Project Title: Large Language Model-Enabled Vulnerability Investigation and Triaging System

---

Short Project Description:

The system automates scanning, enriches findings with contextual insights, prioritizes risks, and generates remediation recommendations using Large Language Models (LLMs).

Component	Role
Vulnerability Scanner	Basic scanner to gather initial findings
Finding Context Enhancer	Uses LLM to understand the full context of each finding
Exploitability Assessor	LLM rates how exploitable each finding is
Risk Prioritizer	Ranks findings based on impact and ease of exploitation
Remediation Recommendation Generator	Suggests best mitigation actions
Final Report Composer	Writes a full prioritized vulnerability report

---

Component Details:

1. **Vulnerability Scanner**:
   - Tools like Nessus/OpenVAS/Nmap/etc initiate simple scans.
2. **Finding Context Enhancer**:
   - LLM reads scan results and expands:
     - What is vulnerable
     - Why it's risky
3. **Exploitability Assessor**:
   - Uses few-shot reasoning to predict likelihood of exploitability.
4. **Risk Prioritizer**:
   - Sorts findings into Critical/High/Medium/Low based on exploitability + asset value.
5. **Remediation Recommendation Generator**:
   - Drafts fix suggestions based on industry best practices and online documentation.
6. **Final Report Composer**:
   - Polishes report into human-readable executive + technical summaries.

Overall System Flow:

- Input: Initial scan results
- Output: Full prioritized and contextualized vulnerability report
- Focus: **LLM-driven triaging and prioritization**.

Internal Functioning of Each Module:

1. Vulnerability Scanner

- **Purpose**:
  - Gather raw vulnerabilities.
- **How it works**:
  - Runs traditional scanners:
    - OpenVAS
    - Nessus
    - Nuclei
    - Etc
- **Output**:
  - Unstructured or loosely structured findings.

---

2. Finding Context Enhancer

- **Purpose**:
  - Add **rich context** to each finding.
- **How it works**:
  - LLM reads scanner output.
  - Generates:
    - Threat models
    - Possible real-world attack scenarios
    - Business impacts
- **Example**:
  - From `"Open Redis without password"`, LLM outputs:
    - `"Could allow remote code execution if exploited with module upload."`

---

3. Exploitability Assessor

- **Purpose**:
  - Predict **likelihood of real exploitation**.
- **How it works**:
  - Considers:
    - Service exposure (internal/external)
    - Availability of public exploits (Exploit-DB, GitHub)
    - Mitigations in place
    - Etc
- **Score output**:
  - High/Medium/Low exploitability.

---

4. Risk Prioritizer

- **Purpose**:
  - Prioritize vulnerabilities for action.
- **How it works**:
  - Combine:
    - Exploitability
    - Asset criticality
    - CVSS base scores
    - Etc
- **Output**:
  - Top vulnerabilities first.

---

5. Remediation Recommendation Generator

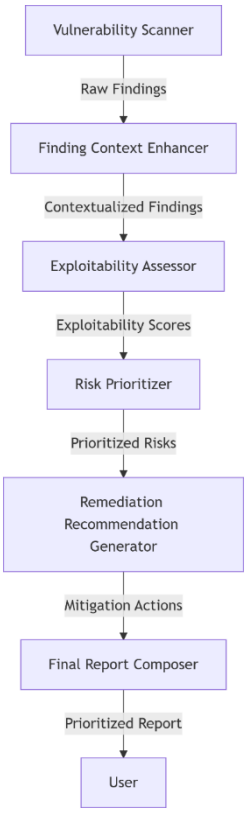
- **Purpose**:
  - Suggest practical fixes.
- **How it works**:
  - Generates:
    - Config changes
    - Patch upgrade suggestions
    - Infrastructure redesign if needed

---

6. Final Report Composer

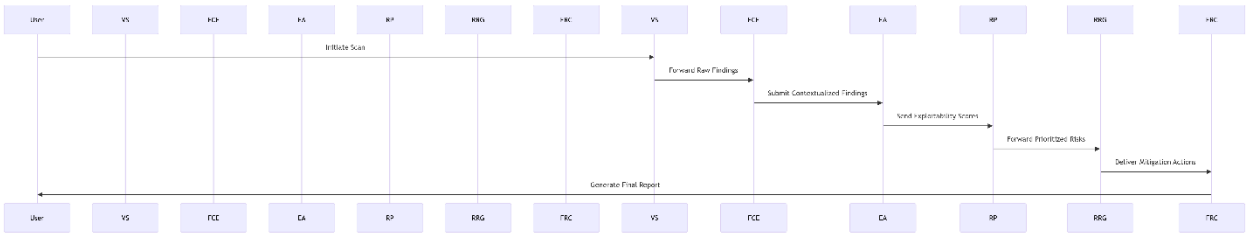
- **Purpose**:
  - Create final polished report.
- **How it works**:
  - Executive Summary + Technical Findings
  - PDF/HTML/etc output formats.

Component Diagram



- **Vulnerability Scanner**: Runs tools like Nessus/Nmap/etc to gather raw scan results.
- **Finding Context Enhancer**: Uses LLM to enrich findings with threat models and attack scenarios.
- **Exploitability Assessor**: Evaluates likelihood of exploitation (High/Medium/Low).
- **Risk Prioritizer**: Ranks vulnerabilities by combining exploitability, asset value, and CVSS scores.
- **Remediation Recommendation Generator**: Generates fixes (e.g., patches, config changes, etc).
- **Final Report Composer**: Produces polished reports with executive summaries and technical details.

Sequence Diagram



1. **User** triggers a scan via the **Vulnerability Scanner**.
2. **Scanner** sends raw findings to the **Finding Context Enhancer** for LLM-driven enrichment.
3. **Context Enhancer** passes contextualized findings to the **Exploitability Assessor** for scoring.
4. **Assessor** sends scores to the **Risk Prioritizer** for ranking.
5. **Prioritizer** forwards ranked risks to the **Remediation Recommendation Generator**.
6. **Recommendation Generator** submits mitigation actions to the **Final Report Composer**.
7. **Report Composer** delivers a prioritized, actionable report to the **User**.

Detailed Project Description: LLM-Enabled Vulnerability Investigation and Triaging System

An AI-driven vulnerability triage system. This system automates scanning, enriches findings with contextual insights, prioritizes risks, and generates remediation recommendations using Large Language Models (LLMs).

---

1. System Components and Roles

1.1 Vulnerability Scanner

Purpose: Perform initial vulnerability scans using industry-standard tools.
Implementation Details (e.g.):

- **Tools**:
  - **Nessus/OpenVAS**: For comprehensive vulnerability scanning.
  - **Nuclei**: For targeted checks (e.g., misconfigurations, CVEs).
  - **Etc**
- **Example Command**:
  ```
  nessuscli scan --target 192.168.1.0/24 --policy "Full Audit" --output scan_results.xml
  ```
- **Output**: XML/JSON reports (e.g., Nessus `.nessus` files).

1.2 Finding Context Enhancer

Purpose: Use LLMs to add threat context and business impact analysis.
Implementation Details (e.g.):

- **LLM Integration**:
  ```
  from openai import OpenAI
  client = OpenAI(api_key="API_KEY")
  def enrich_finding(finding):
      prompt = f"""
      Vulnerability: {finding['description']}
      Provide a 3-sentence analysis of potential real-world attack scenarios and business risks.
      """
      response = client.chat.completions.create(
  ```

```
        model="gpt-4",
        messages=[{"role": "user", "content": prompt}]
    )
    return response.choices[0].message.content
```

- **Example Output**:

  "An open Redis instance without authentication (CVE-2022-0543) could allow attackers to execute arbitrary code via Lua sandbox escape. This is critical for internet-facing systems storing sensitive session data. Attackers could exfiltrate or corrupt data, leading to compliance violations."

1.3 Exploitability Assessor

Purpose: Predict likelihood of exploitation using LLM few-shot learning.
Implementation Details (e.g.):

- **Prompt Template**:
```
prompt = f"""
Service: {service}
Exposure: {exposure}
Public Exploits: {exploit_links}
Mitigations: {mitigations}
Score exploitability as High/Medium/Low.
"""
```

- **Scoring Logic**:
  - **High**: Internet-facing service with public exploits (e.g., Log4j).
  - **Medium**: Internal service with theoretical exploits.
  - **Low**: Patched or non-exploitable findings.

1.4 Risk Prioritizer

Purpose: Rank vulnerabilities using CVSS, exploitability, and asset value.
Implementation Details (e.g.):

- **Algorithm**:
```
Risk Score = (CVSS Base Score × 0.6) + (Exploitability Score × 0.3) + (Asset Criticality × 0.1)
```
- **Asset Criticality**: Predefined tiers (e.g., 1.0 for databases, 0.5 for test servers).

1.5 Remediation Recommendation Generator

Purpose: Suggest actionable fixes using LLM reasoning.
Implementation Details (e.g.):

- **Example Prompt**:
  `"Generate step-by-step remediation for CVE-2021-44228 (Log4j):"`
- **Sample Output**:

  "1. Upgrade Log4j to version 2.17.1. 2. Set `log4j2.formatMsgNoLookups=true`. 3. Monitor for JNDI lookup patterns in logs."

1.6 Final Report Composer

Purpose: Generate executive and technical reports.
Implementation Details (e.g.):

- **Tools**:
  - **Jinja2**: For HTML/PDF templating.
  - **Pandas**: To structure findings into tables.
  - **Etc**
- **Template Example**:

```
<h2>Executive Summary</h2>
<p>{{ executive_summary }}</p>
<h2>Technical Findings</h2>
<table>
  {% for finding in findings %}
  <tr><td>{{ finding.title }}</td><td>{{ finding.risk }}</td></tr>
  {% endfor %}
</table>
```

---

2. System Integration and Component Interaction

1. **Scanning**:
   - **Vulnerability Scanner** runs Nessus/OpenVAS/Nuclei/etc → outputs `scan_results.xml`.
2. **Context Enrichment**:

    - **Finding Context Enhancer** processes XML → enriches with LLM-generated context.
3. **Exploitability Scoring**:
    - **Exploitability Assessor** assigns High/Medium/Low scores → forwards to **Risk Prioritizer**.
4. **Prioritization**:
    - **Risk Prioritizer** calculates risk scores → sorts findings.
5. **Remediation**:
    - **Remediation Generator** drafts fixes → sends to **Report Composer**.
6. **Reporting**:
    - **Report Composer** generates PDF/HTML report.

---

3. Evaluation Criteria

1. **Context Accuracy**: Manual review of LLM-generated context against expert analysis.
2. **Exploitability Precision**: Compare LLM scores against real-world exploit data (e.g., Exploit-DB, etc).
3. **Remediation Actionability**: Validate fixes with IT teams (e.g., patch success rate, etc).
4. **Report Clarity**: User surveys on readability and usefulness.

---

4. Ethical Considerations

- **Authorization**: Only scan authorized networks.
- **Data Privacy**: Anonymize sensitive data in reports.
- **Validation**: Cross-check LLM recommendations with security experts.

---

5. Tools and Resources (e.g.)

- **Scanning**: Nessus, Nuclei, Nmap, OpenVAS, etc.
- **LLMs**: GPT-4, Claude, Mistral.
- **Reporting**: Jinja2, Pandas, LaTeX.
