Uncommon Insights for Common Vulnerabilities

Team Members:

Chris Armstrong, Spencer Hutton, Karina Kanjaria

Important Project Links:

- GitHub Repository: https://github.com/karinakanjaria/cyber-etl
- Data Location: https://github.com/karinakanjaria/cyber-etl/tree/main/data
- Jupyter Notebook: https://github.com/karinakanjaria/cyber-etl/blob/main/data_integration.ipynb
- Graph Diagram: https://github.com/karinakanjaria/cyber-etl/blob/main/CVE%20Meta%20Diagram.pdf

Overview:

In this project, we explored relationships between vulnerabilities and a range of organizations, persons, products, and geopolitical entities. Relationships were based on being mentioned in CISA alerts or through common URLs linking two resources together.

Original Data Sets Used:

- Common Vulnerability Database (CVD)
    - https://nvd.nist.gov/vuln/data-feeds#JSON_FEED
    - Provided by National Institute of Standards and Technology (NIST)
    - Lists Common Vulnerabilities and Exposures (CVEs)
    - NVD feeds with US government repository of standards based vulnerability management
    - Details of issues help organizations know what needs to be fixed, but are often also leveraged by hackers to exploit the vulnerabilities
    - 20 years worth of structured raw JSON (2002 - 2022)
- Cybersecurity and Infrastructure Security Agency (CISA)
    - https://www.cisa.gov/uscert/ncas/alerts
    - Alerts provide information about security issues, vulnerabilities, and exploits. Alerts contain a summary, details on the nature of the threat, impact, and mitigation steps to prevent, detect, and respond to the threat.
    - Nearly 300 alerts from 2008 - 2022
    - Unstructured text similar to news articles
        - References to CVEs
        - Related entities
        - Affected systems

        - Techniques, tactics, and procedures
- GitHub
    - https://github.com/
    - NIST CVE data links to github pages
    - Structured data from GitHub APIs
    - Collaborators and programming languages from repositories

Processing and Ingesting the Data:

CVEs

Common Vulnerability and Exposures data included published vulnerability information for the entire CVD from NIST. We used requests to pull the data programmatically then JSON Path queries to find the following data across all CVEs efficiently.

- "cve_id": ID of the Vulnerability
- "score": Base Score CVSSv3
- "exploitability": Scored exploitability based on CVSSv3
- "impact": Scored impact based on CVSSv3
- "attack_vector": Method of Attack (Network, local, etc)
- "published": Date the CVE was initially published
- "refs": URLs related to the vulnerability
- "description": Description of the vulnerability
- "cpes": Affected software and system configurations

Alerts

In this project, we extracted data from about 300 alerts from 2008-2022. We used the CISA website's archive of alerts to get a list of alert URLs. Then the text of each alert is extracted. The alert is in the style of an article - text meant to be read by humans. From the text, we extracted: Alert name, Alert date, CVEs using regular expressions, Entities using Spacy, and Techniques, Tactics, Procedures

Entities are reduced using Dedupe. E.g. "The Federal Bureau of Investigation" should be merged into "Federal Bureau of Investigation". Product typed entities are related to CVE configs - e.g. "microsoft windows" and "microsoft windows 10" using string similarity. Edge weights are assigned according to how frequently the entity appears in each alert.

Alerts enrich the graph by telling us which vulnerabilities are actively used in attacks. In addition to that, entity extraction provided additional information about who used vulnerabilities and how they are being exploited.

GitHub

We wanted to learn more about vulnerabilities and specifically see the major contributions that were made in this domain. In order to do that, we chose to use github repositories that were related to vulnerabilities.

First, we started with pulling all url links that were referenced in the NIST CVD vulnerability dataset. From this list, we then filtered for all urls that included “github” in the domain. This gave us a list of github links and their CVE counterparts that they were associated with. We found that some links were associated with multiple vulnerabilities and we retained this information to make sure they were all connected properly in the graph.

After obtaining the list of github urls, we used url split to obtain the github repository locations. This allowed us to leverage the github APIs instead of accessing the individual pages. We used two different github API endpoints: contributors and languages. We found that github has a rate limit of 5,000 requests per hour with an API token so we ensured we used a token in the request. With ~10,000 links to process and two different endpoints, we had ~20,000 get requests to send. We introduced delays into the request code and tracking to read completed requests to ensure requests were not being rerun and were not being sent when they could not be completed.

Once we obtained information from the APIs about repository contributors and languages, we exploded these lists. One github repository could have multiple contributors and multiple languages used. From there we wrote the data into feather files which included CVE urls, contributors, and languages. Our data was then ingested into the graph.

We added contributors from github repositories as “GitHubUser” nodes. We then created links between these user nodes to the CVEs with a relationship “Written_By”. Essentially, (GitHubUser) - [Written_By] - (CVE) is the relationship created. We also added “Language” nodes from this github data and linked these to CVEs as well. Essentially, (Language) - [Written_In] - (CVE) is the relationship created.

Exploring the Graph:

1. How are vulnerabilities tagged?
    a. MATCH (c:CVEs)-[]-(t:Tags)
       WITH COUNT(c) AS CVEs, t
       RETURN CVEs, t.tag
       ORDER BY CVEs desc
    b. To get an idea of how vulnerabilities are tagged in the data. With this query we found that most of them are Third Party Advisories which means that a third party other than the company or institution with the vulnerability found the vulnerability and made it public before the company did. We also found a lot of the vulnerabilities were exploits and patches, with issue tracking and vendor advisory rounding out the top 5 types.
2. How many vulnerabilities are there per alert?
    a. MATCH (c:CVEs)-[]-(a:Alerts)
       WITH COUNT(c) as CVEs, a
       RETURN avg(CVEs)
    b. To get an idea of how many vulnerabilities are included in a single alert on average.
3. What is the average time between a CVE being published and an Alert being issued?
    a. MATCH (c:CVEs)-[:REFERENCED]-(a:Alerts)
       RETURN avg(duration.between(a.date, c.published)) AS Incubation

b. The average time between when a CVE is published and when an alert is issued is about 1 year and 4 months

Queries:

1. What geopolitical entities are related to the most severe vulnerabilities?

```
MATCH
(ac:Actors)<-[m:MENTIONED]-(at:Alerts)-[r:REFERENCED]->(c:CVEs)-[:OPEN_TO]-(v:AttackVectors)
WHERE ac.type = "GPE"
   AND NOT ac.best_label CONTAINS "united" // Removing US
   AND NOT ac.best_label CONTAINS "cisa"// Removing US
   AND NOT ac.best_label CONTAINS "us" // Removing US
   AND NOT ac.best_label CONTAINS "cyber" // Removing US
   AND NOT ac.best_label CONTAINS "lan" // Removing US
   AND NOT ac.best_label CONTAINS "north america" // Removing US
   AND NOT at.title CONTAINS "Top" // Removing Aggregated Alerts
   AND c.score >= 10 // PERFECT SCORE
RETURN ac,at,c,v
```

2. What attack vectors do Russians use the most?

```
MATCH
(ac:Actors)<-[m:MENTIONED]-(at:Alerts)-[r:REFERENCED]->(c:CVEs)-[:OPEN_TO]-(v:AttackVectors)
WHERE ac.type = "GPE"
   AND NOT ac.best_label CONTAINS "united" // Removing US
   AND NOT ac.best_label CONTAINS "cisa"// Removing US
   AND NOT ac.best_label CONTAINS "us" // Removing US
   AND NOT ac.best_label CONTAINS "cyber" // Removing US
   AND NOT ac.best_label CONTAINS "lan" // Removing US
   AND NOT ac.best_label CONTAINS "north america" // Removing US
   AND NOT at.title CONTAINS "Top" // Removing Aggregated Alerts
   AND c.score >= 10 // PERFECT SCORE
RETURN ac,at,c,v
```

3. Contributor centrality

```
CALL gds.graph.project('myGraph',
   ['GitHubUser', 'CVEs', 'Alerts','Configs','Language','AttackVectors','Tags'],
   {WRITTEN_BY: {orientation: 'undirected'},
   REFERENCED: {orientation: 'undirected'},
   OPEN_TO:{orientation: 'undirected'},
   AFFECTS: {orientation: 'undirected'},
   WRITTEN_IN: {orientation: 'undirected'},
   LINKED_TO: {orientation: 'undirected'}
   })

CALL gds.articleRank.stream('myGraph')
YIELD nodeId, score
WHERElabels(gds.util.asNode(nodeId))=["GitHubUser"]
RETURN gds.util.asNode(nodeId).login AS GitHubUser, score
```

```
ORDER BY score DESC

MATCH (n)-[a]-(o)-[b]-(p)
WHERE n.login="cclauss"
RETURN n,o,p, a,b
```

4. Community detection

   ```
   CALL gds.louvain.write('myGraph', { writeProperty: 'community' })
   YIELD communityCount, modularity, modularities

   MATCH (n)
   RETURN n.community, count(n) as cnt
   ORDER BY cnt desc

   MATCH (n)
   WHERE n.community=28649
   RETURN n
   ```

5. Which languages are the most popular amongst actors?
   a. Languages used the most overall in CVEs

      ```
      MATCH (c:CVEs)-[:WRITTEN_IN]->(lan:Language)
      RETURN lan.language, COUNT(c) as nums
      ORDER BY nums DESC
      ```

   b. Languages used the most by actors in alerts

      ```
      MATCH
      (a:Actors)<-[:MENTIONED]-(l:Alerts)-[r:REFERENCED]->(c:CVEs)-[:WRI
      TTEN_IN]->(lan:Language)
      WHERE NOT lan.language CONTAINS "message"
      RETURN lan.language, COUNT(a) as nums
      ORDER BY nums DESC
      ```

   c. Languages used the most by Russian actors

      ```
      MATCH
      (a:Actors)<-[:MENTIONED]-(l:Alerts)-[r:REFERENCED]->(c:CVEs)-[:WRI
      TTEN_IN]->(lan:Language)
      WHERE a.best_label CONTAINS "russia" AND NOT
      lan.language="message"
      RETURN lan.language, COUNT(a) as nums
      ORDER BY nums DESC
      ```

Insights:

- ~1.5 yrs between vulnerability publicly known and alert issued
- Albania was the most mentioned entity related to severe vulnerabilities
- Programming languages connected to most vulnerabilities had slightly different trends than those connected to vulnerabilities with alerts and actors

- Scripting languages are more popular overall while actors prefer procedural languages

Lessons Learned:

- API rate limits made getting the complete set of data difficult
- Separate entities by enriching with a property such as “domestic” vs “foreign” entities on the Actors nodes. This would have made it possible to sort through who or what the CISA alerts were addressing in a more directed context.
    - For example, US and China nodes being connected to the same alert have different contexts in this data. The US is being warned about attacks from China, but that distinction wasn’t possible in our current data.