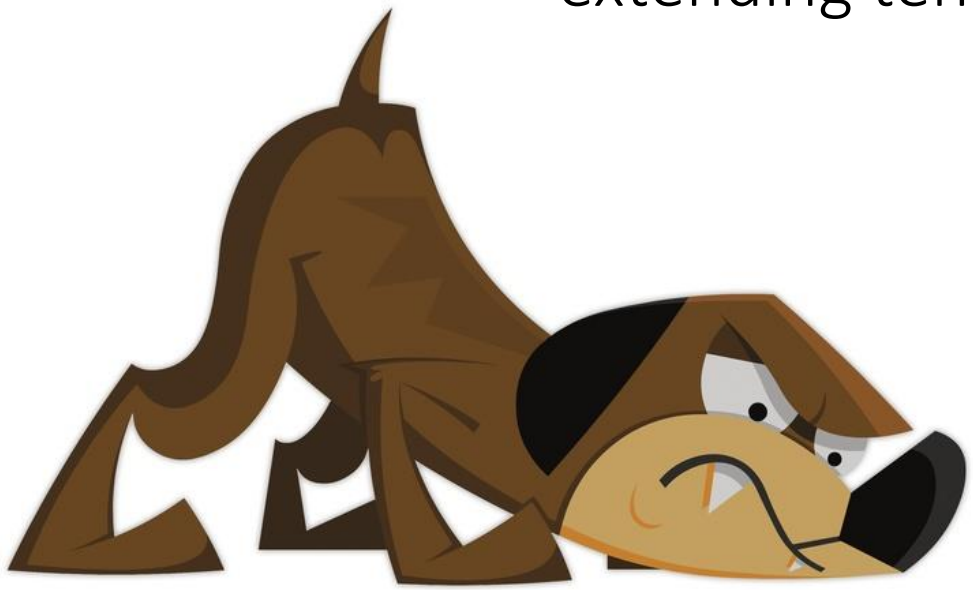
The power of the API

extending tenable.io to customize your data



Derek Hill

CSO Security Manager – HP Inc.

@secureITtoday

derek@dh-solutions.com

https://github.com/derekhillhp/tenable.io-presentation

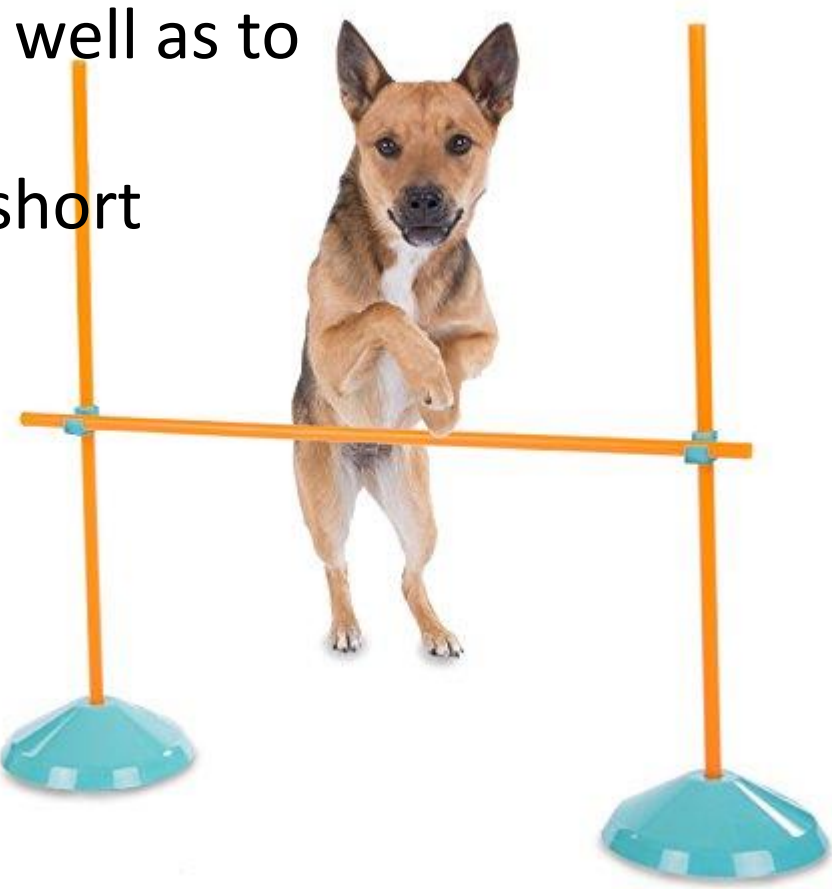
or http://bit.ly/ten-api

Our somewhat unique situation

- We are an internal consulting / service provider organization
  - 30+ customers (and growing)
  - 120+ AWS accounts
  - 3000+ instances under “management”
  - No centralized DevOps teams – many of our customers “manage” their own
  - Environment is fast changing
    - Short lifecycle of instances
    - New projects come up, often unannounced
  - Needed to get a better grip on what we have

tenable.io - challenges

- UI does provide a lot of detail – but it lacked customization (this is changing)
- We have a need for detailed reports for customer as well as to executive management
- Data needs to be current as the instance lifecycle is short
- Need to be able to integrate with AWS for tracking purposes
  - Need to be able to tell quickly which instances don’t have agents



tenable.io – to the rescue
(with the API)

- A lot more flexible and powerful than the UI
- Ability to extract data on a more granular level
- Pull just the data you need, nothing else (with some good scripting)
- Combine data across several “workbenches”

- tenable.io API – start here
- https://cloud.tenable.com/api#/overview - output in JSON format
- For more advanced scripting, there is also the tenable.io SDK
- https://github.com/tenable/Tenable.io-SDK-for-Python

Script development

- Created initial scripts done in-house
  - Start using the API in interactive mode, to see the data you are looking for and what data is being returned as well as the format (list, array, single value)
  - API output is in JSON format, but we wanted in CSV format for imports into Excel
- Reached out to Tenable PS for some help with one script
  - Data correlation challenge
- Continued refinement and maintenance of scripts
- Integrated other tools such as Security Monkey and in-house AWS tracking tool (Zeus)
- Creating new scripts can be addicting
  “what if I could see this….”

Code examples (don't worry, code is on github and link will be provided)

```
def getData(item):
    assetID, i, listLen, headers, baseURL = item
    print('Looking up info for asset ID: ', assetID, ' (', i, '/', listLen, ')', sep='')
    jsonResponse = getJsonFromRequest(baseURL + '/workbenches/assets/' + assetID + '/info', headers)
    if jsonResponse is not None:
        return jsonResponse['info']
    return None

def SaveAssetVulnerabilities(headers, nessusBaseURL, outputFolder):
    print('Looking up asset vulnerabilities')
    #Added ?date_range=7 to limit to systems seen in the last 7 days, if you want complete data simply remove
    response = getJsonFromRequest(nessusBaseURL + '/workbenches/assets/vulnerabilities?date_range=1', headers)
    #This next section is to take the output from the API call that contains arrays and store the values in separate columns.
    assets = response['assets']
    #Breaking down the list of fqdn's and IP's into separate colums
    for asset in assets:
        asset['fqdn-1'] = asset['fqdn'] [0]
        asset['ipv4-1'] = asset['ipv4'][0]
        if len(asset['fqdn']) > 1:
            asset['fqdn-2'] = asset['fqdn'][1]
        else:
            asset['fqdn-2'] = ""
……(removed code for formatting reasons)….
        else:
            asset['ipv4-3'] = ""
        #asset['Info'] = asset['severities'][0]['count'] <--Removed Info column as it is not used
        #Breaking down the list of vulnerabilities into separate columns, creating a total column and adding all the values
        asset['Low'] = asset['severities'][1]['count']
        asset['Medium'] = asset['severities'][2]['count']
        asset['High'] = asset['severities'][3]['count']
        asset['Critical'] = asset['severities'][4]['count']
        asset['Total'] = asset['Low'] + asset['Medium'] + asset['High'] + asset['Critical']
        #asset['fqdn-1'] = asset['fqdn'][0]
        del asset['severities']
        del asset['fqdn']
        del asset['ipv4']
        del asset['ipv6'] # <--removing for now as we are not using it, will have to add back to 'keys' list if re-enabled
    assets = {'assets': assets}
    currentdate = datetime.datetime.strftime(datetime.datetime.now(), '%Y-%m-%d')
    print('Saving asset vulnerabilities to asset_vulnerabilities.csv')
    keys = ['id', 'last_seen', 'ipv4-1', 'ipv4-2', 'ipv4-3', 'fqdn-1', 'fqdn-2', 'fqdn-3', 'Low', 'Medium', 'High', 'Critical', 'Total']
    json2csv(assets, outputFolder + 'asset_vulnerabilities-' + currentdate + '.csv', keys=keys)
```

Building the URL

Specifying the data and error handling

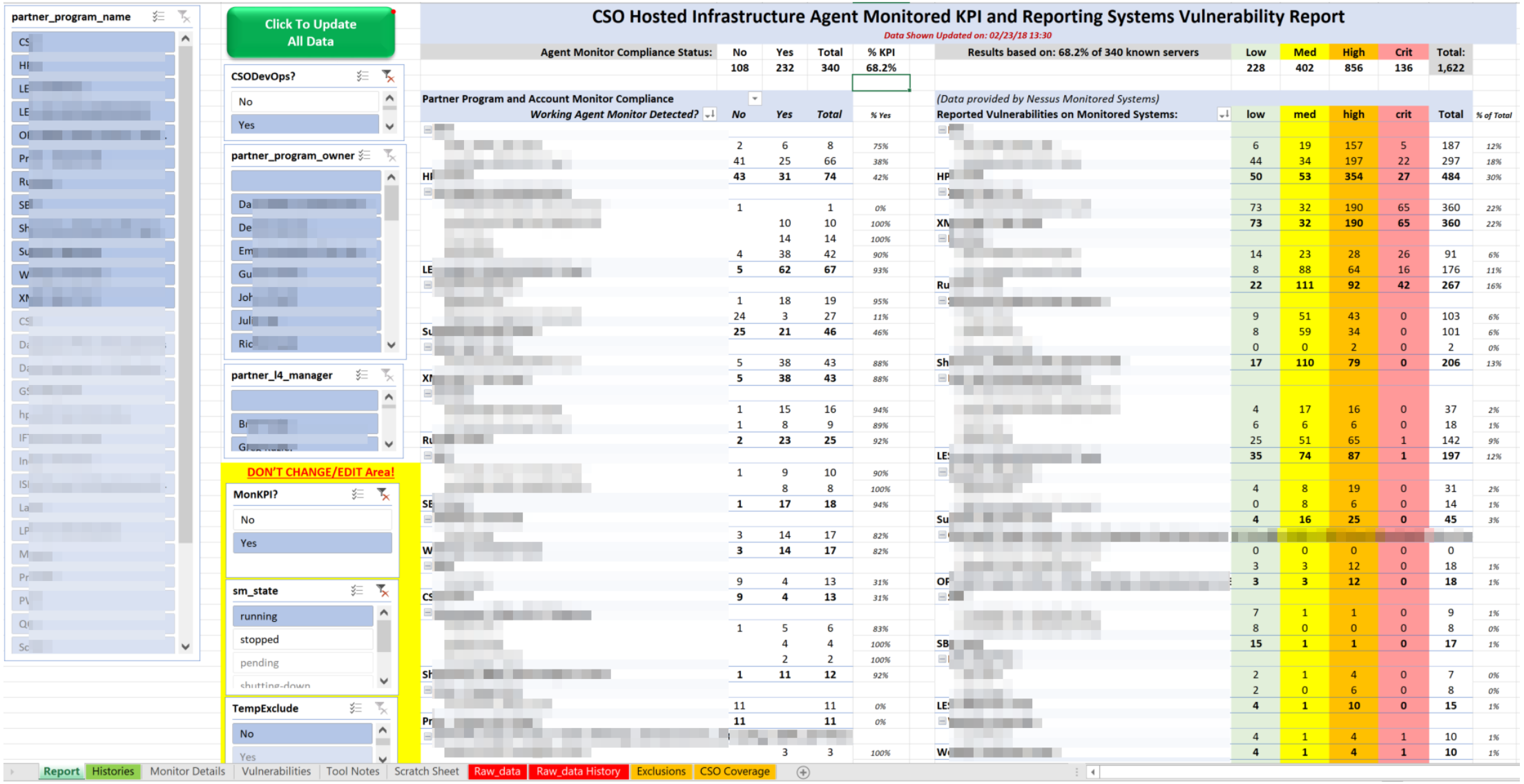
Specifying what data and CSV format

Results

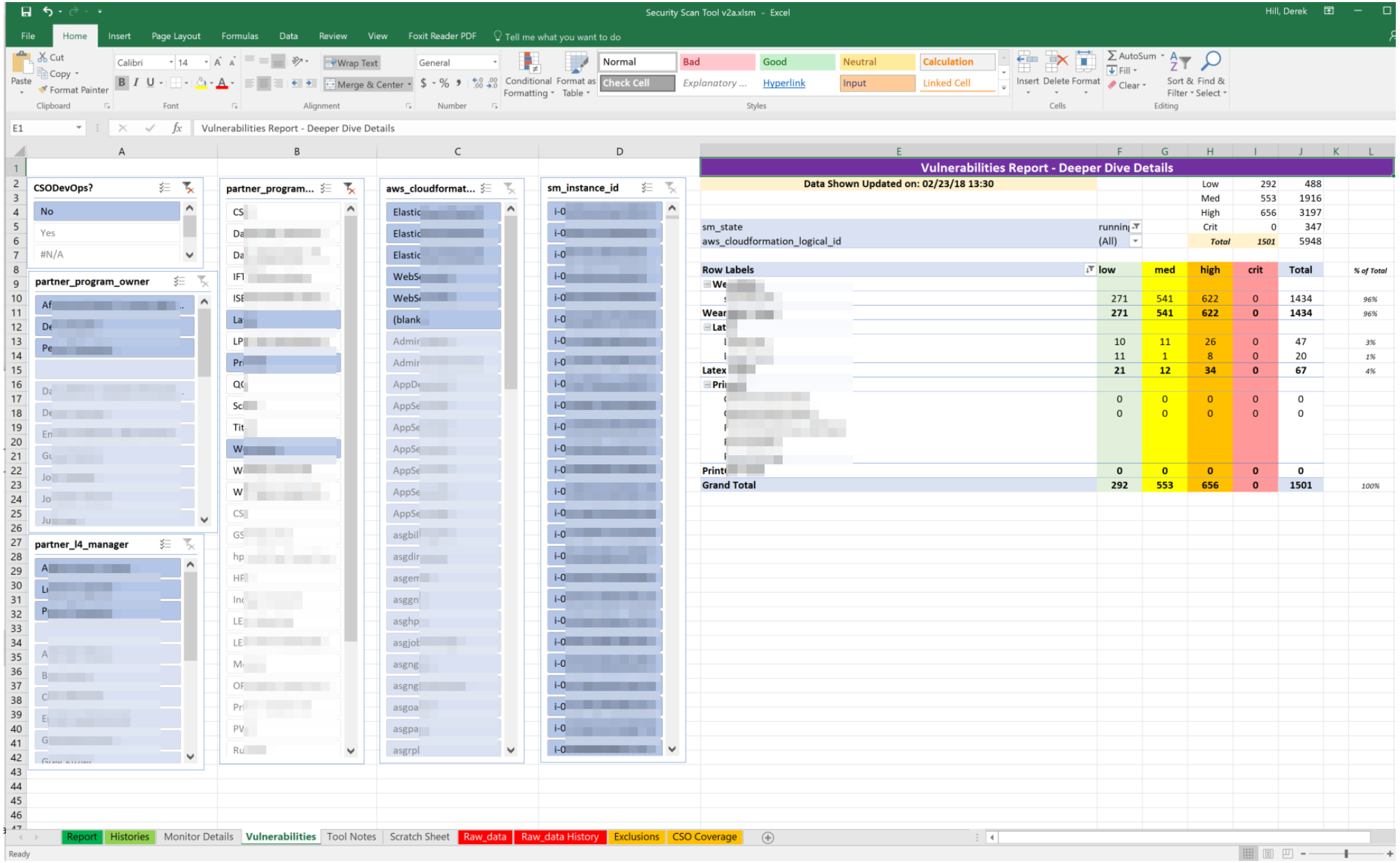
- Early results:
  - Initially, we used the CSV files and performed manual correlation using Excel. Obviously this is not scalable or sustainable. We started pulling the data into a RedShift db and combining it with other data such as Security Monkey & Zeus

- Current results:
  - Customized emails to customers
  - Dashboards
  - Customer reports (detailed)
  - Executive reports (aggregated and score)
  - Faster response times from customers to resolve vulnerabilities

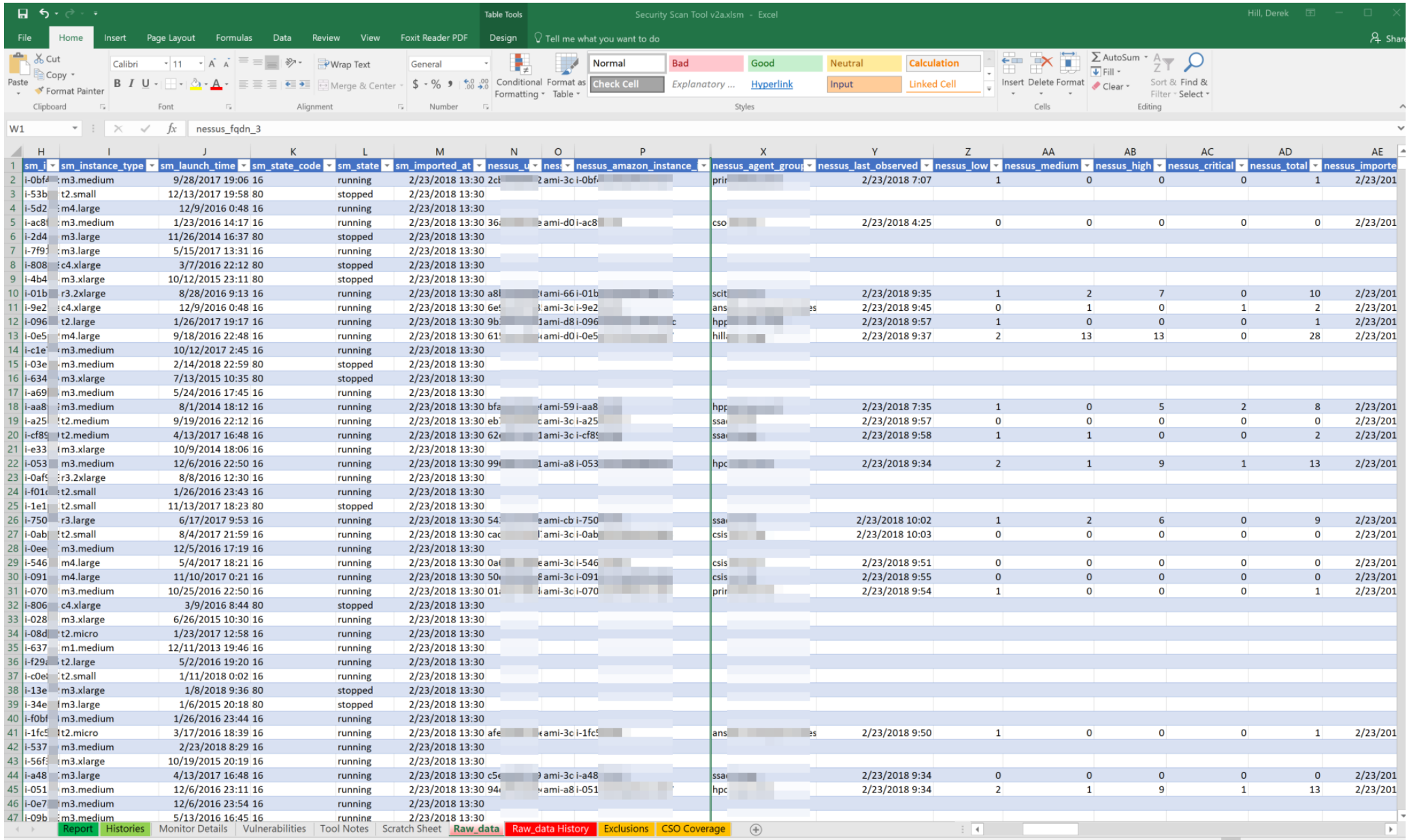
Sample Report in Excel (showing agent coverage as well as vulnerabilities by partner



Vulnerability detail by customer and instance



The raw data from multiple sources linked together



Script sharing

- Some of the code will be on GitHub for sharing
  (but not all due to copyright or intellectual property issues)
  - https://github.com/derekhillhp/tenable.io-scripts
  - Or http://bit.ly/api-scripts
  - Expect more scripts to be added over time

Questions

- Presentation can be downloaded here:
- https://github.com/derekhillhp/tenable.io-presentation or
- http://bit.ly/ten-api