# Project Title:

CSV Reporting API decomposition from Controller

# SCOPE OF WORK

Vendor shall provide the following work (“Work”) to Tenable:

Establish a new service written in Kotlin, with supporting infrastructure, monitoring and testing that provides equivalent CSV export functionality to the existing APIs as defined below.

Each endpoint expects a header auth in the form of:

X-ApiKeys: accessKey=ACCESS\_KEY;secretKey=SECRET\_KEY

Where ACCESS\_KEY and SECRET\_KEY are generated from the cloud.tenable.com portal after a valid Tenable.io user account has logged in.

The new service should replicate the existing functionality provided by the following cloud.tenable.com endpoints:

GET /workbenches/export

* Exports the specified workbench to a file.
* Returns the export ID of the requested export task (subsequently referred to as export\_id or file\_id in this document) in the form of:
  + @{file=export\_id}
* Query Parameters
  + format [string]
    - The file format to use (CSV)
  + report [string]
    - The type of workbench report to be exported (vulnerabilities)
  + start\_date [integer: int32]
    - The date (in unixtime) at which the exported results should begin to be included. Defaults to current day.
  + chapter [string]
    - Semicolon-separated list of chapters to include for vulnerabilities/hosts reports (vuln\_by\_plugin, vuln\_by\_asset, vuln\_hosts\_summary) or a single chapter for Executive Summary (exec\_summary).
  + date\_range [integer: int32]
    - The number of days of data prior to and including start\_date that should be returned. If not specified, data for all dates is returned.
  + filter.0.filter [string]
    - The name of the filter to apply to the exported scan report. You can find available filters by using the [GET /filters/workbenches/assets](https://developer.tenable.com/#filters-assets-filter) endpoint. If you specify the name of the filter, you must specify the operator as the filter.0.quality parameter and the value as the filter.0.value parameter. To use multiple filters, increment the <INDEX> portion of 'filter.<INDEX>.filter', for example, 'filter.0.filter' and ‘filter.1.filter’.
  + filter.0.quality [string]
    - The operator of the filter to apply to the exported scan report. You can find the operators for the filter using the [GET /filters/workbenches/assets](https://developer.tenable.com/#filters-assets-filter)endpoint. To use multiple filters, increment the <INDEX> portion of 'filter.<INDEX>.quality', for example, 'filter.1.quality'.
  + filter.0.value [string]
    - The value of the filter to apply to the exported scan report. You can find valid values for the filter in the 'control' attribute of the objects returned by the [GET /filters/workbenches/assets](https://developer.tenable.com/#filters-assets-filter)endpoint. To use multiple filters, increment the <INDEX> portion of 'filter.<INDEX>.value', for example, 'filter.1.value'.
  + filter.search\_type [string]
    - For multiple filters, specifies whether to use the AND or the OR logical operator. The default is AND.
  + minimum\_vuln\_info [Boolean]
    - When 'true', Tenable.io returns only a minimal subset of scan details for each result, excluding plugin attributes. In this case, only plugin\_output and vulnerability\_state fields are always returned; first\_found, last\_found and last\_fixed are also returned if possible.
  + plugin\_id [integer: int32]
    - A plugin ID. Restricts the export data to vulnerabilities that only the specified plugin detects.
  + asset\_id [string]
    - The UUID of an asset. Restricts the export data to findings on the specified asset only.

GET  /workbenches/export/{export\_id}/status

* Returns the status of a pending export. When an export has been requested, it is necessary to poll this endpoint until a "ready" status is returned, at which point the file is complete and can be downloaded using the [export download](https://developer.tenable.com/#workbenches-export-download) endpoint.
  + Example response:
    - @{progress\_total=0; progress=0; status=ready}
* Query Parameters
  + file\_id [integer: int32]
    - The unique identifier of the workbench report being exported. The value for this parameter can be obtained from the response of the initial export request (/workbenches/export).

GET  /workbenches/export/{export\_id}/download

* Downloads a CSV file that has been prepared for export after the initial export request (/workbenches/export).
* Query Parameters
  + file\_id [integer: int32]
    - The unique identifier of the workbench report being downloaded. The value for this parameter can be obtained from the response of the initial export request.

GET /filters/workbenches/vulnerabilities

* Returns the filters available for the Vulnerabilities Workbench.
  + Response should include all filter details served by the existing endpoint today. Information should be returned for each of the following filters:  
    host.id, plugin.attributes.bid, plugin.attributes.exploit\_framework\_canvas, plugin.attributes.canvas\_package, plugin.attributes.xref:CERT-CC, plugin.attributes.xref:CERT, plugin.attributes.exploit\_framework\_core, plugin.attributes.cpe, plugin.attributes.cve.raw, plugin.attributes.cvss\_base\_score, plugin.attributes.cvss\_temporal\_score, plugin.attributes.cvss\_temporal\_vector.raw, plugin.attributes.cvss\_vector.raw, plugin.attributes.cvss3\_base\_score, plugin.attributes.cvss3\_temporal\_score, plugin.attributes.cvss3\_temporal\_vector.raw, plugin.attributes.cvss3\_vector.raw, plugin.attributes.xref:CWE, compliance\_description, compliance\_reference, plugin.attributes.default\_account, plugin.attributes.exploit\_framework\_d2\_elliot, plugin.attributes.d2\_elliot\_name, plugin.attributes.exploit\_available, plugin.attributes.xref:EDB-ID, plugin.attributes.exploit\_framework\_exploithub, plugin.attributes.exploitability\_ease.raw, plugin.attributes.exploited\_by\_malware, plugin.attributes.exploited\_by\_nessus, host.target, plugin.attributes.xref:IAVA, plugin.attributes.xref:IAVB, plugin.attributes.stig\_severity, plugin.attributes.xref:IAVT, plugin.attributes.in\_the\_news, plugin.attributes.malware, plugin.attributes.exploit\_framework\_metasploit, plugin.attributes.metasploit\_name, plugin.attributes.xref:MSFT, plugin.attributes.xref:OSVDB, plugin.attributes.patch\_publication\_date, plugin.attributes.description, plugin.family\_id, plugin.id, plugin.attributes.plugin\_modification\_date, plugin.name, output, plugin.attributes.plugin\_publication\_date, plugin.attributes.plugin\_type, port.port, port.protocol, modification, plugin.attributes.xref:Secunia, plugin.attributes.see\_also, severity, plugin.attributes.solution, plugin.attributes.synopsis, target\_group, plugin.attributes.unsupported\_by\_vendor, tracking.first\_found, tracking.last\_found, plugin.attributes.vpr.score, plugin.attributes.vuln\_publication\_date, tracking.state
  + and any additional filters as they are added to the list of available filters.

Endpoint definitions should match existing contracts whenever possible based on published standards in Tenable.io public API documentation

* All code shall be versioned using SCM, specifically GitHub, as provided by Tenable.
* Code will be delivered incrementally during development.
* Code will adhere to team quality requirements of Tenable and will be reviewed by Tenable every two weeks.
* New service architecture must differ from the existing service by splitting data processing functions from export generation
* New service must differ from the existing service by receiving asset information from the Asset Service instead of Elasticsearch queries

USE CASES  
  
Each of the HTTP requests in the subsequent examples assume the user is including an auth header in the following format:  
  
X-ApiKeys: accessKey=ACCESS\_KEY;secretKey=SECRET\_KEY

Where ACCESS\_KEY and SECRET\_KEY are generated from the cloud.tenable.com portal after a valid Tenable.io user account has logged in.

# Generating the CSV workbench export for all vulnerabilities with a vulnerability publication date earlier than July 15th, 2019.

# Steps:

# Initiate the generation of the report by sending a GET request to the following endpoint: https://cloud.tenable.com/workbenches/vulnerabilities?date\_range=30&filter.0.quality=date-lt&filter.0.filter=plugin.attributes.plugin\_publication\_date&filter.0.value=2019%2F07%2F15&filter.search\_type=and

# Receive the following response from the request:

# @{file=1403201051}

# Request the status of the creation of the report by sending a GET request to the following endpoint: https://cloud.tenable.com/workbenches/export/1403201051/status

# Receive the following response from the request: @{progress\_total=0; progress=0; status=ready}

# Request the download of the report by sending a GET request to the following endpoint: https://cloud.tenable.com/workbenches/export/1403201051/download

# Receive the expected report in CSV format (output trimmed for brevity): Plugin ID,CVE,CVSS,Risk,Host,Protocol,Port,Name,Synopsis,Description,Solution,See Also,Plugin Output,Asset UUID,Vulnerability State,IP Address,FQDN,NetBios,OS,MAC Address,Plugin Family,CVSS Base Score,CVSS Temporal Score,CVSS Temporal Vector,CVSS Vector,CVSS3 Base Score,CVSS3 Temporal Score,CVSS3 Temporal Vector,CVSS3 Vector,System Type,Host Start,Host End,Vulnerability Priority Rating (VPR)

# "119241","CVE-2018-15715","9.0","High (etc.)

# Get the list of available workbench filters, and then generating a CSV workbench export for all active vulnerabilities that were first identified prior to June 15th, 2019. Steps:

# Get the list of available workbench filters by sending a GET request to the following endpoint: https://cloud.tenable.com/filters/workbenches/vulnerabilities

# Receive the following response from the request (output trimmed to relevant filters for brevity): name : tracking.first\_found

# readable\_name : Vulnerability First Seen

# control : @{type=datefield; regex=^[0-9]{4}/[0-9]{2}/[0-9]{2}$; readable\_regex=YYYY/MM/DD}

# operators : {date-lt, date-lte, date-gt, date-gte...}

# group\_name :

# name : tracking.state

# readable\_name : Vulnerability State

# control : @{type=dropdown; list=System.Object[]}

# operators : {eq, neq}

# group\_name : (etc.)

* Initiate the generation of the report by sending a GET request to the following endpoint:https://cloud.tenable.com/workbenches/vulnerabilities?date\_range=30&filter.0.quality=date-lt&filter.0.filter=tracking.first\_found&filter.0.value=2019%2F07%2F15&filter.1.quality=eq&filter.1.filter=tracking.state&filter.1.value=Active&filter.search\_type=and

# Receive the following response from the request:

# @{file=2051428165}

# Request the status of the creation of the report by sending a GET request to the following endpoint: https://cloud.tenable.com/workbenches/export/2051428165/status

# Receive the following response from the request: @{progress\_total=0; progress=0; status=ready}

# Request the download of the report by sending a GET request to the following endpoint: https://cloud.tenable.com/workbenches/export/2051428165/download

# Receive the expected report in CSV format (output trimmed for brevity):

Plugin ID,CVE,CVSS,Risk,Host,Protocol,Port,Name,Synopsis,Description,Solution,See Also,Plugin Output,Asset UUID,Vulnerability State,IP Address,FQDN,NetBios,OS,MAC Address,Plugin Family,CVSS Base Score,CVSS Temporal Score,CVSS Temporal Vector,CVSS Vector,CVSS3 Base Score,CVSS3 Temporal Score,CVSS3 Temporal Vector,CVSS3 Vector,System Type,Host Start,Host End,Vulnerability Priority Rating (VPR)

# "119241","CVE-2018-15715","9.0","High… (etc.)

# WRITTEN REPORTS

Vendor shall send to Tenable written reports on a 1 week basis which shall include:

The past week’s accomplishments, the next week’s goals, any outstanding blockers, burn up, cumulative flow diagram and estimated completion date.

Vendor shall provide all material and resources (including personnel) necessary to complete the work at no additional cost to Tenable. Upon completion of the SOW, the Vendor shall deliver to Tenable the following items (“Deliverables”):

# MILESTONES / DELIVERABLES

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Milestones/Deliverables** | | **Due Date** | **Amount Payable upon Completion & Acceptance** |
| a. | Automation for unit, component, integration test execution with code coverage of at least 70% | |  |  |
| b. | Metrics for monitoring the new service   * Synthetic tests composed of the following automated checks:   + POST to /workbenches/export   + GET to /workbenches/export/{export\_id}/download   + GET to /workbenches/export/{export\_id}/status   + GET to /filters/workbenches/vulnerabilities     - Each check should be considered its own metric, and not be tested in conjuction with other checks as outlined above     - A check is to be considered successful when the REST query as described above receives a response other than an error     - A check is to be considered unsuccessful when a REST query as described above receives an error as a response. | |  |  |
| c. | Dashboard for the service   * Dashboard to be created in Tenable.io Datadog instance and show time series visualizations of metrics from Milestone c. spanning the past 24 hours. | |  |  |
| d. | Demonstration of new service   * Demonstration conducted by Pivot must include usage of the new services per the use cases as described in section 4. * Review of monitoring solution including technical implementation and display of metrics in Datadog per Milestone d. | |  |  |
|  | |  |  |  |