Project Title: Real-Time Vulnerability Prioritization through Continuous Attack Surface Monitoring

---

Core Idea:

- Instead of prioritizing once after scanning, **prioritize vulnerabilities continuously** as the **attack surface changes** (new assets exposed, old ones retired).

Component	Role
Continuous Asset Discovery Engine	Constantly discovers public and internal assets
Vulnerability Mapping System	Maps vulnerabilities to live assets
Real-Time Risk Adjustment LLM	Dynamically recalculates vulnerability criticality
Adaptive Remediation Recommender	Shifts patch priorities based on real-world asset exposure
Visualization Dashboard	Live maps of exposure and associated risks

---

Component Details:

1. **Continuous Asset Discovery Engine**:
    - Uses:
        - Passive DNS.
        - SSL certificates.
        - Scanning.
        - Cloud APIs (AWS, Azure inventory)
        - Etc.
2. **Vulnerability Mapping System**:
    - Connects discovered assets to known vulnerabilities dynamically.
3. **Real-Time Risk Adjustment LLM**:
    - Re-evaluates priorities:
        - For example, if an asset becomes **publicly exposed**, the associated vulnerabilities become much more critical immediately.
4. **Adaptive Remediation Recommender**:
    - Suggests:
        - Immediate patching.
        - Temporary compensating controls (WAF rules, access restrictions, etc).
5. **Visualization Dashboard**:
    - Shows changing attack surfaces and vulnerability posture **live**.

Overall System Flow:

- Input: Live asset exposure data
- Output: Real-time vulnerability priority lists
- Focus: **Environment-aware continuous reprioritization**

Internal Functioning of Each Module:

1. Continuous Asset Discovery Engine

- **Discovery methods**:
  - Passive DNS monitoring (e.g., SecurityTrails, Farsight DNSDB).
  - SSL/TLS certificate discovery (e.g., Censys).
  - Cloud asset crawling (AWS Config, Azure Resource Graph).
  - Scanning.
  - Etc.

---

2. Vulnerability Mapping System

- **Mapping**:
  - For each discovered asset:
    - Map any known vulnerabilities (open source vuln databases, NIST NVD, CVE, EDB, dark web, etc).

---

3. Real-Time Risk Adjustment LLM

- **Behavior**:
  - When asset exposure changes:
    - If a database is suddenly open to the internet, associated vulnerabilities become **Priority 1**.
- **Re-prioritization triggers**:
  - New public IP assignments.
  - New open ports.
  - Software version upgrades/downgrades.
  - Etc.

---

4. Adaptive Remediation Recommender

- **Tactics suggested**:
  - Emergency patch.
  - WAF shielding.
  - Temporary asset isolation.

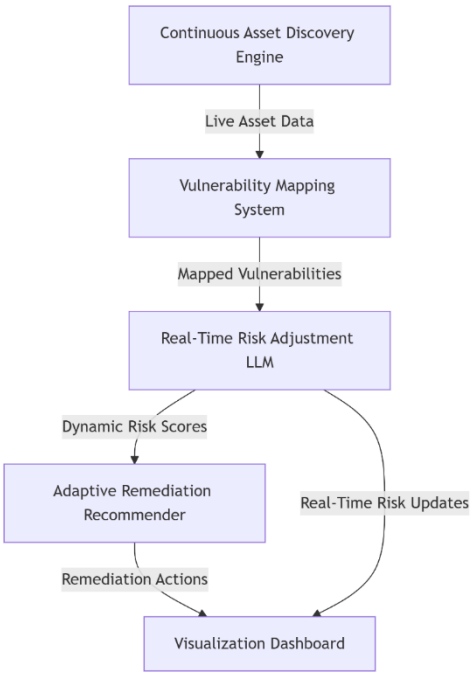
- Etc.

---

5. Visualization Dashboard

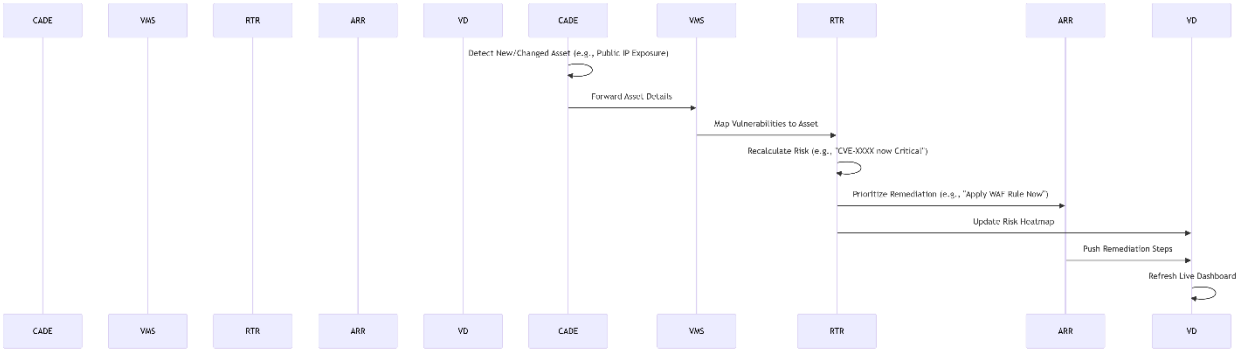
- **Live Views**:
  - Exposure heatmaps.
  - "Newly critical vulnerabilities" alerts.
  - Trend graphs of risk over time.
  - Etc.

Component Diagram



1. **Continuous Asset Discovery Engine**: Discovers assets in real-time via DNS, SSL certificates, cloud APIs, etc.
2. **Vulnerability Mapping System**: Links live assets to vulnerabilities (e.g., CVEs from NVD).
3. **Real-Time Risk Adjustment LLM**: Dynamically adjusts risk scores based on exposure changes (e.g., public-facing servers).
4. **Adaptive Remediation Recommender**: Recommends immediate actions (e.g., patching, WAF rules, etc).
5. **Visualization Dashboard**: Displays live attack surface maps and prioritized risks.

Sequence Diagram



1. **Asset Discovery Engine** detects a new public IP or open port.
2. **Vulnerability Mapping System** identifies associated vulnerabilities (e.g., CVE-2023-XXXX).
3. **Risk Adjustment LLM** recalculates risk (e.g., "CVE-2023-XXXX: Priority 1 due to public exposure").
4. **Remediation Recommender** suggests urgent actions (e.g., "Block port 443 temporarily").
5. **Visualization Dashboard** updates to reflect new risks and remediation steps in real-time.

Detailed Project Description: Real-Time Vulnerability Prioritization through Continuous Attack Surface Monitoring

A framework that dynamically prioritizes vulnerabilities based on real-time changes to an organization's attack surface. This framework integrates continuous asset discovery, vulnerability mapping, and AI-driven risk adjustment to enable proactive defense.

---

1. System Architecture Overview

Core Components & Interactions

1. **Continuous Asset Discovery Engine**
    - *Inputs*: DNS records, SSL certificates, cloud APIs, etc.
    - *Outputs*: Real-time inventory of public/internal assets.
2. **Vulnerability Mapping System**
    - *Inputs*: Asset data, vulnerability databases (NVD, CVE, etc).
    - *Outputs*: Asset-vulnerability pairs (e.g., "Server X → CVE-2023-XXXX").
3. **Real-Time Risk Adjustment LLM**
    - *Inputs*: Asset exposure changes, vulnerability mappings.
    - *Outputs*: Updated risk scores (e.g., "CVE-2023-XXXX: Priority 1").
4. **Adaptive Remediation Recommender**
    - *Inputs*: Risk scores.
    - *Outputs*: Actionable recommendations (patching, WAF rules, etc).
5. **Visualization Dashboard**
    - *Inputs*: Risk data, remediation steps, etc.
    - *Outputs*: Live attack surface maps, prioritized alerts, etc.

Integration Flow:

1. Asset Discovery → Vulnerability Mapping → Risk Adjustment → Remediation → Visualization.
2. Feedback loop: Remediation actions trigger re-evaluation of the attack surface.

---

2. Component Implementation Details

2.1 Continuous Asset Discovery Engine

Objective: Continuously identify new and retired assets across environments.

Tools & Workflow (e.g.):

- **Data Sources**:
  - *Passive DNS*: SecurityTrails API, Farsight DNSDB.
  - *SSL/TLS Certificates*: Censys API or SSLMate.
  - *Cloud Assets*: AWS Resource Groups, Azure Resource Graph.
  - *Scanning*.
  - *Etc.*
- **Example Code (AWS Asset Discovery)**:

```
import boto3
def get_aws_instances():
    ec2 = boto3.client('ec2')
    instances = ec2.describe_instances()
    return [instance['PublicIpAddress'] for instance in instances]
```

- **Validation**:
  - Deduplicate assets using unique identifiers (IP, hostname).
  - Flag ephemeral assets (e.g., cloud instances) for dynamic tracking.

2.2 Vulnerability Mapping System

Objective: Link live assets to known vulnerabilities dynamically.

Implementation Steps (e.g.):

1. **Asset Profiling**:
   - Use `nmap` or `Shodan` to detect software versions, open ports.
2. **Vulnerability Lookup**:
   - Query NVD API for CVEs matching detected software/versions.

- *Example Code*:

```
import requests
def get_cves_for_software(software, version):
    response = requests.get(f"https://services.nvd.nist.gov/rest/jso
n/cves/2.0?keyword={software} {version}")
    return response.json()["vulnerabilities"]
```

3. **Automation**:
    - For example, schedule hourly scans to update mappings.

2.3 Real-Time Risk Adjustment LLM

Objective: Re-prioritize vulnerabilities when asset exposure changes.

Implementation Steps (e.g.):

1. **LLM Setup**:
    - *Base Model*: GPT-4 or fine-tuned open-source model (Llama 2) on cybersecurity datasets.
    - *Prompt Engineering*:

```
prompt = f"""
Asset: Public-facing web server (IP: 203.0.113.5)
Vulnerability: CVE-2023-XXXX (Remote Code Execution)
Exposure Change: Port 443 opened to the internet.
Recalculate risk priority.
"""
response = openai.ChatCompletion.create(model="gpt-4", messages=[{"r
ole": "user", "content": prompt}])
# Output: "Priority 1: Critical risk due to public exposure."
```

2. **Risk Scoring**:
    - Assign weights: Exposure (50%), CVSS (30%), asset criticality (20%).

2.4 Adaptive Remediation Recommender

Objective: Suggest immediate actions to mitigate high-risk vulnerabilities.

Implementation Strategies (e.g.):

1. **Recommendation Rules (example)**:
    - *Public Exposure*:
        - Block port via firewall.
        - Apply WAF rules (e.g., AWS Shield, Cloudflare).
        - Etc.
    - *Critical Vulnerability*:
        - Patch within 24h.
        - Isolate asset temporarily.
        - Etc
2. **Integration with Tools (optional)**:
    - Automate firewall rules via Terraform/Ansible.
    - *Example Code (AWS WAF)*:

```
def block_ip(ip):
    waf = boto3.client('wafv2')
    waf.update_ip_set(IPSetId='...', Updates=[{'Action': 'INSERT', '
IPSetDescriptor': {'Type': 'IPV4', 'Value': ip}}])
```

2.5 Visualization Dashboard

Objective: Provide real-time visibility into attack surface risks.

Implementation Details (e.g.):

- **Tools**:
    - *Dashboard*: Grafana or Elastic Kibana.
    - *Mapping*: Kepler.gl for geospatial asset visualization.
    - *Etc*.
- **Key Visualizations**:
    - Heatmaps of exposed assets.

    - Trend lines showing risk fluctuations.
    - Top 10 critical vulnerabilities.
    - Etc.
- **Example Alert**:

```
{
  "timestamp": "2023-10-01T14:30:00Z",
  "asset": "203.0.113.5",
  "cve": "CVE-2023-XXXX",
  "action": "Block port 443 immediately."
}
```

---

3. Evaluation Metrics

1. **Detection Speed**:
    - Time from asset exposure to risk adjustment (e.g., <5 minutes).
2. **Remediation Efficacy**:
    - Percentage reduction in high-risk vulnerabilities post-action.
3. **Accuracy**:
    - False positive/negative rates in asset-vulnerability mapping.

---

4. Challenges & Mitigation (optional)

- **Data Overload**:
    - Use edge computing to preprocess asset data locally.
- **LLM Latency**:
    - Cache common risk scenarios for faster responses.
- **Ephemeral Assets**:
    - Tag short-lived cloud resources for dynamic tracking.

---

6. Tools & Technologies (e.g.)

- **Asset Discovery**: SecurityTrails, Censys, AWS SDK, etc.
- **Vulnerability Mapping**: Nmap, Shodan, NVD API, etc.
- **LLM**: OpenAI GPT-4, Hugging Face Transformers, etc.
- **Dashboard**: Grafana, Elastic Kibana, etc.
- **Automation**: Ansible, Terraform, etc.
