

IBM Resilient



Incident Response Platform Integrations

QRadar Advisor Integration Function V1.0.0

Release Date: August 2018

Resilient Functions simplify development of integrations by wrapping each activity into an individual workflow component. These components can be easily installed, then used and combined in Resilient workflows. The Resilient platform sends data to the function component that performs an activity then returns the results to the workflow. The results can be acted upon by scripts, rules, and workflow decision points to dynamically orchestrate the security incident response activities.

This guide describes the QRadar Advisor Integration Function.

Overview

Backed by IBM Watson, QRadar Advisor applies artificial intelligence to automatically investigate indicators of compromise (IOC), utilizes cognitive reasoning to provide critical insights, and ultimately accelerates the response cycle. It can augment a security analyst to gain a head start in assessing incidents and reduce the risk of missing threats.

QRadar Advisor Integration Function enables Resilient users to gather Cyber Threat Intelligence(CTI) data from IBM Watson and QRadar. This information is critical for effective identification of potential IOC and quick response to incidents.

QRadar Advisor integration includes 3 example functions:

- Perform a Watson Search on an indicator and retrieve suspicious observables related to it.
- Perform a Watson Search with Local Context on an indicator and retrieve a cyber threat intelligence (CTI) report on it in Structured Threat Information eXpression ([STIX2](#)) format.
- Perform an analysis on a QRadar offense, and retrieve CTI data from QRadar Advisor and IBM Watson in STIX format.

The package also includes three workflow examples to demonstrate the usage of the above three functions.

The remainder of this document describes the above three functions and how to configure them in custom workflows or using the configuration file.

Installation

Before installing, verify that your environment meets the following prerequisites:

- Resilient platform is version 30 or later.
- You have a Resilient account to use for the integrations. This can be any account that has the permission to view and modify administrator and customization settings, and read and update incidents. You need to know the account username and password.
- You have access to the command line of the Resilient appliance, which hosts the Resilient platform; or to a separate integration server where you will deploy and run the functions code. If using a separate integration server, you must install Python version 2.7.10 or later, or version 3.6 or later, and “pip”. (The Resilient appliance is preconfigured with a suitable version of Python.)

QRadar Advisor Configuration

You need to have QRadar Advisor installed to a QRadar server, and fully configured, as shown in the following configuration page.

QRadar Advisor with Watson Administration - Mozilla Firefox: IBM Edition

https://console/plugins/1102/app_proxy/admin

IBM QRadar Advisor with Watson Configuration

- ✓ Proxy Configuration
- ✓ XFE Credentials
- ✓ License Terms
- ✓ Authorized Service Token
- ✓ Retention Policy
- ✓ Property Mapping
- ✓ Threat Intelligence Mapping
- ✓ Optimization
- ✓ Automatic Investigation
- ✓ Reference Set Export
- ✓ Complete

Proxy Configuration

The QRadar Advisor with Watson app uses the [IBM X-Force Exchange](#) to submit offenses for analysis. Enter details below, if you need to use a secure proxy.

☒ Use Proxy

☒ HTTPS ☐ SOCKS5 ☐ Disable Authentication

Proxy server

0

Proxy username

Proxy password ☐ Show Password

☐ Enable Custom SSL Certificate Validation

[Submit](#)

[XFE Credentials](#) →

To access the QRadar Advisor REST API, you need to know its `app_id`. This can be easily accessed by clicking the QRadar Advisor's Configuration icon. For example, in the URL address shown in the configuration page screenshot, the `app_id` is 1102 for this QRadar Advisor instance.

You also need an access token to use the REST API. Access tokens can be obtained from the Authorized Service Token section of the Admin page.

Install the Python components

The functions package contains Python components that are called by the Resilient platform to execute the functions during your workflows. These components run in the Resilient Circuits integration framework.

The package also includes Resilient customizations that will be imported into the platform later.

Complete the following steps to install the Python components:

1. Ensure that the environment is up-to-date, as follows:

```
sudo pip install --upgrade pip
sudo pip install --upgrade setuptools
sudo pip install --upgrade resilient-circuits
```

2. Run the following command to install the package:

```
sudo pip install --upgrade fn_qradar_advisor-<version>.<zip>
```

Configure the Python components

The Resilient Circuits components run as an unprivileged user, typically named integration. If you do not already have an integration user configured on your appliance, create it now.

Complete the following steps to configure and run the integration:

1. Using sudo, switch to the integration user, as follows:

```
sudo su - integration
```

2. Use one of the following commands to create or update the resilient-circuits configuration file. Use `-c` for new environments or `-u` for existing environments.

```
resilient-circuits config -c
```

or

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

3. Edit the resilient-circuits configuration file, as follows:

- a. In the [resilient] section, ensure that you provide all the information required to connect to the Resilient platform.
- b. In the [fn_qradar_advisor] section, edit the settings as follows:

```
qradar_host=host of your QRadar server with QRadar Advisor installed
qradar_advisor_token=qradar token (res-keyring protected recommended)
qradar_advisor_app_id=qradar app id for qradar advisor
verify_cert=[true|false] whether to validate the QRadar server cert
#optional settings
full_search_timeout=timeout for full search in seconds (1200 default)
full_search_period=period for full search in seconds (5 default)
offense_analysis_timeout=timeout for analysis in seconds (1200 default)
offense_analysis_period=period for analysis in seconds (5 default)
```

4. (Recommended) Use res-keyring to store the qradar advisor token:
 - a. Instead of storing your token in plaintext, use this instead in your app.config for the token

```
qradar_advisor_token=^qradar_advisor_token
```

- b. Now run the following command from a terminal in the same folder of your app.config

```
res-keyring
```

- c. Follow the prompt to enter your token

Deploy customizations to the Resilient platform

This package contains three function definitions and includes example workflows and rules that run these functions.

Function	Example Workflow	Rule
Watson Search	Example of Watson Search	Watson Search
Watson Search with Local Context	Example of Watson Search with Local Context	Watson Search with Local Context
QRadar Advisor Offense Analysis	Example of QRadar Advisor Offense Analysis	QRadar Advisor Offense Analysis

In addition, the package contains two custom data tables, “QRadar Advisor analysis results” and “Watson Search with Local Context results”. They are used by the example workflows to show the observables that are extracted from the QRadar Advisor STIX response.

Two demo scripts and two associated rules are also included. Each rule is a menu item added to its own data table. The user can click on a rule to create an artifact based on the selected row.

Data Table	Rule	Script
QRadar Advisor analysis results	Create Artifact (QRadar Advisor Analysis)	Create Artifact for QRadar Advisor Analysis Observable
Watson Search with Local Context results	Create Artifact (Watson Search with Local Context)	Create Artifact for Watson Search with Local Context

1. Use the following command to deploy these customizations to the Resilient platform:

```
resilient-circuits customize
```

2. Respond to the prompts to deploy functions, message destinations, workflows, scripts, and rules.

Run the integration framework

To test the integration package before running it in a production environment, you must run the integration manually with the following command:

```
resilient-circuits run
```

The resilient-circuits command starts, loads its components, and continues to run until interrupted. If it stops immediately with an error message, check your configuration values and retry.

Configure Resilient Circuits for restart

For normal operation, Resilient Circuits must run continuously. The recommended way to do this is to configure it to automatically run at startup. On a Red Hat appliance, this is done using a systemd unit file such as the one below. You may need to change the paths to your working directory and app.config.

1. The unit file must be named `resilient_circuits.service` To create the file, enter the following command:

```
sudo vi /etc/systemd/system/resilient_circuits.service
```

2. Add the following contents to the file and change as necessary:

```
[Unit]
Description=Resilient-Circuits Service
After=resilient.service
Requires=resilient.service

[Service]
Type=simple
User=integration
WorkingDirectory=/home/integration
ExecStart=/usr/local/bin/resilient-circuits run
Restart=always
TimeoutSec=10
Environment=APP_CONFIG_FILE=/home/integration/.resilient/app.config
Environment=APP_LOCK_FILE=/home/integration/.resilient/resilient_circuits.lock

[Install]
WantedBy=multi-user.target
```

3. Ensure that the service unit file is correctly permissioned, as follows:

```
sudo chmod 664 /etc/systemd/system/resilient_circuits.service
```

4. Use the `systemctl` command to manually start, stop, restart and return status on the service:

```
sudo systemctl resilient_circuits [start|stop|restart|status]
```

You can view log files for systemd and the resilient-circuits service using the `journalctl` command, as follows:

```
sudo journalctl -u resilient_circuits --since "2 hours ago"
```

Function Descriptions

Once the function package deploys the functions, you can view them in the Resilient platform Functions tab, as shown below.

The screenshot shows the 'Customization Settings' interface with the 'Functions' tab selected. The 'Functions' tab is highlighted in the top navigation bar. Below the tab bar, the 'Functions' section is displayed with a search bar and a table of functions. The table has columns for 'Name' and 'Description'. Three functions are listed: 'QRadar Advisor Offense Analysis', 'Watson Search', and 'Watson Search with Local Context'. Each function has a trash icon to its right. A 'New Function' button is located in the top right corner of the Functions section.

Name	Description
QRadar Advisor Offense Analysis	Given a Resilient artifact, this function performs a QRadar Advisor analysis and returns Local, Watson enriched, or Expanded local context (default) results.
Watson Search	Given a Resilient artifact, this function performs a Watson Search (a QRadar Advisor quick search) and returns a summary.
Watson Search with Local Context	Given a Resilient artifact, this function performs a Watson Search with Local Context (a QRadar Advisor full search) and returns Local, Watson enriched, or Expanded local context (default) results.

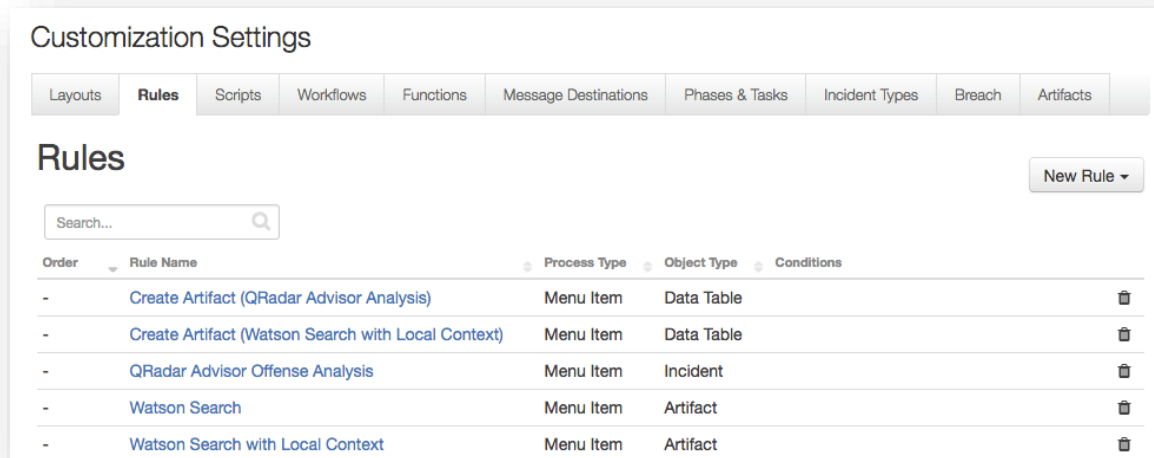
© Copyright IBM Corporation 2018

The package also includes example workflows and rules that show how the functions can be used. You can copy and modify these workflows and rules for your own needs.

The screenshot shows the 'Customization Settings' interface with the 'Workflows' tab selected. The 'Workflows' tab is highlighted in the top navigation bar. Below the tab bar, the 'Workflows' section is displayed with a search bar and a table of workflows. The table has columns for 'Workflow Name', 'Description', 'Object Type', and 'Rules'. Three workflows are listed: 'Example of QRadar Advisor Offense Analysis', 'Example of Watson Search', and 'Example of Watson Search with Local Context'. Each workflow has a trash icon to its right. A 'New Workflow' button is located in the top right corner of the Workflows section.

Workflow Name	Description	Object Type	Rules
Example of QRadar Advisor Offense Analysis	Performs an offense analysis in QRadar Advisor. Results will populate a Data Table, create a Task, and generate a STIX representation in Notes.	Incident	QRadar Advisor Offense Analysis
Example of Watson Search	Performs a Watson Search (a quick search) of observables in QRadar Advisor. Results will be displayed in incident Notes. This sample workflow also creates artifacts for suspicious observables which can be mapped to Resilient artifacts.	Artifact	Watson Search
Example of Watson Search with Local Context	Performs a Watson Search with Local Context (a full search) of observables in QRadar Advisor. Results will populate a Data Table, create a Task, and generate a HTML representation of the STIX bundles in Notes.	Artifact	Watson Search with Local Context

© Copyright IBM Corporation 2018



qradar_advisor_full_search: Watson Search with Local Context

This function calls the QRadar Advisor REST API to perform a Watson Search with Local Context on an indicator.

To use the example workflow and rule included in the package for this function, the user needs to create an incident and add an artifact. For this function to work, the artifact type must correspond to one indicator type. QRadar Advisor supports searches on the following five indicators:

- IP addresses
- Hashes
- Domains
- Urls
- Persons

QRadar Advisor supports three return stages:

- Stage1: feature hunt
- Stage2: cognitive investigation added on top of result of stage 1
- Stage3: wider feature hunt added on top of result of stage 2

The user can specify the desired return stage in the pre-process script of the example workflow.

The screenshot shows the 'Customization Settings' window for a workflow. The 'Workflows' tab is selected. The workflow name is 'Example of Watson Search with Local Context'. The API Name is 'qradar_advisor_full_search'. The Description is 'Performs a Watson Search with Local Context (a full search) of observables in QRadar Advisor. Results will populate a Data Table, create a Task, and generate a HTML'. The Object Type is 'Artifact'. The Creator is 'masterfirst', Last Modified is '08/01/2018 11:25', Last Modified By is 'masterlast', and the Associated Rule is 'Watson Search with Local Context'.

The workflow diagram shows an input circle connected to a function block labeled 'Watson Search with Local Context'. The function block has two output arrows: one labeled 'qradar_advisor_search_value' and 'qradar_advisor_result_stage' pointing to a circle, and another labeled 'results.note', 'results.summary', and 'results.stix' pointing to a circle.

The 'Input Parameter' table is shown below the diagram:

Input Parameter	Value
qradar_advisor_search_value	
qradar_advisor_result_stage	stage1 stage2 stage3

The search REST API of QRadar Advisor returns CTI information in Structured Threat Information Expression (STIX 2.0) format. It is normally a STIX bundle with STIX objects. The function processes the STIX data and performs the following:

- Generates a HTML representation of the STIX data
- Extracts observables from the STIX objects
- Generates a summary from the STIX data

The return data from this function includes the raw STIX data in json dictionary format.

In the post-process script, the HTML representation is used to create a note. The observables are used to populate the custom data table, “Watson Search with Local Context results”, and the summary is used to create a task. Note that the raw STIX data is accessible from the post-process script as results.stix, and can be parsed to create custom code.

Customization Settings

LayoutsRulesScripts**Workflows**FunctionsMessage DestinationsPhases & TasksIncident TypesBreachArtifacts

Workflows / Example of Watson Search with Local Context

CancelSave & CloseSave

Name *

Example of Watson Search with Local Context

API Name *

qradar_advisor_full_search

Description

Performs a Watson Search with Local Context (a full search) of observables in QRadar Advisor. Results will populate a Data Table, create a Task, and generate a HTML

Object Type *

Artifact

Creator

masterfirst

Last Modified

08/01/2018 11:25

Last Modified By

masterfirst

Associated Rules

Watson Search with Local Con

Input: qradar_advisor_search_value
qradar_advisor_result_stage

Output: results.observables
results.note
results.summary
results.stix

Watson Search with Local

Input

Pre-Process Script

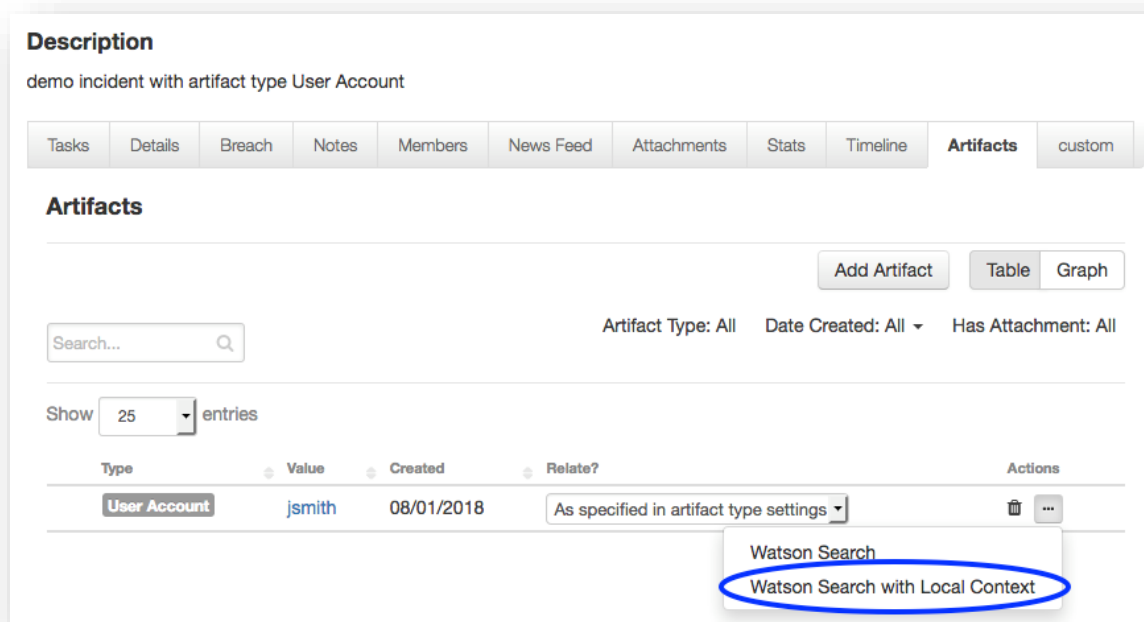
Output

Post-Process Script

Language: Python Theme light Mode Default Tab Size 2 - Font + Font

```
10 # We publish a data table according to the stix
11 date_str = str(Date())
12 for observable in results.observables:
13     qradar_obs = incident.addRow("qradar_advisor_observable_for_artifact")
14     qradar_obs.qradar_advisor_toxicity = observable.toxicity
15     qradar_obs.qradar_advisor_relevance = observable.relevance
16     qradar_obs.qradar_advisor_type = observable.type
17     qradar_obs.qradar_advisor_description = observable.description
18     qradar_obs.artifact_related = artifact.value
19     qradar_obs.full_search_time = date_str
20 # Our STIX tree
21 html = helper.createRichText(results.note)
22 incident.addNote(html)
```

In the following example, a User Account artifact was added to an incident with value “jsmith”. The user can then select Watson Search with Local Context from the artifact menu to search QRadar Advisor for the observable.



Please note that both Watson Search and Watson Search with Local Context perform queries for information about an indicator. Therefore, only those artifacts that can be mapped into indicators are supported. The types of artifacts that can be searched include:

- DNS Name
- Malware SHA-256 Hash
- Malware SHA-1 Hash
- Malware MD5 Hash
- IP Address
- URL
- User Account

The menu items for Watson Search and Watson Search with Local Context are only shown for the artifact types listed above.

Note that a full search like this could take up to 15 minutes. Once it is completed, the note created for this indicator can be viewed from the Notes tab of this incident.

Description
demo incident with artifact type User Account

TasksDetailsBreach**Notes**MembersNews FeedAttachmentsStatsTimelineArtifactscustom

Notes

Sans SerifNormalB I U W

PostCancel

Search...

☒ Show Task Notes ☐ Oldest Notes FirstCreated By: AllDate Created: All

masterfirst masterlast added a note to the Incident 08/01/2018 13:22

- jsmith
 - 10.103.22.35
 - file with name unknown
 - 1ea41812a0114e5c6ae76330e7b4af69
 - gmuweb.exe
 - Backdoor.Darkmoon
 - file with name unknown
 - Packed.Generic.347
 - ab.exe
 - 7b0cb4d14d3d8b6ccc7453f7ddb33997
 - 10.103.22.50
 - file with name unknown
 - 765f46vb.exe
 - fd4da1b404961e6db45469a27a201f41
 - Trojan.Gen
 - file with name unknown
 - JS.Downloader
 - d565937b49df96e6a8b88fedcf15d82a
 - 9758W-TERBEDOC-RS62937-15000.zip

Please note that the icons shown in the above note use external URL referencing to the official site for STIX2 icons (<https://raw.githubusercontent.com/freetaxii>). Therefore, those icons are shown only if the Resilient platform can access the above website.

Also note that some indicators have a link icon at the end. These indicators are basically placeholders for the other (real) indicators with the same value. Think of them as symbolic links in a folder tree.

The data table can be viewed if the user adds the “Watson Search with Local Context results” data table into one tab of an incident. Note that this package includes a rule, “Create Artifact (Watson Search with Local Context)”, which is added to the “Watson Search with Local Context results” data table. This enables the user to create an artifact based on a selected row from this data table as shown below.

Artifact Searched ⓘ	Search Time ⓘ	Description ⓘ	Type ⓘ	Toxicity ⓘ	Relevance ⓘ	
jsmith	Wed Aug 01 17:22:14 UTC 2018	1ea41812a0114e5c6ae76330e7b4af69	file	very-low	medium	...
jsmith	Wed Aug 01 17:22:14 UTC 2018	10.103.22.35	ipv4-addr	very-low	medium	...
jsmith	Wed Aug 01 17:22:14 UTC 2018	Trojan.Gen	mal			...
jsmith	Wed Aug 01 17:22:14 UTC 2018	JS.Downloader	malware	very-high	medium	...
jsmith	Wed Aug 01 17:22:14 UTC 2018	a26fb483371ba5b77b5be7b3f8c74cf4	file	very-low	medium	...
jsmith	Wed Aug 01 17:22:14 UTC 2018	765f46vb.exe	file	very-low	medium	...

The newly created task can be viewed from the Tasks tab.

Description
 No description.

Tasks

Details

Breach

Notes

Members

News Feed

Attachments

Stats

Timeline

Artifacts

QRadar

Splunk

Tasks

0% Complete

Filter: All

Selected

Add Task

Task Name	Owner	Due Date	Flags	Actions
Initial				
Review Watson full search of artifact: jsmith	Unassigned	No due date	0 0	
Respond				
Respond - (Data Breach - Organizational)				
Investigate exposure of PI	Unassigned	No due date	0 0	

Since Watson Search with Local Context could potentially take a long time to complete depending on the performance of QRadar Advisor, additional configuration settings are available in the app.config file.

Setting	Explanation
full_search_timeout	Timeout in seconds. It is the time the function waits for the result returned from QRadar Advisor. It is optional, and defaulted to 1200 seconds if absent.
full_search_period	In seconds. It specifies how often the function checks the search status. It is optional, and defaulted to 5 seconds if absent.

qradar_advisor_quick_search: Watson Search

Watson Search calls the QRadar Advisor REST API to perform a quick search on an indicator.

To use the example workflow for this function, the user creates an incident and then adds an artifact with the desired artifact type as shown in the Watson Search with Local Context function.

The QRadar Advisor REST API for Watson Search returns data in json format. The json dictionary contains two lists, one for `suspicious_observables`, and the other for `other_observables`. In the post-process script of this example workflow, the `suspicious_observables` are mapped to default artifact types, using a dictionary defined there. The user can easily map observables to custom artifacts by modifying the dictionary mapping.

Note the `other_observables` are not used in this example workflow. If user wants to make use of them, they can be accessed in the post-process script as `results.other_observables`.

In the following example, a Watson Search on the artifact, "jsmith", is initiated when selecting Watson Search from the artifact menu.

Description
No description.

Tasks Details Breach Notes Members News Feed Attachments Stats Timeline **Artifacts** custom

Artifacts

Add Artifact Table Graph

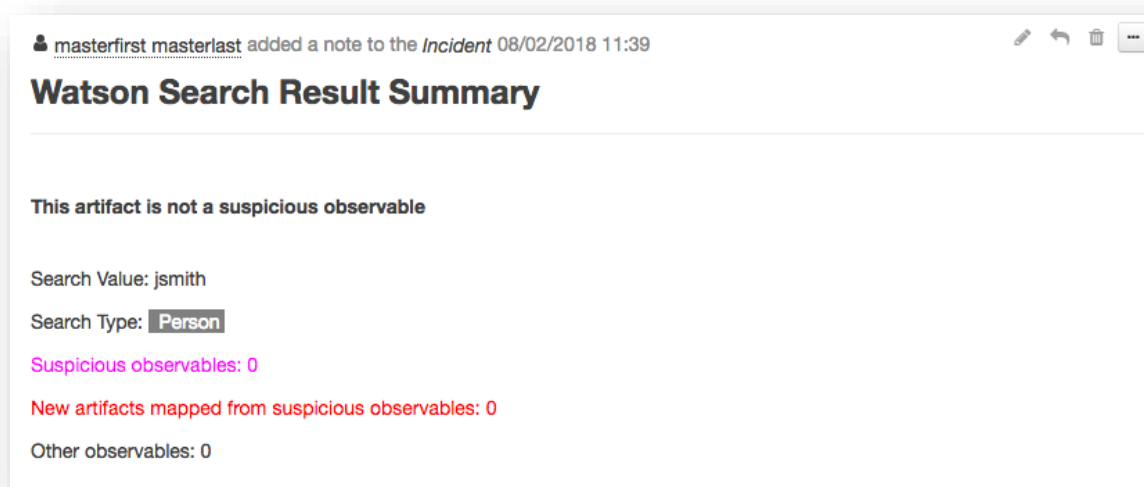
Search... Artifact Type: All Date Created: All Has Attachment: All

Show 25 entries

Type	Value	Created	Relate?	Actions
IP Address	37.201.193.196	08/02/2018	As specified in artifact type settings	...
User Account	jsmith	08/02/2018	As specified in artifact type settings	...

Watson Search
Watson Search with Local Context

For this example, the Watson Search search of “jsmith” does not return any suspicious observables. As a result, no new artifacts are added. A note was added to the incident to summarize this.



qradar_advisor_offense_analysis: QRadar Advisor Offense Analysis

This function calls the QRadar Advisor REST API to:

- get the insights of a QRadar Advisor offense.
- perform analysis of the offense.

The QRadar Advisor return of insights is in json format, and the result of an analysis is in STIX format.

Similar to the Watson Search with Local Context, this function generates a HTML representation of the STIX data. It also extracts observables from the STIX objects.

Just like Watson Search with Local Context, the user can also specify the return stage from the pre-process script of the example workflow.

Input	Pre-Process Script	Output	Post-Process Script								
<table><thead><tr><th>Input Parameter</th><th>Value</th></tr></thead><tbody><tr><td>qradar_offense_id</td><td><input type="text"/></td></tr><tr><td>qradar_advisor_result_stage</td><td>stage3</td></tr><tr><td>qradar_analysis_restart_if_existed ⓘ</td><td>Yes</td></tr></tbody></table>				Input Parameter	Value	qradar_offense_id	<input type="text"/>	qradar_advisor_result_stage	stage3	qradar_analysis_restart_if_existed ⓘ	Yes
Input Parameter	Value										
qradar_offense_id	<input type="text"/>										
qradar_advisor_result_stage	stage3										
qradar_analysis_restart_if_existed ⓘ	Yes										

One more setting is qradar_analysis_restart_if_existed. If this flag is set to Yes, the function restarts a new analysis even if a previous result exists for this offense.

In the post-process script, the HTML representation is used to create a note. The observables are used to populate the “QRadar Advisor analysis results” data table. The insights are used to create a task.

The screenshot shows the 'Customization Settings' window for a workflow. The 'Workflows' tab is selected. The workflow name is 'Example of QRadar Advisor Offense Analysis', the API name is 'qradar_advisor_offense_analysis', and the object type is 'Incident'. The description states: 'Performs an offense analysis in QRadar Advisor. Results will populate a Data Table, create a Task, and generate a STIX representation in Notes.' The creator is 'masterfirst' and the last modified date is '08/01/2018 11:25'. The associated rule is 'QRadar Advisor Offense'.

The workflow diagram shows a process flow starting with an input, followed by a 'Pre-Process Script' step, then a 'Post-Process Script' step. The 'Post-Process Script' step is highlighted, and its script content is displayed below the diagram. The script is in Python and includes comments and code for publishing data, creating a note, and creating a task.

```
10 # We publish a data table according to the stix
11 for observable in results.observables:
12     qradar_obs = incident.addRow("qradar_advisor_observable")
13     qradar_obs.qradar_advisor_toxicity = observable.toxicity
14     qradar_obs.qradar_advisor_relevance = observable.relevance
15     qradar_obs.qradar_advisor_type = observable.type
16     qradar_obs.qradar_advisor_description = observable.description
17 # Our STIX tree
18 html = helper.createRichText(results.note)
19 incident.addNote(html)
20
21 # Task
22 task_title = "Review QRadar Advisor Analysis for Offense " + str(incident.properties.qradar_id)
23 task_summary = results.insights.insights + "\n\n" + results.insights.stage3_insights
24 incident.addTask(task_title, "Initial", task_summary)
```

Note the raw STIX data from QRadar Advisor is accessible from the post-process script as `results.stix`, if the customer wants to create custom code to parse the STIX data.

To use the example workflow, a Resilient incident must have a valid QRadar offense ID stored in the `qradar_id` field. In the following example, the incident is linked to QRadar offense 27.

Description
demo incident with Offense Analysis

Tasks	Details	Breach	Notes	Members	News Feed	Attachments	Stats	Timeline
Artifacts	custom							

Details Edit

Basic Details

Name ⓘ	offense analysis 27
qradar_id	27
Description ⓘ	demo incident with Offense Analysis
Incident Type ⓘ	—

The offense analysis begins upon selection of the rule “QRadar Advisor Offense Analysis” from the Action menu of the incident.

offense analysis 27 Actions ▾

Summary

ID 2113

Phase Respond

Severity Low

Date Created 08/01/2018

Date Occurred —

Date Discovered 08/01/2018

Was personal information or personal data involved? Unknown

Incident Type —

Description
demo incident with Offense Analysis

Tasks	Details	Breach	Notes	Members	News Feed
Artifacts	custom				

Details Edit

Basic Details

Name ⓘ	offense analysis 27
qradar_id	27
Description ⓘ	demo incident with Offense Analysis

QRadar Advisor Offense Analysis

Action Status

Workflow Status

Close Incident

Delete Incident

A normal analysis can take up to 20 minutes. Once completed, the HTML representation is shown in the Notes tab.

Description
demo incident with Offense Analysis

Tasks

Details

Breach

Notes

Members

News Feed

Attachments








Stats

Timeline

Artifacts


custom

Notes

Sans Serif ▾ Normal ▾ B I U     A    W▾






Post











Cancel

Search... 

☐ Show Task Notes ☐ Oldest Notes First

Created By: All Date Created: All ▾

 masterfirst masterlast added a note to the *Incident* 08/02/2018 12:08    

-  bare.ru
 -  50.63.202.5
 -  50.63.202.20
 -  184.168.221.3
 -  50.63.202.34
 -  184.168.221.27
 -  184.168.221.14
 -  184.168.221.11
 -  50.63.202.19
 -  50.63.202.31

Observables are added to the data table, "QRadar Advisor analysis results". A menu rule is included for this data table. Users can use it to create a new artifact based on the selected row.

Description
demo incident with Offense Analysis

TasksDetailsBreachNotesMembersNews FeedAttachmentsStatsTimelineArtifacts**custom**

custom

Edit

QRadar Advisor analysis results

Search...

Q

Print

Export

Description ⓘ	Type ⓘ	Toxicity ⓘ	Relevance ⓘ	
50.63.202.5	ipv4-addr	very-high	low	...
scorpena.com	domain	very-high	medium	...
50.63.202.17	ipv4-addr	very-high		
http://eudactica.com/765f46vb.exe	url	very-high	medium	...
spam zero-day	malware	very-high	medium	...

Create Artifact (QRadar Advisor Analysis)

A task is created based on the insights returned from QRadar Advisor. The insights are kept in the instruction of the task.

Description
demo incident with Offense Analysis

TasksDetailsBreachNotesMembersNews FeedAttachmentsStatsTimelineArtifacts**custom**

Tasks

0% Complete

Task Name

Initial

Review QRadar Advisor Analysis for Offense 27

Respond

Respond - (Data Breach - Organization)

* Investigate Exposure of Personal Information/Data

Instructions

The Watson analysis of 18 observables from this offense has finished. The analysis found 134 new indicators that were not included in the offense. A total of forty data points were found to be linked with the offense. 41 indicators were related to suspicious activity, and five indicators were active. From the newly found indicators, 36 have ties to suspicious activity. In particular,

Filter: All

Selected

Add Task

Flags

Actions

0

0

0

0

Since an analysis could potentially take a long time to complete depending on the performance of QRadar Advisor, additional configuration settings are available in the app.config file.

Setting	Explanation
offense_analysis_timeout	Timeout in seconds. It is the time the function waits for the result returned from QRadar Advisor. It is optional, and defaulted to 1200 seconds if absent.
offense_analysis_period	In seconds. It specifies how often the function checks the analysis status. It is optional, and defaulted to 5 seconds if absent.

Troubleshooting

There are several ways to verify the successful operation of a function.

- Resilient Action Status

When viewing an incident, use the Actions menu to view Action Status. By default, pending and errors are displayed. Modify the filter for actions to also show Completed actions. Clicking on an action displays additional information on the progress made or what error occurred.

- Resilient Scripting Log

A separate log file is available to review scripting errors. This is useful when issues occur in the pre-processing or post-processing scripts. The default location for this log file is:
`/var/log/resilient-scripting/resilient-scripting.log`.

- Resilient Logs

By default, Resilient logs are retained at `/usr/share/co3/logs`. The `client.log` may contain additional information regarding the execution of functions.

- Resilient-Circuits

The log is controlled in the `.resilient/app.config` file under the section `[resilient]` and the property `logdir`. The default file name is `app.log`. Each function will create progress information. Failures will show up as errors and may contain python trace statements.

Support

For additional support, contact support@resilientsystems.com.

Including relevant information from the log files will help us resolve your issue.