Exporting Data via the Kenna API

Whether it's appeasing auditors or collecting data for historical analysis in a data warehouse, there are several use cases for exporting the wealth of data that Kenna collects and processes. Kenna has many customers making creative use of this functionality such as importing Kenna data into Splunk and other Elasticsearch/Logstash/Kibana platforms.

Thankfully Kenna makes it easy and painless to perform bulk exports of your entire Kenna data. With the power of the data in your hands there really isn't a limit to what you can accomplish.

Kenna allows users to export three different model types: Asset, Vulnerability, and Fix. Let's take a quick look at what data is included in each data model.

The asset export includes all the asset metadata that Kenna contains for a given asset. This export does not include vulnerabilities, or the fixes for those vulnerabilities. You'll be able to see the asset priority, operating system, Kenna locator, risk score, count of vulnerabilities on the asset, the associated tags and risk meters, and much more.

When pulling the vulnerability export the data set will show each individual finding of every vulnerability in your environment. So if CVE-2017-5638 is detected on 50 assets in your environment and CVE-2019-0708 is on 100 assets then 150 vulnerabilities will be included in the export. Each vulnerability will include the asset ID for the respective asset so that you can cross reference to the asset export to grab the asset metadata. The vulnerability export model will include the vulnerability score provided by the scanner, the Kenna risk score, the CVE ID and description, the connector where the vulnerability was detected, any custom fields, and basic threat intelligence data.

Lastly, the Fix export will include the title, diagnosis, and solution for each relevant vulnerability fix in the environment. The export will include vendor data and links where appropriate, as well as the list of assets and CVEs that are associated with the fix.

Now that we understand the data that is available for export, let's get started on actually pulling the data out.

Exports are available in two data formats -- .xml and .json -- and are delivered in a compressed file format called "gzip". Once downloaded, .gzip files can be unzipped using free Windows software such as WinZip, 7-Zip, or the gzip command built into Linux distributions.

How to Create a Data Export

Downloading your desired data export via the Kenna API is a two step process. First, you'll need to request the data export with body parameters defining your intended export scope, then use the search id provided in the response to then request the actual download of the export gzip.

Here is an example of cURL POST request to download an export of all open vulnerabilities in json format:

```
curl -H "X-Risk-Token: <token>" -H "Content-type: application/json"
"https://api.kennasecurity.com/data_exports" -X POST -d '{
    "status" : ["open"],
    "export_settings" : {
        "format": "json",
        "model": "vulnerability"
    }
}'
```

Which would return a response of { "search_id": 1, "record_count": 2,000}. This lets us know that the request was successful and 2,000 vulnerabilities will be included in the gzip file once the search_id of 1 is used in the next API call. Let's take a look at that request.

```
curl -H "X-Risk-Token: <token>" "https://api.kennasecurity.com/data_exports?search_id=1" -X
GET -o kenna_asset_export.gzip
```

Once we unzip the output .gzip file, we'll end up with a .json file that looks like this.

```
{
  "meta": {
    "total_count": 2,000
  },
  "vulnerabilities": [
    {
      "id": 629607,
      "status": "open",
      "closed_at": null,
      "created_at": "2020-11-06T21:18:21Z",
      "due_date": null,
      "notes": null,
      "port": [
```

```
      3389
    ],
    "priority": 10,
    "identifiers": [
      "18405"
    ],
    "last_seen_time": "2013-07-01T11:47:23.000Z",
    "scanner_score": 2,
    "fix_id": 1770971,
    "scanner_vulnerabilities": [
      {
        "port": 3389,
        "external_unique_id": "18405",
        "open": true
      }
    ],
    "asset_id": 11012,
    "connectors": [
      {
        "id": 156474,
        "name": "Nessus XML",
        "connector_definition_name": "Nessus XML",
        "vendor": "Tenable"
      }
    ],
    "service_ticket": null,
    "urls": {
      "asset": "api.kennasecurity.com/assets/11012"
    },
    "solution": "- Force the use of SSL as a transport layer for this service if supported, or/and\n- Select the 'Allow connections only from computers running Remote Desktop with Network Level Authentication' setting if it is available.",
    "patch": true,
    "patch_published_at": null,
    "cve_id": "CVE-2005-1794",
    "cve_description": "Microsoft Terminal Server using Remote Desktop Protocol (RDP) 5.2 stores an RSA private key in mstlsapi.dll and uses it to sign a certificate, which allows remote attackers to spoof public keys of legitimate servers and conduct man-in-the-middle attacks.",
    "cve_published_at": "2005-06-01T04:00:00.000Z",
    "description": null,
    "wasc_id": null,
    "severity": 6,
    "threat": 10,
    "popular_target": true,
```

```
  "active_internet_breach": false,
  "easily_exploitable": false,
  "malware_exploitable": false,
  "predicted_exploitable": false,
  "custom_fields": [
    {
      "name": "Custom Field Dummy",
      "custom_field_definition_id": 1,
      "value": null
    }
  ],
  "first_found_on": "2020-11-06T21:18:21Z",
  "top_priority": false,
  "risk_meter_score": 32,
  "closed": false
},...}
```


Optimizing Data Exports

Now that we have a firm grasp on how exports work, let's spend some time optimizing our process. As your organization and vulnerability management program grows, the file size of your data exports will exponentially increase over time, causing your generation and download times to potentially become prohibitive. It's important that we try to trim down our exports as much as possible.

Before you establish a defined scheduled export process, take some time to consider your scope. Do you have a compliance requirement or business need to export fix data? Are you only concerned with open vulnerabilities on active assets, or do you need to export closed vulnerabilities and inactive assets as well? This type of data scoping is easily accomplished by changing the body parameters to match your objective.

Also consider the regularity of your exports. Do you need this data refreshed on a monthly basis? Weekly? Daily?

Lastly, we highly, highly recommend using the incremental exports feature so that you're only exporting data that is new or changed since your previous export. This is by far the best way to ensure your file size remains reasonable. Of course you'll have to export the totality of your Kenna environment the first go-round, however once you have that data on hand there really isn't any need to export data you already have stored locally.

The Kenna API accepts both relative time differences as well as literal timestamps via the “records_update_since” parameter. Here’s an example of a request to export only active assets that have been created or updated in the past week:

```
curl -k -H "X-Risk-Token:{api_token}" -H "Content-type: application/json"
"https://api.kennasecurity.com/data_exports" -X POST -d '{
    "status" : ["active"],
    "records_updated_since" : "now-1w",
    "export_settings" : {
        "format": "json",
        "model": "asset"
    }
}'
```