Project Title: Contextual Vulnerability Risk Scoring via LLM Reasoning

---

Core Idea:

- Use LLMs to **analyze each vulnerability** in context (asset type, exposure, business impact) and dynamically assign a **risk score** beyond simple CVSS metrics.

Component	Role
Asset Context Collector	Gathers asset metadata (type, criticality, exposure level, etc)
LLM-Based Risk Reasoning Engine	Reason about impact, exposure, and criticality for each vulnerability
Business Impact Modeler	Calculates operational, financial, reputational risk
Prioritization Score Calculator	Generates dynamic, context-aware risk scores
Reporting Engine	Creates prioritized vulnerability remediation lists

---

Component Details:

1. **Asset Context Collector**:
    - Pulls:
        - CMDB (Configuration Management Database) data
        - Cloud inventory APIs
        - Asset criticality tags
        - Etc
2. **LLM-Based Risk Reasoning Engine**:
    - Input:
        - Vulnerability description
        - Asset context
    - Output:
        - Dynamic qualitative and quantitative risk reasoning.
3. **Business Impact Modeler**:
    - Simulates:
        - Downtime costs
        - Data breach costs
        - Regulatory fines (e.g., GDPR, HIPAA, NIS 2)
4. **Prioritization Score Calculator**:
    - Combines:
        - LLM qualitative output
        - Business impact models
    - Calculates **dynamic risk scores** far beyond static CVSS.
5. **Reporting Engine**:

- Creates dashboards and reports:
  - Sorted by business risk, not just technical severity.

Overall System Flow:

- Input: Vulnerability + Asset context
- Output: Ranked vulnerability list by real-world impact
- Focus: **Business-aware risk prioritization**

Internal Functioning of Each Module:

1. Asset Context Collector

- **Data sources**:
  - CMDB (asset metadata)
  - Cloud inventory APIs (AWS, Azure Resource Graph, etc)
  - Service type detection (web servers, DBs, control systems, etc)
  - Etc
- **Metadata collected**:
  - Asset type (server, workstation, IoT, etc)
  - Criticality (high, medium, low business value, etc)
  - Exposure level (public internet, internal network, etc)
  - Etc

---

2. LLM-Based Risk Reasoning Engine

- **Process**:
  - Input:
    - Vulnerability details (CVE, attack vector, complexity etc)
    - Asset context
  - Reasoning using chain-of-thought prompting :
    - `« This CVE allows remote unauthenticated access. The target is a public-facing web server hosting customer data ➔ High risk. »`
  - Output:
    - Narrative risk explanation.

---

3. Business Impact Modeler

- **Model elements**:
  - Direct costs (downtime × cost/hour)
  - Regulatory fines (HIPAA, NIS 2, GDPR calculations)
  - Brand damage (proxy via historical breach studies)

---

4. Prioritization Score Calculator

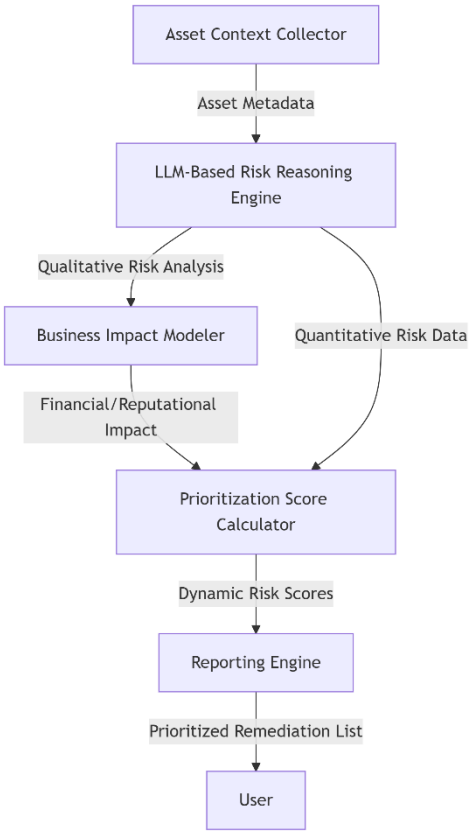
- **Risk aggregation**:
  - Weighted scoring:
    - (LLM threat level × Asset exposure × Business impact)
- **Output**:
  - Prioritized list with transparent scoring breakdown.

---

6. Reporting Engine

- **Deliverables**:
  - Graphical risk rankings.
  - Top vulnerabilities.
  - Suggested remediation orders.
  - Etc.

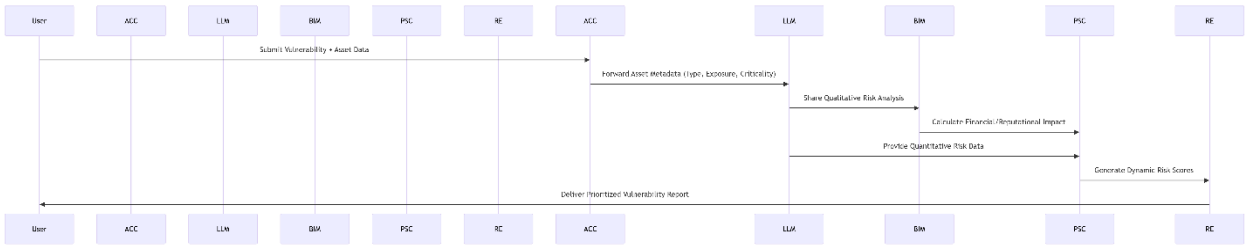
Component Diagram



1. **Asset Context Collector**: Gathers metadata (e.g., asset type, exposure level, etc) from CMDB, cloud APIs, and criticality tags.

2. **LLM-Based Risk Reasoning Engine**: Analyzes vulnerabilities in context (e.g., “Public web server with customer data ➔ high risk”).
3. **Business Impact Modeler**: Estimates operational/financial impact (e.g., downtime costs, GDPR/NIS 2 fines, etc).
4. **Prioritization Score Calculator**: Combines LLM insights and business impact to compute dynamic scores.
5. **Reporting Engine**: Ranks vulnerabilities by business-critical risk, not just CVSS.

Sequence Diagram



1. **User** submits a vulnerability and asset data (e.g., CVE-2023-XXXX + public-facing server details).
2. **Asset Context Collector** forwards metadata (e.g., "public internet exposure, high criticality") to the **LLM Engine**.
3. **LLM Engine** performs risk reasoning (e.g., "Remote code execution on a customer portal → severe reputational risk").
4. **Business Impact Modeler** calculates financial/reputational costs and shares them with the **Score Calculator**.
5. **Score Calculator** combines LLM outputs and business impact to generate a dynamic risk score (e.g., 9.8/10).
6. **Reporting Engine** delivers a prioritized list (e.g., "Patch this server first due to high business impact").

Detailed Project Description: Contextual Vulnerability Risk Scoring via LLM Reasoning

A framework that dynamically prioritizes vulnerabilities by combining Large Language Model (LLM) reasoning, asset context, and business impact analysis. This framework moves beyond static CVSS scores to deliver actionable, business-aware risk assessments.

---

1. System Architecture Overview

Core Components & Interactions

1. **Asset Context Collector**
    - *Inputs*: CMDB data, cloud APIs (AWS/Azure), criticality tags, etc.
    - *Outputs*: Asset metadata (type, exposure, business criticality, etc).
2. **LLM-Based Risk Reasoning Engine**
    - *Inputs*: Vulnerability details (CVE, exploitability, etc), asset context.
    - *Outputs*: Narrative risk explanations (e.g., "RCE on public-facing server").
3. **Business Impact Modeler**
    - *Inputs*: Asset criticality, industry-specific risk factors.
    - *Outputs*: Financial/reputational impact estimates (downtime costs, GDPR / NIS 2, etc, fines).
4. **Prioritization Score Calculator**
    - *Inputs*: LLM reasoning, business impact data.
    - *Outputs*: Dynamic risk scores (e.g., 9.8/10).
5. **Reporting Engine**
    - *Inputs*: Risk scores.
    - *Outputs*: Prioritized remediation lists, dashboards, etc.

Integration Flow:

1. Asset data → Context Collector → LLM Engine → Business Impact Modeler → Score Calculator → Reporting Engine.
2. Feedback loop: Adjust risk weights based on historical incident outcomes.

---

2. Component Implementation Details

2.1 Asset Context Collector

Objective: Aggregate asset metadata for risk contextualization.

Tools & Workflow (e.g.):

- **Data Sources**:
  - *CMDB*: ServiceNow or Jira Service Management, etc.
  - *Cloud APIs*: AWS Resource Groups, Azure Resource Graph, etc.
  - *Criticality Tags*: Custom tags (e.g., "PCI-DSS Tier 1").
- **Metadata Extraction**:
  - **Asset Type**: Classify using regex or ML (e.g., "web server" vs. "database").
  - **Exposure Level**: Check network segmentation (public IP, VLAN, etc).
  - **Criticality**: Pull from CMDB or tag systems (e.g., "high" for ERP systems).
- **Example Code [python]**:

```python
def get_asset_context(asset_id):
    cmdb_data = servicenow_api.get_asset(asset_id)
    cloud_data = aws_api.describe_instance(asset_id)
    return {
        "type": cmdb_data["type"],
        "exposure": "public" if cloud_data["public_ip"] else "internal",
        "criticality": cmdb_data["tags"].get("criticality", "medium")
    }
```

2.2 LLM-Based Risk Reasoning Engine

Objective: Generate contextual risk narratives using LLMs.

Implementation Steps (e.g.):

1. **LLM Setup**:
   - *Base Model*: GPT-4 or fine-tuned Llama 2 on cybersecurity datasets (CVE descriptions, MITRE ATT&CK, etc).

- *Prompt Engineering*:

```
prompt = f"""
Vulnerability: {cve_description}
Asset: {asset_type}, {exposure}, {criticality}.
Explain the risk in business terms.
"""
response = openai.ChatCompletion.create(model="gpt-4", messages=[{"r
ole": "user", "content": prompt}])
# Output: "Remote code execution on a public-facing payment gateway →
High risk of data breach and PCI-DSS non-compliance."
```

2. **Validation**:
   - Compare LLM outputs with expert assessments (precision/recall, etc).

2.3 Business Impact Modeler

Objective: Quantify operational, financial, and reputational risks.

Implementation Strategies (e.g.):

- **Cost Models**:
  - *Downtime Costs*: `(hourly revenue) × (MTTR estimate)`.
  - *Regulatory Fines*: Use GDPR calculator (4% of global turnover) or HIPAA penalties, etc.
  - *Reputational Damage*: Estimate via historical breach data (e.g., 20% customer churn).
- **Example Code**:

```
def calculate_fines(asset_type, region):
    if "GDPR" in asset_type["compliance_tags"]:
        return revenue * 0.04
    elif "HIPAA" in asset_type["compliance_tags"]:
        return 50000  # Base penalty
```

2.4 Prioritization Score Calculator

Objective: Compute dynamic risk scores.

Implementation Steps (e.g.):

1. **Weighted Formula**:
   ```
   Risk Score = (LLM Threat Level × 0.5) + (Business Impact × 0.3) + (Exposur
   e × 0.2)
   ```
   - *LLM Threat Level*: Normalized score from 1–10 based on risk narrative.
   - *Business Impact*: Financial/reputational impact (Euro).
   - *Exposure*: Public (1.0) vs internal (0.5).
2. **Example Code**:
   ```
   def calculate_score(threat_level, business_impact, exposure):
       return (threat_level * 0.5) + (business_impact * 0.3) + (exposure * 0.
   2)
   ```

2.5 Reporting Engine

Objective: Generate actionable reports for stakeholders.

Implementation Details (e.g.):

- **Dashboard**:
  - *Tools*: Grafana or Tableau for visualizations, etc.
  - *Metrics*: Top vulnerabilities by risk score, cost projections, etc.
- **Remediation Playbooks**:
  - Auto-generate Jira tickets with priority labels (e.g., "Critical: Patch CVE-2023-XXXX within 48h").
- **Example Output**:
  ```
  ## Priority 1: CVE-2023-XXXX (Score: 9.8/10)
  - **Asset**: Public-facing payment gateway (High criticality).
  - **Business Impact**: $2M potential downtime + $1.2M GDPR fines.
  - **Action**: Apply patch KB123456 by [date].
  ```

---

3. Evaluation Metrics

1. **Accuracy**:
    - Compare LLM risk narratives vs expert assessments (F1-score).
2. **Business Alignment**:
    - Reduction in downtime costs post-remediation (e.g., 30% faster patching).
3. **Usability**:
    - Stakeholder feedback on report clarity (Likert scale).

---

4. Challenges & Mitigation (optional)

- **Data Silos**:
    - Use middleware (Apache NiFi) to unify CMDB/cloud data.
- **LLM Hallucinations**:
    - Rule-based validation (e.g., flag non-existent CVEs).
- **Dynamic Environments**:
    - Schedule periodic asset context refreshes (e.g., nightly scans).

---

5. Tools & Technologies (e.g.)

- **LLM Framework**: Hugging Face Transformers, OpenAI API, Claude ai, DeepSeek, etc.
- **Data Integration**: Apache NiFi, AWS SDK, etc.
- **Visualization**: Grafana, Tableau, etc.
- **Database**: PostgreSQL, MongoDB, etc.
