Vulnerability Data Analysis with Google Sheets and Apps Script for Fun and Profit

Andrew Pollock

Who am I?

- Do a lot of aggregate analysis of CVE metadata
- Like to be able to visually eyeball data for patterns
- Love a good bit of Spreadsheet Engineering
    - Sorting
    - Filtering
    - Pivot tables
- Love JavaScript and Apps Script and a good bit of dynamism

Who are you?

- First time at VulnCon?
- Operates a CNA?
- Has to do vulnerability management at their organization?
- Does vulnerability research?
- How do you do things today?
- What are your hopes and dreams for this workshop?

What are we going to learn today?

- Everything you need to be able to do this on your own
- The wonderful solution that is github.com/bradjasper/ImportJSON
- The JSON REST APIs available for vulnerability metadata
  - CVE List
  - NVD
  - OSV.dev
  - GitHub Advisory Database
- How to create a Google Sheet template to easily get going with JSON REST APIs
- What's possible glueing this all together in a Google Sheet

Why would you want to do this?

- You need to know something about a set of vulnerabilities, by ID
- Quick and dirty vulnerability management by spreadsheet
- Visual inspection
- Anything tabular
- Filtering
- Pivot tables
- Why not?

What you need

- A Google Account (Gmail, Workspace)
- Internet access
- Optional: a browser extension for JSON output presentability

Browsing JSON API output

- I like https://github.com/arnav-kr/json-formatter (https://json-formatter.js.org/)
  - Chrome
    - https://chrome.google.com/webstore/detail/json-formatter/gpmodmeblccallcadopbcoeoejepgpnb
  - Firefox
    - https://addons.mozilla.org/firefox/addon/json_formatter/
  - Edge
    - https://microsoftedge.microsoft.com/addons/detail/json-formatter/hdebmbedhflilekbidmmdiaiilaegkjl

Housekeeping

- We’ve 90 minutes together
- Interactive
  - I’ll give some background
  - I’ll demonstrate
  - We’ll do it together
  - Lather, rinse repeat
- Stop me at any time
- There are no stupid questions!
- This is mildly an “unworkshop”
  - Let’s use it as an opportunity to explore specific use cases together
- I’d love the gift of your constructive feedback, positive or negative

Let’s get into it!

github.com/bradjasper/ImportJSON

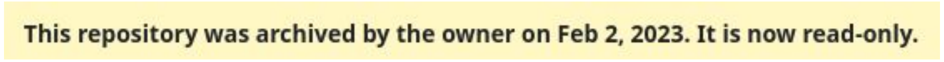
- Amazing canned Apps Script for querying JSON REST APIs
- Full credit to
  - Brad Jasper
  - Trevor Lohrbeer

github.com/bradjasper/ImportJSON

- Add this to a Google Sheet and you get these additional functions:

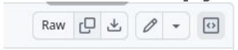
Function	Description
ImportJSON	For use by end users to import a JSON feed from a URL
ImportJSONFromSheet	For use by end users to import JSON from one of the Sheets
ImportJSONViaPost	For use by end users to import a JSON feed from a URL using POST parameters
ImportJSONBasicAuth	For use by end users to import a JSON feed from a URL with HTTP Basic Auth
ImportJSONAdvanced	For use by script developers to easily extend the functionality of this library

github.com/bradjasper/ImportJSON

- 
  This repository was archived by the owner on Feb 2, 2023. It is now read-only.
  - I highly recommend forking it, in case it disappears completely
  - Consider talking to Brad about taking it over if you like it and are an Apps Script/JavaScript aficionado

Creating a Google Sheet template with ImportJSON

This means you do the Apps Script legwork once

1. Copy the Apps Script code
   - https://github.com/bradjasper/ImportJSON/blob/master/ImportJSON.gs
   - Click the **`Copy raw file`** icon
   - 

2. Create a new Google Sheet
   - https://spreadsheet.new
   - Name it **`ImportJSON Template`**
3. Add the Apps Script
   - `Extensions ➡ Apps Script`
   - Name the project **`ImportJSON`**
   - Replace the boilerplate code with what you copied
   - (Optional) Rename **`Code.gs`** to **`ImportJSON.gs`**
   - Click the Save icon
   - Close the Apps Script tab
4. Bookmark this sheet as a template
   - Change the URL from **`/edit`** to **`/template/preview`**
   - Bookmark this URL

Here’s one I prepared earlier

- https://docs.google.com/spreadsheets/d/1Rdo09SBn_5Nf7vg1qMy9uZVdssboLC66dR2l0gFlmcc/template/preview
- https://tinyurl.com/ijgstemplate
- https://tinyurl.com/ijgsplayground

JSON REST APIs for vulnerability metadata

- 🗃️ CVE List
  - 💻 https://cveawg.mitre.org/api/cve
    - 👀 https://cveawg.mitre.org/api/cve/CVE-2024-3094
  - 📖 https://cveawg.mitre.org/api-docs/
  - ⏳ Not aware of any rate limiting
- 🗃️ NVD
  - 💻 https://services.nvd.nist.gov/rest/json/cves/2.0
    - 👀 https://services.nvd.nist.gov/rest/json/cves/2.0?cveId=CVE-2024-3094
  - 📖 https://nvd.nist.gov/developers/vulnerabilities
  - ⏳ Rate limited (less so with an API key)
- 🗃️ OSV.dev
  - 💻 https://google.github.io/osv.dev/get-v1-vulns/
    - 👀 https://api.osv.dev/v1/vulns/CVE-2024-3094
  - 📖 https://google.github.io/osv.dev/api/
  - ⏳ No rate limit
- 🗃️ GitHub Advisory Database
  - 💻 https://api.github.com/advisories/
    - 👀 https://api.github.com/advisories?cve_id=CVE-2024-3094
  - 📖 https://docs.github.com/en/rest/security-advisories/global-advisories
  - ⏳ Various rates apply

Pro tip: (Chrome) custom search engines

- Chrome
  - **`chrome://settings/searchEngines`**
  - Under `Site Search` add
    - `cve` ➡ `https://cveawg.mitre.org/api/cve/%s`
    - `nvd` ➡ `https://services.nvd.nist.gov/rest/json/cves/2.0?cveId=%s`
    - `ghsa` ➡ `https://api.github.com/advisories/%s`
    - `osv` ➡ `https://api.osv.dev/v1/vulns/%s`

Then you can just type in the omnibox

⌨️ `cve CVE-2024-3094`
⌨️ `nvd CVE-2024-3094`
⌨️ `ghsa CVE-2024-3094`
⌨️ `osv CVE-2024-3094`

Now for the fun*

Beginner: Basic (CVE List) CVE metadata

`e.g.` https://cveawg.mitre.org/api/cve/CVE-2024-21887

We want:

- `.cveMetadata.assignerShortName`
- `.containers.cna.descriptions[].value`

```
=ImportJSON("https://cveawg.mitre.org/api/cve/" & A2,
"/cveMetadata/assignerShortName,/containers/cna/descriptions
/value", "noHeaders")
```

Useful things to know

- Named Ranges
  - Useful for more readable and concise formulae
    - See the playground spreadsheet for examples
- Avoiding recalculations
  - Get the data *once* and then copy (and paste) the formula *values*
- Needing to do error checking on specific values requires a separate call to `ImportJSON`
  - The *whole* function call has to work
  - Incrementally build up to a single invocation that returns what you want

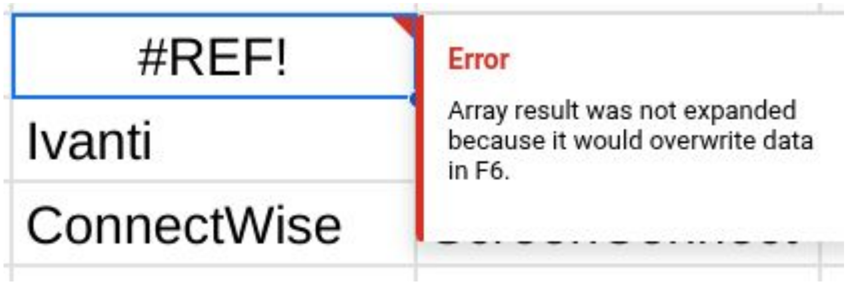
Beginner-Intermediate: Severity information

- Annoying
  - Is it CVSS v3 or CVSS v3.1?
  - Who knows!
- Need to be more fault-tolerant and try multiple keys

  -
    ```
    =FILTER(
       ImportJSON("https://cveawg.mitre.org/api/cve/" & A2,
    "/containers/cna/metrics/cvssV3_1/baseSeverity,/containers/cna/metric
    s/cvssV3_0/baseSeverity", "noHeaders"),
       ImportJSON("https://cveawg.mitre.org/api/cve/" & A2,
    "/containers/cna/metrics/cvssV3_1/baseSeverity,/containers/cna/metric
    s/cvssV3_0/baseSeverity", "noHeaders") <> ""
    )
    ```
  - May wind up with multiple API calls per CVE

Arrays

- We’re starting to push the friendship
  - Overwriting other populated cells is an error
- Need to somehow coerce the result into a single row (and potentially column)
- Your friends
  - `UNIQUE` – remove duplicates
  - `TRANSPOSE` – make multiple rows become multiple columns instead
  - `INDEX(1, 1)` – return only a specific row/column (i.e. cell)


#REF!
Ivanti
ConnectWise
Error
Array result was not expanded because it would overwrite data in F6.

Intermediate-Advanced: Vendors and Products

- Really pushes the limits due to variable-sized, matrixed results
- Use TEXTJOIN to merge multiple values into a single cell
  - This may impact on the utility of the data

Experimentation Time

Summary

- Use a template so the initial Apps Script setup is one-time
- Consider using Named Ranges as “constants” for more readable formulae
- If you need the values retrieved only once, consider copying and pasting the values to avoid unnecessary future recalculations

Thank you!