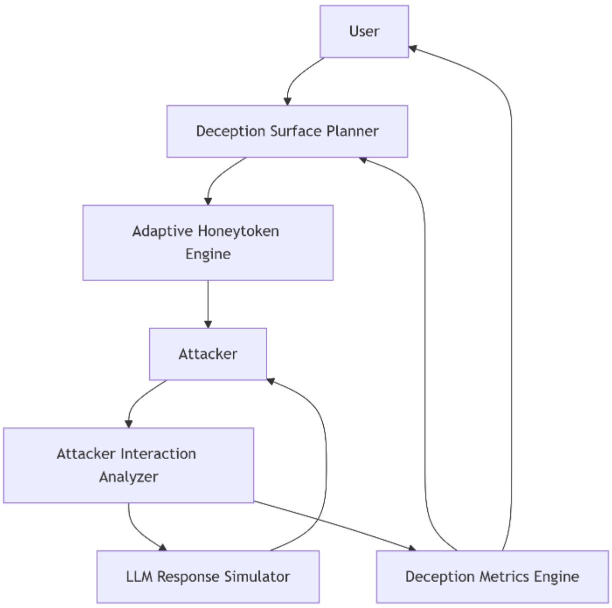
- **Simulation**:
  - Dynamically generate fake outputs.
  - Example:
    - For SQL queries:
      `"SELECT * FROM users"` ➔ `Return fake-looking records.`
    - For file browsing:
      `ls -al /etc` ➔ `Show fake configs.`
- **Behavior**:
  - Context-aware responses to maintain realism.

---

5. Deception Metrics Engine

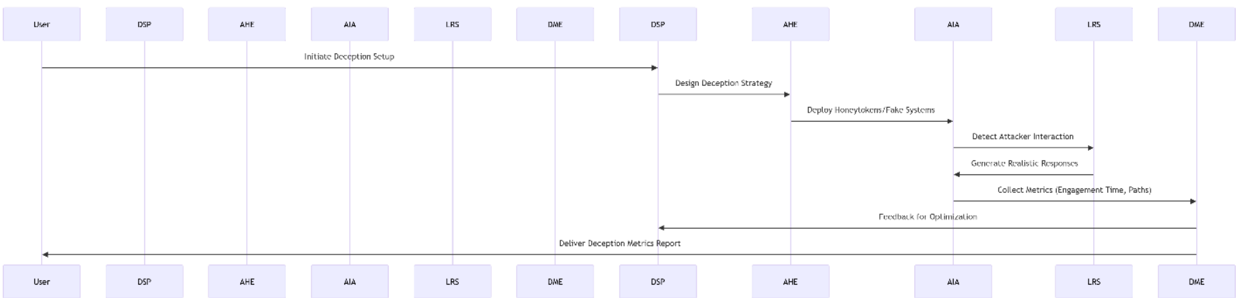
- **Analysis**:
  - Measure:
    - Average attacker time inside honeypot
    - Steps before detection
    - Paths taken
    - Etc
  - Compare across campaigns to optimize deception strategies.

Component Diagram



- **Flow Top to Bottom:**
    1. **User** starts the process.
    2. **Deception Surface Planner (LLM)** designs fake systems/honeypots.
    3. **Adaptive Honeytoken Engine** deploys fake credentials and assets.
    4. **Attacker** interacts with the deceptive environment.
    5. **Attacker Interaction Analyzer** captures and classifies attacker behavior.
    6. **LLM Response Simulator** generates dynamic responses to keep the attacker engaged.
    7. **Deception Metrics Engine** analyzes data, feeds insights back to the Planner, and reports to the User.
- **Key Feedback Loops:**
    - Metrics Engine → Planner: Optimizes future deception strategies.
    - Metrics Engine → User: Provides actionable reports.
    - Simulator → Attacker: Maintains engagement through adaptive responses.

Sequence Diagram



1. **User** initiates the deception setup.
2. **Deception Surface Planner** designs fake systems and forwards the strategy to the **Adaptive Honeytoken Engine**.
3. **Honeytoken Engine** deploys fake assets (credentials, databases) into the environment.
4. **Attacker Interaction Analyzer** detects attacker activity and triggers the **LLM Response Simulator** to generate believable responses.
5. **Response Simulator** dynamically interacts with attackers to prolong engagement.

6. **Attacker Interaction Analyzer** sends interaction data to the **Deception Metrics Engine** for analysis.
7. **Metrics Engine** feeds optimization insights back to the **Deception Surface Planner** and delivers a report to the **User**.

Detailed Project Description: LLM-based Cyber Deception and Trap System (AI Honeypots)

An adaptive cyber-deception system leveraging Large Language Models (LLMs). This system creates dynamic honeypots, honeytokens, and honey APIs to mislead attackers, gather intelligence, and reduce real-world breach risks.

---

1. System Components and Roles

1.1 Deception Surface Planner (LLM)

Purpose: Design adaptive deception environments tailored to real-time threats.
Implementation Details (e.g.):

- **Inputs**:
  - Network topology, technology stack (e.g., web apps, databases, etc), threat intelligence feeds (e.g., MITRE ATT&CK, etc).
- **LLM Workflow**:
```
prompt = f"""
Network map: {network_map}.
Threat intel: Recent ransomware targeting Jenkins servers.
Design a fake high-value target for this environment.
"""
response = openai.ChatCompletion.create(
    model="gpt-4",
    messages=[{"role": "user", "content": prompt}]
)
# Output: "Deploy a fake Jenkins server at 192.168.1.100 with a vulnerable
plugin (CVE-2023-1234)."
```
- **Output**: Fake services (e.g., fake admin portals, vulnerable APIs) mapped to attacker trends.

1.2 Adaptive Honeytoken Engine

Purpose: Deploy and manage fake credentials, databases, and API keys.
Implementation Details (e.g.):

- **Tools**:

- **CanaryTokens**: Generate and inject fake credentials into codebases, Git repos, and databases.
- **Terraform**: Automate deployment of fake cloud resources (e.g., S3 buckets, VMs, etc).
- **Etc**.

- **Example**:

```
# Generate a fake AWS key
canarytoken generate aws-id --memo "Fake AWS key for honeytoken" --output fake_aws_key.txt
```

- **Alerting**: Integrate with SIEM (e.g., Splunk, Elastic SIEM, etc) to trigger alerts on honeytoken usage.

1.3 Attacker Interaction Analyzer

Purpose: Capture and classify attacker behavior in real time.

Implementation Details (e.g.):

- **Data Collection**:
  - **Honeypot Logging**: Use `Cowrie` (SSH honeypot) or `T-Pot` to log attacker commands.
  - **Network Traffic**: Capture API calls with `Zeek` or `Suricata`.
- **Classification**:
  - **Machine Learning**: Train a model to categorize actions (recon, exploitation):

```
from sklearn.svm import SVC
classifier = SVC()
classifier.fit(training_data, labels)  # Labels: ["recon", "credential_harvesting", ...]
```

1.4 LLM Response Simulator

Purpose: Generate dynamic, context-aware responses to mimic real systems.

Implementation Details (e.g.):

- **Real-Time Interaction**:

```
def simulate_sql_response(query):
    prompt = f"""
    Attacker SQL query: {query}
    Generate fake but plausible database records.
```

```
    """
    return llm.generate(prompt)  # Output: "1 | admin | encrypted_password | ..."
```

- **File System Simulation**:
  - For `ls /etc`, return fake config files (e.g., `fake_nginx.conf`).
- **APIs**: Use Flask/Django to host honey APIs with LLM-generated responses.

1.5 Deception Metrics Engine

Purpose: Measure deception effectiveness and optimize strategies.

Implementation Details (e.g.):

- **Metrics**:
  - **Engagement Time**: Track session duration via timestamps.
  - **Attacker Paths**: Graph traversal using tools like `Neo4j`.
- **Optimization**:

```
if avg_engagement_time < 10 minutes:
    planner.adjust_deception_strategy(increase_complexity=True)
```

---

2. System Integration and Component Interaction

1. **Initiation**: User defines network scope and threat intel feeds.
2. **Planning**: Deception Surface Planner designs fake Jenkins server, honey API, etc.
3. **Deployment**: Honeytoken Engine injects fake AWS keys into Git repos.
4. **Interaction**: Attacker triggers honeytoken → Analyzer logs actions → Simulator responds.
5. **Analysis**: Metrics Engine calculates engagement time (e.g., 45 minutes) → Recommends adding fake database.

---

3. Evaluation Criteria

1. **Attacker Engagement**: Time spent in deception environment (target: >30 minutes).
2. **Detection Rate**: Percentage of attackers triggering honeytokens.
3. **False Positives**: Legitimate users labeled as attackers by the honeypot.

4. **Operational Impact (optional)**: Reduction in real-system breaches.

---

4. Ethical and Operational Considerations

- **Authorization**: Deploy only in authorized network segments.
- **Data Privacy**: Ensure honeytokens contain no real data.
- **Transparency**: Document deception systems for internal audits.

---

5. Tools and Resources (e.g.)

- **Honeypots**: Cowrie, T-Pot, HoneyDB, etc.
- **LLMs**: GPT-4, Claude, Hugging Face models, etc.
- **Deployment**: Terraform, Ansible, Docker, etc.
- **Analytics**: Elasticsearch, Neo4j, Pandas, etc.
