nvd_api.py

```
import requests
from config import NVD_API_URL, NVD_API_KEY

#Simple cache dictionary to avoid repeated API calls
_cve_cache = {}

def get_cve_detail(cve_id):
    """
    Fetch detailed CVE information from NVD API by CVE ID.
    Uses query parameter cveId to properly fetch the vulnerability.
    Caches results in memory.
    """
    global _cve_cache
    if cve_id in _cve_cache:
    # Already fetched this one? Just use cache.
        return _cve_cache[cve_id]

    headers = {"apiKey": NVD_API_KEY}
    params = {"cveId": cve_id}

    try:
    #Actually hit NVD API for full details
        response = requests.get(NVD_API_URL, headers=headers, params=params,
timeout=30)
        response.raise_for_status()
        data = response.json()
    except requests.RequestException as e:
        print(f"Error fetching CVE {cve_id} from NVD API: {e}")
        #Give a "not found" response in cache so we don't repeat bad requests
        _cve_cache[cve_id] = {"ID": cve_id, "Description": "Not found",
"Severity": "UNKNOWN"}
        return _cve_cache[cve_id]

    vulnerabilities = data.get("vulnerabilities", [])
    if not vulnerabilities:
    #NVD sent back an empty result for this ID
        _cve_cache[cve_id] = {"ID": cve_id, "Description": "Not found",
"Severity": "UNKNOWN"}
        return _cve_cache[cve_id]

    cve_data = vulnerabilities[0].get("cve", {})
```

```
    cve_id_ret = cve_data.get("id", cve_id)
    #Pull a human-readable description.
    description = next(
        (desc.get("value") for desc in cve_data.get("descriptions", []) if
desc.get("lang") == "en"),
        "No description available."
    )

    #Figure out which CVSS metric version is available (try v3.1, v3.0, then
v2)
    severity = "UNKNOWN"
    cvss_score = None
    metrics = cve_data.get("metrics", {})

    if "cvssMetricV31" in metrics:
        sev_metric = metrics["cvssMetricV31"][0]
        severity = sev_metric["cvssData"].get("baseSeverity", severity)
        cvss_score = sev_metric["cvssData"].get("baseScore", cvss_score)
    elif "cvssMetricV30" in metrics:
        sev_metric = metrics["cvssMetricV30"][0]
        severity = sev_metric["cvssData"].get("baseSeverity", severity)
        cvss_score = sev_metric["cvssData"].get("baseScore", cvss_score)
    elif "cvssMetricV2" in metrics:
        sev_metric = metrics["cvssMetricV2"][0]
        severity = sev_metric.get("baseSeverity", severity)
        cvss_score = sev_metric["cvssData"].get("baseScore", cvss_score)

    #Try to extract the CWE ID if listed
    cwe = None
    for weakness in cve_data.get("weaknesses", []):
        for desc in weakness.get("description", []):
            if desc.get("lang") == "en":
                cwe = desc.get("value")
                break
        if cwe:
            break

    published = cve_data.get("published")
    last_modified = cve_data.get("lastModified")

    references = [ref.get("url") for ref in cve_data.get("references", []) if
ref.get("url")]

    products = []
    #Note: could also extract affected products/configurations, but not
```

```
implemented

    cve_detail = {
        "ID": cve_id_ret,
        "Description": description,
        "Severity": severity,
        "CVSS_Score": cvss_score,
        "CWE": cwe,
        "Published": published,
        "lastModified": last_modified,
        "References": references,
        "Products": products,
        "metrics": metrics,
    }

    _cve_cache[cve_id] = cve_detail
    return cve_detail
```