

CVE Search Function for IBM Resilient

Table of Contents

- [About This Package](#)
- [Prerequisites](#)
- [Installation](#)
- [Data Table](#)
- [Function Inputs](#)
- [Function Output](#)
- [Pre-Process Script](#)
- [Post-Process Script](#)
- [Rules](#)
- [CVE Search Function Usage](#)

About This Package:

This package contains a Resilient Function that allows you to search for Common Vulnerability Exposures (CVE) from Database.

- Function implements different ways to search the database such as:
 - Browse
 - Search
 - Specific CVE ID
 - Last 30 CVE's
 - CVE Database Information
- The function makes use of the CVE https://cve.circl.lu/api/{search_param}/{vendor name}/{product name} API call to get information on a given query
- For more information see: [CVE Search Database](#)

Customization Settings

Layouts Rules Scripts **Workflows** Functions Message Destinations Phases & Tasks Incident Types Breach Artifacts

Workflows

New Workflow

cve S

| Workflow Name | Description | Object Type | Rules | |
|-------------------------------------|---|-------------|-------------------------------------|----|
| Example: CVE Search | Search Data for Common Vulnerability from CVE Database | Artifact | Example: CVE Search | 🗑️ |
| Example: CVE Browse | Browse Common Vulnerability Exposures Database for vendors and products and database information. | Incident | Example: CVE Browse | 🗑️ |

CVE Search Function layout:

The screenshot shows the 'Customization Settings' for the 'Example: CVE Search' workflow in the Resilient platform. The interface includes a top navigation bar with 'Dashboards', 'Simulations', 'Incidents', 'Create', and 'Loading' buttons. Below this, a tabbed interface shows 'Workflows' selected, with other tabs like 'Layouts', 'Rules', 'Scripts', 'Functions', 'Message Destinations', 'Phases & Tasks', 'Incident Types', 'Breach', and 'Artifacts'. The 'Workflows' tab is further divided into 'Example: CVE Search'. On the right, there are buttons for 'Cancel', 'Save & Close', and 'Save'. The main form contains fields for 'Name' (Example: CVE Search), 'API Name' (example_cve_search), 'Description' (Search Data for Common Vulnerability from CVE Database), and 'Object Type' (Artifact). A metadata section on the right lists 'Creator' (Nitin Kandhare), 'Last Modified' (02/20/2019 14:58), 'Last Modified By' (Nitin Kandhare), and 'Associated Rules' (Example: CVE Search). Below the form is a visual workflow diagram showing a 'Start your workflow here' node connected to a 'CVE Search' function node, which then connects to an output node. A text box explains: 'This CVE Search function is intended to search Vulnerability data from CVE database by using RESTfull API, and different searching methods like search on specific vendor and product vulnerability CVE ID etc. Returns Vulnerability data as per given search input parameters.'

CVE Search Pre-Process Script

The screenshot shows the 'Pre-Process Script' editor for the CVE Search function. The interface includes tabs for 'Input', 'Pre-Process Script', 'Output', and 'Post-Process Script'. The 'Pre-Process Script' tab is active, showing a Python script with the following code:

```
1 inputs.cve_search_data = artifact.value
2 inputs.cve_search_criteria = rule.properties.cve_search_criteria
3 inputs.cve_id = rule.properties.cve_id
4 inputs.cve_vendor = rule.properties.vendor
5 inputs.cve_product = rule.properties.product
6 inputs.cve_published_date_from = rule.properties.cve_published_date_from
7 inputs.cve_published_date_to = rule.properties.cve_published_date_to
```

CVE Search Post-Process Script

The screenshot shows the 'Post-Process Script' editor for the CVE Search function. The interface includes tabs for 'Input', 'Pre-Process Script', 'Output', and 'Post-Process Script'. The 'Post-Process Script' tab is active, showing a Python script with the following code:

```
1 #globals
2 ENTRY_TO_DATATABLE_MAP = {}
10
11 api_call_type = results['api_call']
12 output_data = results['content']
13 api_call_type_text = "<p><b>api call type :</b> {}</p>"
14 browse_rich_text = "<p><b>{}</b></p>"
15 rich_text_tmp = ""
16
17 #Adding data to table
18 ref_link_text = ""
19 if output_data:
20     for dict_element in output_data:
54 else:
55     incident.addNote("No Data Returned from CVE Search..!")
```

CVE Browse Function layout:

Resilient Dashboards Simulations Incidents **Create** Loading

Search Nitin Kandhare POC-IBM-Dev

Customization Settings

Layouts Rules Scripts **Workflows** Functions Message Destinations Phases & Tasks Incident Types Breach Artifacts

Workflows / Example: CVE Browse

Name * Example: CVE Browse

API Name * example_cve_browse

Description Browse Common Vulnerability Exposures Database for vendors and products and database information.

Object Type * Incident

Creator Nitin Kandhare
Last Modified 02/20/2019 14:58
Last Modified By Nitin Kandhare
Associated Rules Example: CVE Browse

Start your workflow here

This CVE Browse function is intended to Browse Vulnerability data from CVE database by using RESTfull API, and different Browse methods like Browse for all vendor & product information or for specific vendor's product details and get more information about the current vulnerability databases in use and when it was updated. Returns Vulnerability data as per given search input parameters.

CVE Browse Pre-Process Script

Input Pre-Process Script Output Post-Process Script

Language: Python Theme light Mode Default Tab Size 5 - Font + Font

```

1 inputs.cve_browse_data = incident.name
2 inputs.cve_browse_criteria = rule.properties.cve_browse_criteria
3 inputs.cve_vendor = rule.properties.vendor

```

CVE Browse Post-Process Script

Input Pre-Process Script Output **Post-Process Script**

Language: Python Theme light Mode Default Tab Size 5 - Font + Font

```

1 api_call_type = results['api_call']
2 output_data = results['content']
3 api_call_type_text = "<p><b>api call type :</b> {}</p>"
4 browse_rich_text = "<p><b>{}&ensp;&ensp</b>{}&ensp;&ensp</p>"
5 rich_text_tmp = ""
6 #Adding Browse data and Database information Notes Section
7 api_call_type_text = api_call_type_text.format(api_call_type)
8 browse_rich_text_final = ""
9-#if api_call_type == 'browse':
10- if output_data:
11-     for x in output_data:
12-         for key_data,value_data in x.items():
13-             text = browse_rich_text.format(key_data,value_data)
14-             api_call_type_text += text
15-         browse_rich_text_final = helper.createRichText(api_call_type_text)
16- else:
17-     browse_rich_text_final = 'No Searched Data returned...!'
18 incident.addNote(browse_rich_text_final)

```

Prerequisites:

- Resilient Appliance >= v31.0.0
- Integrations Server running resilient_circuits >= v30.0.0

Installation

This package requires that it is installed on a RHEL or CentOS platform and uses the resilient-circuits framework.

- Download the **.zip** file from our App Exchange and extract it. You will find a file called:
fn_cve_search-<version>.tar.gz
- Copy this file to your Integrations Server
- To install the package, run:

```
$ pip install fn_cve_search-<version>.tar.gz
```

- To import the function, example rules, data tables and workflows into your Resilient Appliance, run:

```
$ resilient-circuits customize -y -l fn-cve-search
```

- To update your **app.config** file with the required CVE Search configurations, run:

```
$ resilient-circuits config -u
```

- Then open your **app.config** file and check the following configuration data is added:

```
[fn_cve_search]
# Flag display maximum CVE Entries on the resilient table
max_results_display = 50
# Base URL of Common Vulnerability Exposures Data Base.
cve_base_url = https://cve.circl.lu/api
```

Edit the **max_results_display** counter value to limit the maximum no of search results to display on table.

- To uninstall CVE Function from Resilient, run:

```
$ pip uninstall fn_cve_search
```

Data Table

Data Table Utils: CVE Searched Data

CVE Searched Data

Search...



| CVE ID | Published Date | Summary | References | Vulnerability Config | Vulnerable Config Cpe 2 2 |
|-------------|----------------|-------------|--|--|--|
| lorem ipsum | 03/21/2019 | lorem ipsum | <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer et viverra lorem, quis porta tortor.</p> <p>Aliquam euismod dignissim lectus, nec malesuada magna fermentum vel. Ut venenatis molestie augue quis volutpat.</p> <p>Donec pretium dui vitae suscipit fermentum. In finibus ligula sed ipsum scelerisque volutpat.</p> <p>Pellentesque vel sagittis diam. Pellentesque sodales</p> | <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer et viverra lorem, quis porta tortor.</p> <p>Aliquam euismod dignissim lectus, nec malesuada magna fermentum vel. Ut venenatis molestie augue quis volutpat.</p> <p>Donec pretium dui vitae suscipit fermentum. In finibus ligula sed ipsum scelerisque volutpat.</p> <p>Pellentesque vel sagittis diam. Pellentesque sodales</p> | <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer et viverra lorem, quis porta tortor.</p> <p>Aliquam euismod dignissim lectus, nec malesuada magna fermentum vel. Ut venenatis molestie augue quis volutpat.</p> <p>Donec pretium dui vitae suscipit fermentum. In finibus ligula sed ipsum scelerisque volutpat.</p> <p>Pellentesque vel sagittis diam. Pellentesque sodales</p> |

API Name :

cve_data

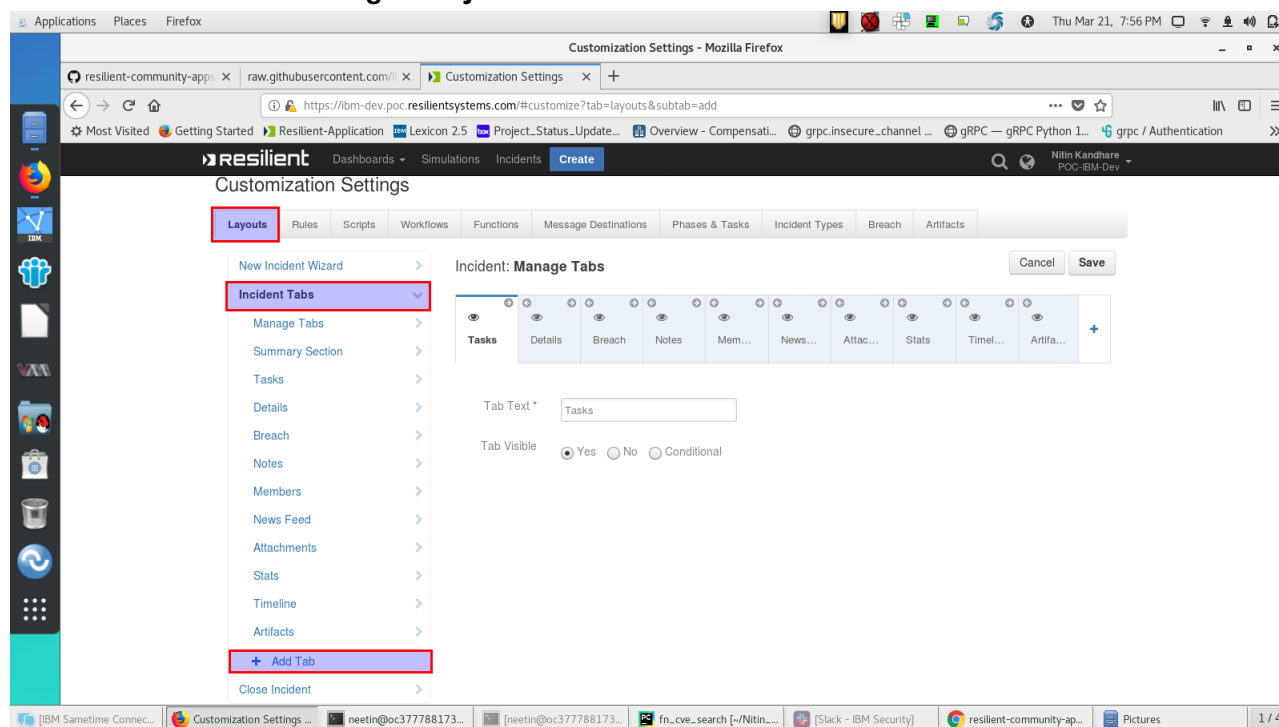
Columns:

| Column Name | API Access Name | Type |
|---------------------------|----------------------------------|-------------|
| CVE ID | cve_id | Text |
| Published Date | published_date | Date Picker |
| Summary | summary | Text |
| References | references | Text Area |
| Vulnerability Config | vulnerability_configuration | Text Area |
| Vulnerable Config Cpe 2 2 | vulnerable_configuration_cpe_2_2 | Text Area |

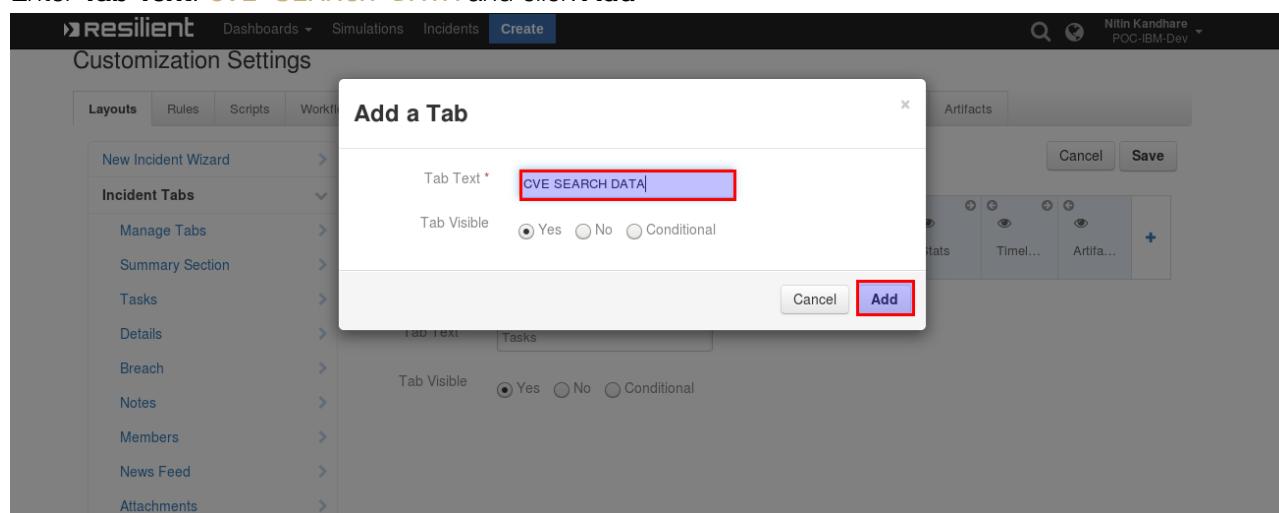
Display the Data table in an Incident

- In order to **display** the CVE Searched Data Table in your Incident, you must **modify your Layout Settings**

1. Go to **Customization Settings > Layouts > Incident Tabs > + Add Tab**



2. Enter **Tab Text: CVE SEARCH DATA** and click **Add**



3. Drag the Data table into the middle and click **Save**

The screenshot shows the 'Customization Settings' page in the Resilient interface. The 'Incident: CVE SEARCH DATA' tab is selected. On the left, there's a sidebar with 'Incident Tabs' and 'CVE SEARCH DATA' checked. On the right, the 'Data Tables' section lists several tables, with 'CVE Searched Data' highlighted by a red box. A red arrow points from this box to the 'Incident: CVE SEARCH DATA' header area. The 'Save' button is also highlighted with a red box.

4. Create a new Incident and you will now see the **My Test Tab** with the **** CVE Searched Data Table****

Function Inputs

CVE Search Function

| Input Name | Type | Required | Example | Info |
|-------------------------|-------------|----------|---------------|--|
| cve_search_data | String | yes | python | A artifact data to search for cve |
| cve_search_criteria | Select | yes | Search | CVE Search Criteria i.e Search(For Specific Product from Vendor),CVE ID(get cve per cve id),Last 30 CVEs(get last 30 cves including CAPEC CWE and CPE Expansion) |
| cve_id | String | yes | CVE-2008-3949 | Specific vulnerability ID |
| cve_vendor | String | yes | microsoft | a vendor name to search for cve |
| cve_product | String | yes | excel | Name of the Product to Search in CVE Database |
| cve_published_date_from | Date Picker | yes | 03/01/2019 | Select CVE Published Date |

| Input Name | Type | Required | Example | Info |
|-----------------------|-------------|----------|------------|-----------------------------------|
| cve_published_date_to | Date Picker | yes | 03/01/2019 | End date range to search cve data |

CVE Browse Function

| Input Name | Type | Required | Example | Info |
|---------------------|--------|----------|----------------------|---|
| cve_browse_data | Text | yes | Name of the Incident | A incident Name to be browse for vendor and product |
| cve_browse_criteria | select | yes | Browse | CVE Browse Criteria i.e Browse(For Vendors & Product),CVE DB Info(get information about current cve database) |
| cve_vendor | text | yes | apple | a vendor name to browse for cve |

Function Output

- The payload from the function will contain the JSON from the CVE API Call and the name of the API Call

```
results = {
  "content": #JSON returned from CVE API Call,
  "api_call": #"last"/"browse"/"search"/"cve"/"db"
}
```

- To see the output of each of the API calls for this Function, we recommend running **resilient-circuits** in **DEBUG** mode.
- To do this run:

```
$ resilient-circuits run --loglevel=DEBUG
```

Pre-Process Script

- CVE Browse

This example sets the **cve_browse_data**, **cve_browse_criteria**, **cve_vendor** inputs to the name of incident and entered vendor name and selections to user took action on

```
# Name of the Incident
inputs.cve_browse_data = incident.name
# Search type browse on CVE Data base (may be Browse, CVE DB Info)
inputs.cve_browse_criteria = rule.properties.cve_browse_criteria
```



```
# Name of the vendor
inputs.cve_vendor = rule.properties.vendor
```

- CVE Search

This example sets the `cve_search_data`, `cve_search_criteria`, `cve_id`, `cve_vendor`, `cve_product`, `cve_published_date_from`, `cve_published_date_to` inputs to search selections user took on action and name of vendor , product and date range to limit the search results.

```
# value of the artifact
inputs.cve_search_data = artifact.value
# cve search criteria (may be any string Search,CVE ID,Last 30 CVES)
inputs.cve_search_criteria = rule.properties.cve_search_criteria
# Specific CVE ID
inputs.cve_id = rule.properties.cve_id
# Name of the Vendor
inputs.cve_vendor = rule.properties.vendor
# Name of the product
inputs.cve_product = rule.properties.product
# Search CVE Data from Date
inputs.cve_published_date_from = rule.properties.cve_published_date_from
# Search CVE Data upto Date
inputs.cve_published_date_to = rule.properties.cve_published_date_to
```

Post-Process Script

It can be parsed within the post-process script as `results.get("content")`. Based on the `api_call` type the data can be represented as user needs.

By default **Example: CVE Browse** function data is displayed on incident Notes, and **Example: CVE Search** function data displayed on the **CVE searched Data** Table.

- CVE Search Function

```
#globals
ENTRY_TO_DATATABLE_MAP = {
    "cve": "cve_id",
    "pubdte": "published_date",
    "sum": "summary",
    "ref": "references",
    "vc": "vulnerability_configuration",
    "vc2": "vulnerable_configuration_cpe_2_2"
}

api_call_type = results['api_call']
output_data = results['content']
api_call_type_text = "<p><b>api call type :</b> {}</p>"
browse_rich_text = "<p><b>{}&ensp;&ensp</b>{}&ensp;&ensp</p>"
rich_text_tmp = ""
```

```

#Adding data to table
ref_link_text = ""
if output_data:
    for dict_element in output_data:
        rich_text_tmp = ""
        table_row_object = incident.addRow("cve_data")
        for key_data,value_data in dict_element.items():
            if key_data == 'Published':
                table_row_object[ENTRY_TO_DATATABLE_MAP["pubdte"]] =
int(value_data)
            elif key_data == 'id':
                table_row_object[ENTRY_TO_DATATABLE_MAP["cve"]] =
value_data
            elif key_data == 'summary':
                table_row_object[ENTRY_TO_DATATABLE_MAP["sum"]] =
value_data
            elif key_data == 'references':
                for link_url in value_data:
                    ref_link_text += '<p><a href="{0}">{0}</a>
</p>'.format(link_url)
                table_row_object[ENTRY_TO_DATATABLE_MAP["ref"]] =
ref_link_text
            elif key_data == 'vulnerable_configuration':
                if value_data:
                    for vc_collection in value_data:
                        if isinstance(vc_collection,dict):
                            for key_data,value_data in
vc_collection.items():
                                text =
browse_rich_text.format(key_data,value_data)
                                rich_text_tmp += text
                            else:
                                rich_text_tmp += "<p>{}
</p>".format(vc_collection)
                        else:
                            rich_text_tmp = "No Data"
                            table_row_object[ENTRY_TO_DATATABLE_MAP["vc"]] =
rich_text_tmp
            elif key_data == 'vulnerable_configuration_cpe_2_2':
                rich_text_tmp_2 = ''
                if value_data:
                    for vc_collection in value_data:
                        rich_text_tmp_2 += "<p>{}
</p>".format(vc_collection)
                    else:
                        rich_text_tmp_2 = "No Data"
                        table_row_object[ENTRY_TO_DATATABLE_MAP["vc2"]] =
rich_text_tmp_2
                else:
                    incident.addNote("No Data Returned from CVE Search..!")

```

- CVE Browse

```

api_call_type = results['api_call']
output_data = results['content']
api_call_type_text = "<p><b>api call type :</b> {}</p>"
browse_rich_text = "<p><b>{}&ensp:&ensp</b>{}&ensp&ensp</p>"
rich_text_tmp = ""
#Adding Browse data and Database information Notes Section
api_call_type_text = api_call_type_text.format(api_call_type)
browse_rich_text_final = ""
#if api_call_type == 'browse':
if output_data:
    for x in output_data:
        for key_data,value_data in x.items():
            text = browse_rich_text.format(key_data,value_data)
            api_call_type_text += text
        browse_rich_text_final = helper.createRichText(api_call_type_text)
else:
    browse_rich_text_final = 'No Searched Data returned..!'
incident.addNote(browse_rich_text_final)

```

Rules

| Rule Name | Object Type | Workflow Triggered | Activity Fields |
|---------------------------|-------------|---------------------------|---|
| Example: CVE Browse | Incident | Example: CVE Browse | CVE Browse Criteria values : Browse, CVE DB Info, CVE Vendor |
| Example: CVE Search | Artifact | Example: CVE Search | CVE Search Criteria values : Search, Specific CVE ID, Last 30 CVEs, CVE ID, CVE Vendor, CVE Product, CVE Published Date From, CVE Published Date To |

CVE Function offers below search configurations to query vulnerabilities from DB

1. Browse :

- Select Browse and all other inputs are empty results all the vendor list from Database
- Select Browse with vendor name given returns all the products associated with the vendor

2. Search :

- Select Search with all other inputs are empty results all the vendor list from Database
- Select Search with vendor name given returns all the vulnerabilities associated with given vendor and no of results returned will be limited by given date range and **max_results_display** flag.
- Select Search with product name given returns all the vulnerabilities associated with given product and no of results returned will be limited by given date range and **max_results_display** flag.
- Select Search with vendor, product name given returns all the vulnerabilities associated with given vendor's product, and no of results returned will be limited by given date range and

`max_results_display` flag.

3. Specific CVE ID

- Select Specific CVE ID option from CVE Search Criteria with CVE ID of Vulnerability, returns data related to specific CVE ID & populates into CVE table.

4. Last 30 CVES

- Returns last 30 latest Vulnerabilities from Database no of results returned controlled by `max_results_display` flag.

5. CVE DB Info

- To get more information about the current databases in use and when it was updated

Using the CVE Function

There are two functions **Example: CVE Browse** and **Example: CVE Search**

Example CVE Browse :

- This function can be accessed on an Incident Object which offers the capabilities to **browse** for vendors and products and current CVE database information.

Example: CVE Search :

- This function can be accessed on an Artifact Object which offers the capabilities to search for product & vendor vulnerabilities, specific CVE data and the latest vulnerabilities in database.