



# 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, 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 Indicator of Compromise(IOC) and quick response to incidents.

QRadar Advisor integration includes 3 functions:

* Perform a quick search on an indicator, and retrieve suspicious observables related to it
* Perform a full search on an indicator, and retrieve cyber threat intelligence (CTI) report on it
* Do an analysis on a QRadar offense, and retrieve CTI data from QRadar Advisor and IBM Watson

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.

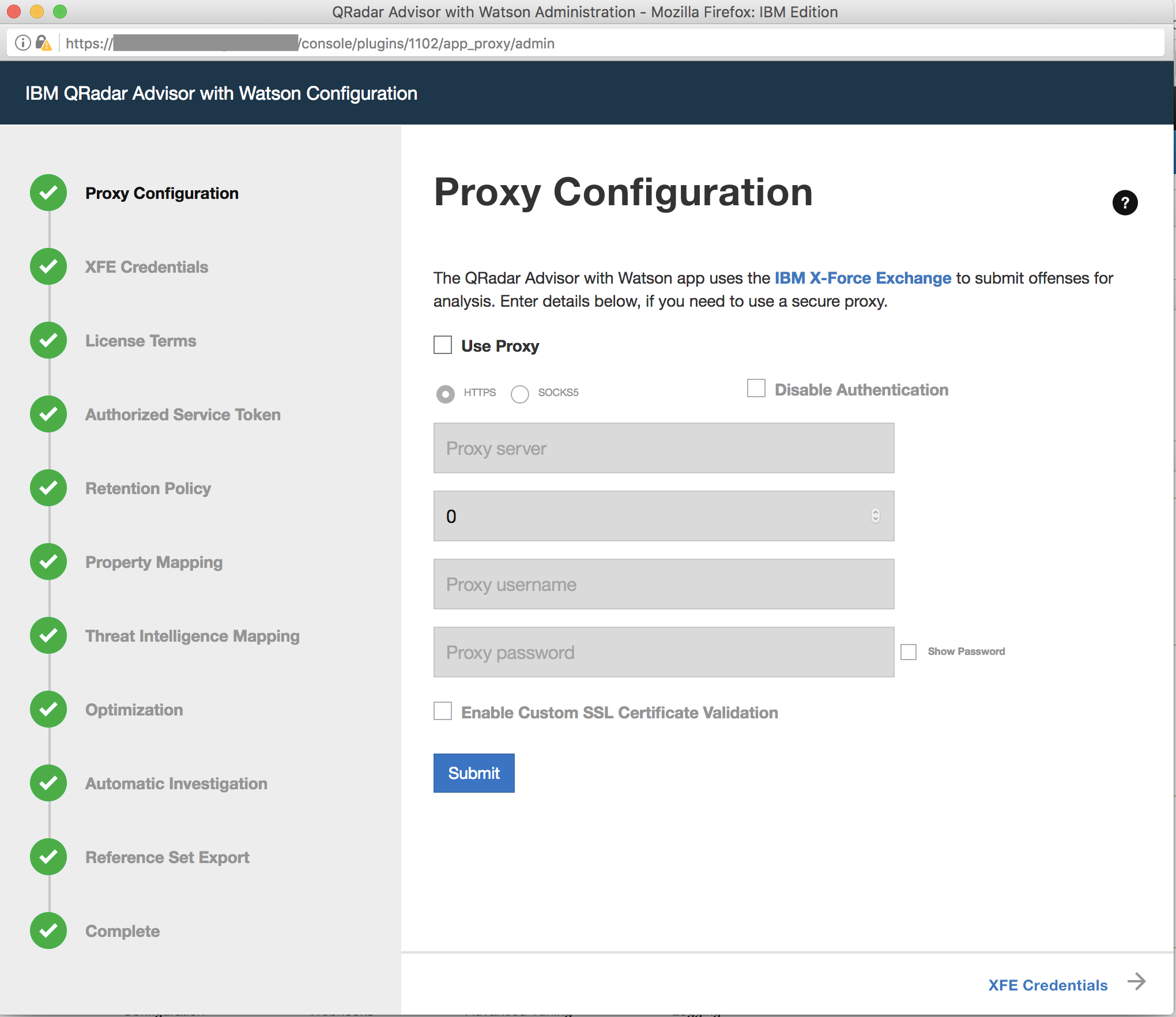
Installation

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.



To access the QRadar Advisor REST API, you need to know the app\_id of it. This can be easily figured out by clicking the QRadar Advisor configure page. For example, the app\_id for QRadar Advisor is shown to be 1102 for this QRadar instance in the url address.

You also need an access token to use the REST API. Access tokens can be found in the Authorized Services from 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

1. 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

1. 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

1. Edit the resilient-circuits configuration file, as follows:
   1. In the [resilient] section, ensure that you provide all the information required to connect to the Resilient platform.
   2. 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 (key-ring 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)

Deploy customizations to the Resilient platform

This package contains function definitions that you can use in workflows, and includes example workflows and rules that show how to use these functions.

|  |  |  |
| --- | --- | --- |
| **Function** | **Example Workflow** | **Rule** |
| QRadar Advisor Quick Search | Example of QRadar Advisor Quick Search | QRadar Advisor Quick Search |
| QRadar Advisor Full Search | Example of QRadar Advisor Full Search | QRadar Advisor Full Search |
| QRadar Advisor Offense Analysis | Example of QRadar Advisor Offense Analysis | QRadar Advisor Offense Analysis |

In addition, the package also contains a custom data table “QRadar Advisor Observable”. It is used by the above example workflows to show the observables extract from the stix reply from QRadar Advisor.

A demo script called “Create Artifact for QRadar Advisor Observable” is included as well. It is a menu item added to the custom data table above. User can click it to add a row as an artifact to the incident.

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

resilient-circuits customize

1. Respond to the prompts to deploy functions, message destinations, workflows 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 recommend 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

1. 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

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

sudo chmod 664 /etc/systemd/system/resilient\_circuits.service

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

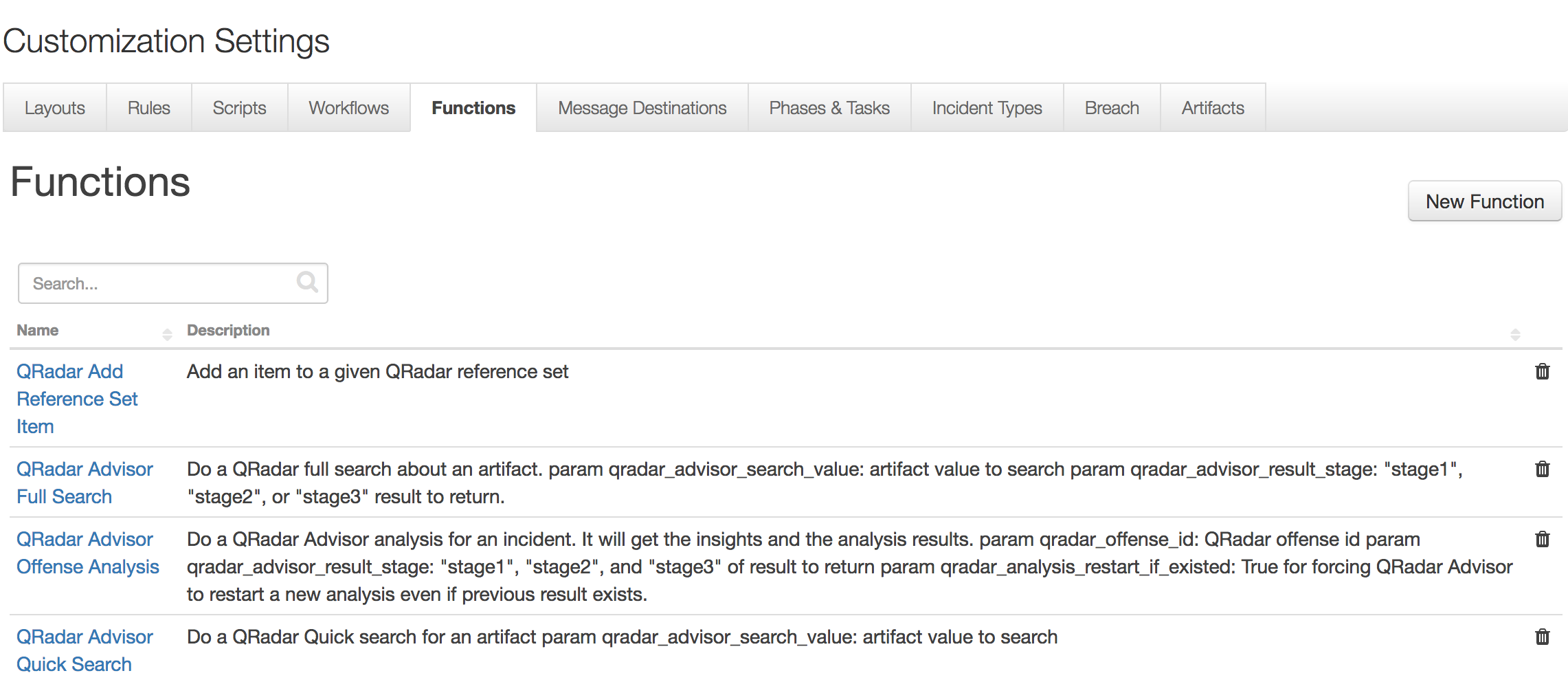
sudo systemctl resilient\_circuits [start|stop|restart|status]

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

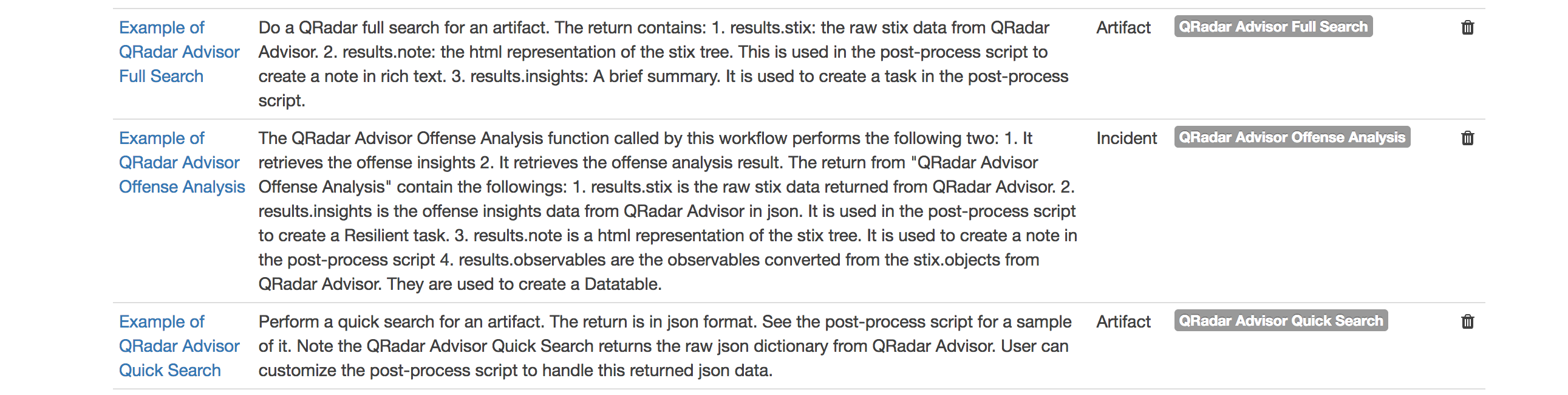
sudo journalctl -u resilient\_circuits --since "2 hours ago"

Function Descriptions

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



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.



qradar\_advisor\_full\_search: QRadar Advisor Full Search

This function calls the QRadar Advisor REST API to perform a full search on an indicator.

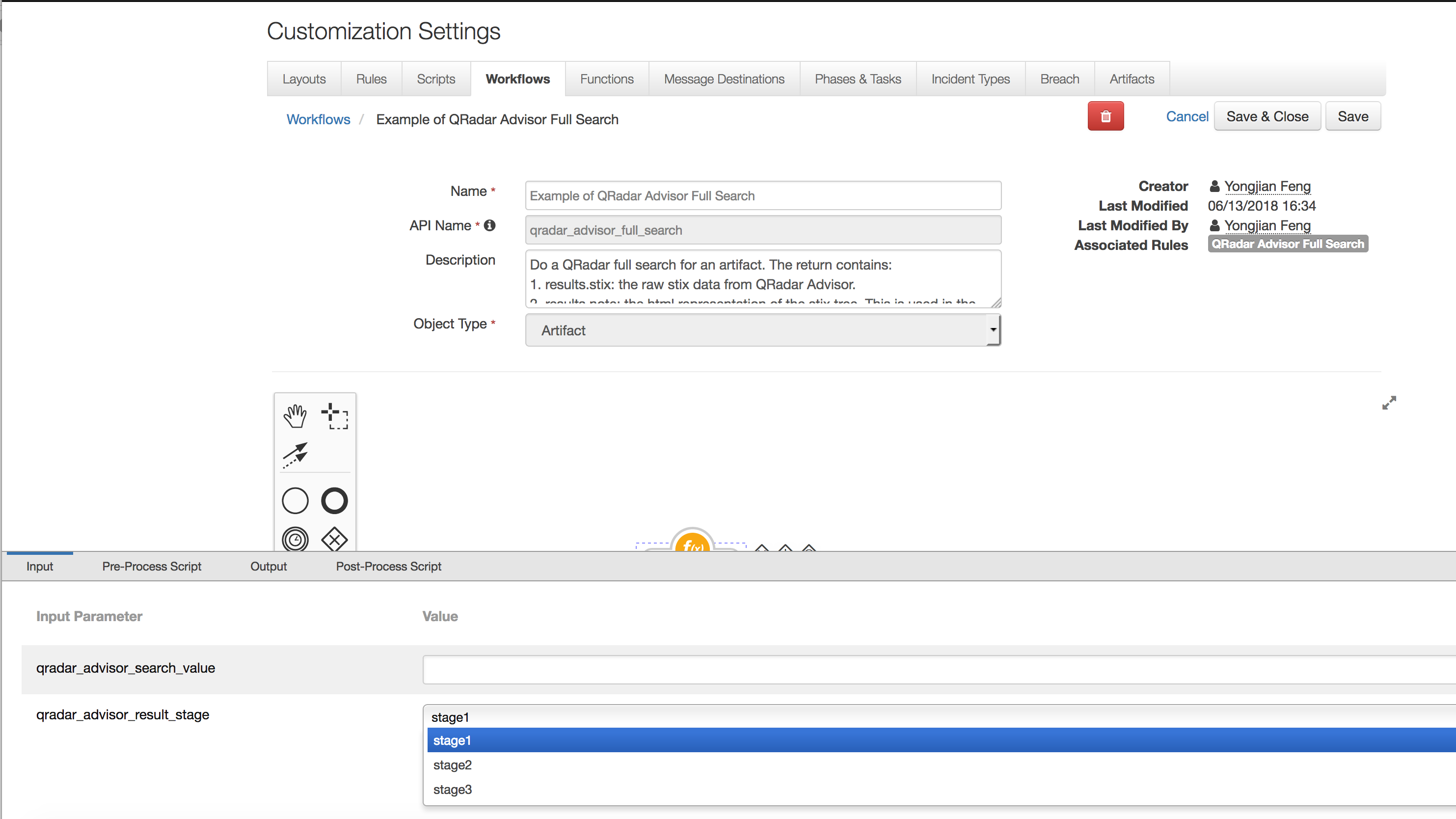
To use the example workflow and rule included in the package for this function, the user needs to create an incident and add an artifact. For this function to work, the artifact type must correspond to one indicator type. Note that currently QRadar Advisor supports full search on the following five indicators:

* IP address
* Hash
* DomainName
* Url
* Person

QRadar Advisor supports 3 return stages:

* Stage1: feature hunt
* Stage2: cognitive investigation
* Stage3: wider feature hunt

User can specify which stage of result is wanted by setting the input in the pre-process script of the example workflow.

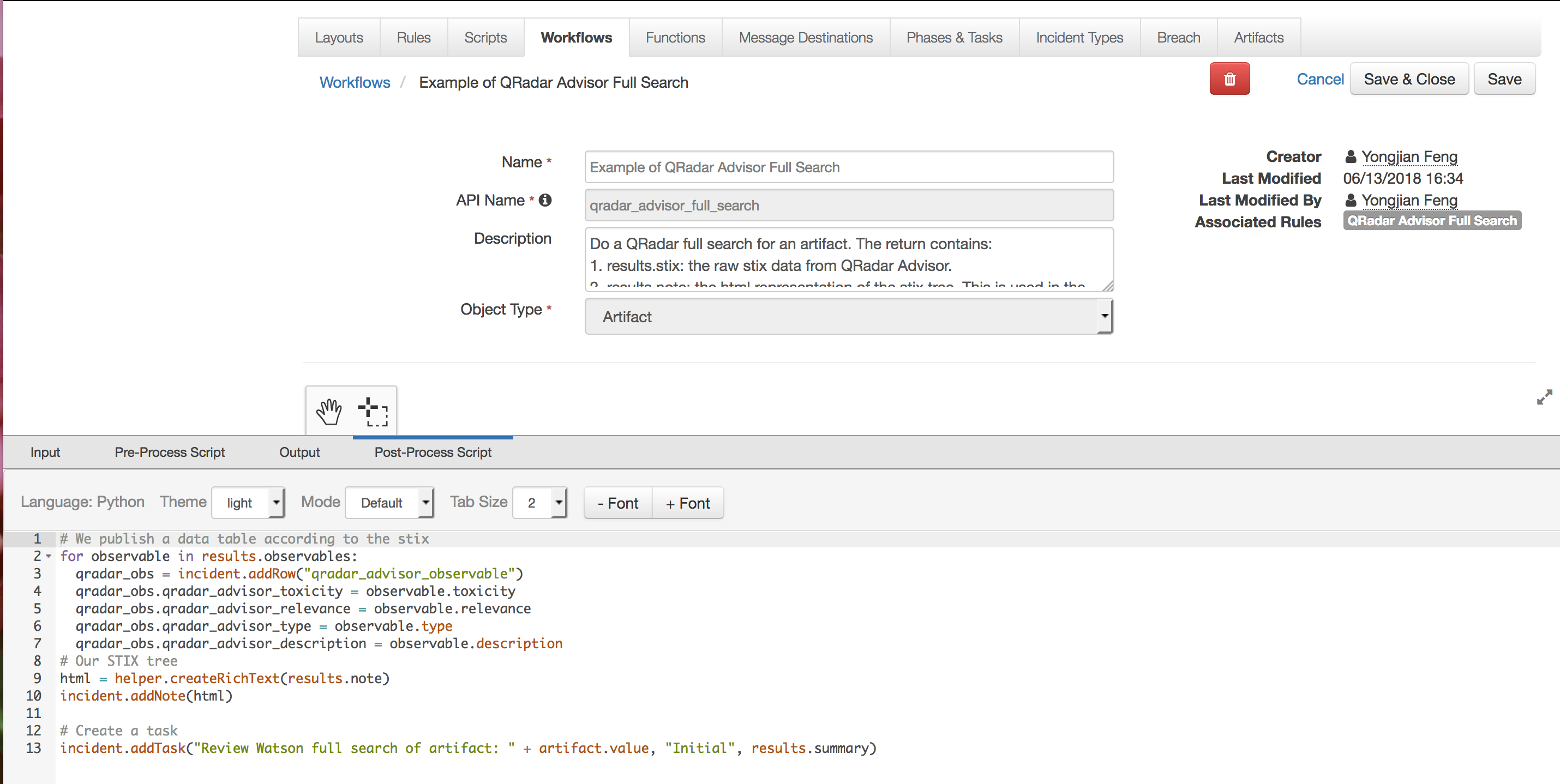


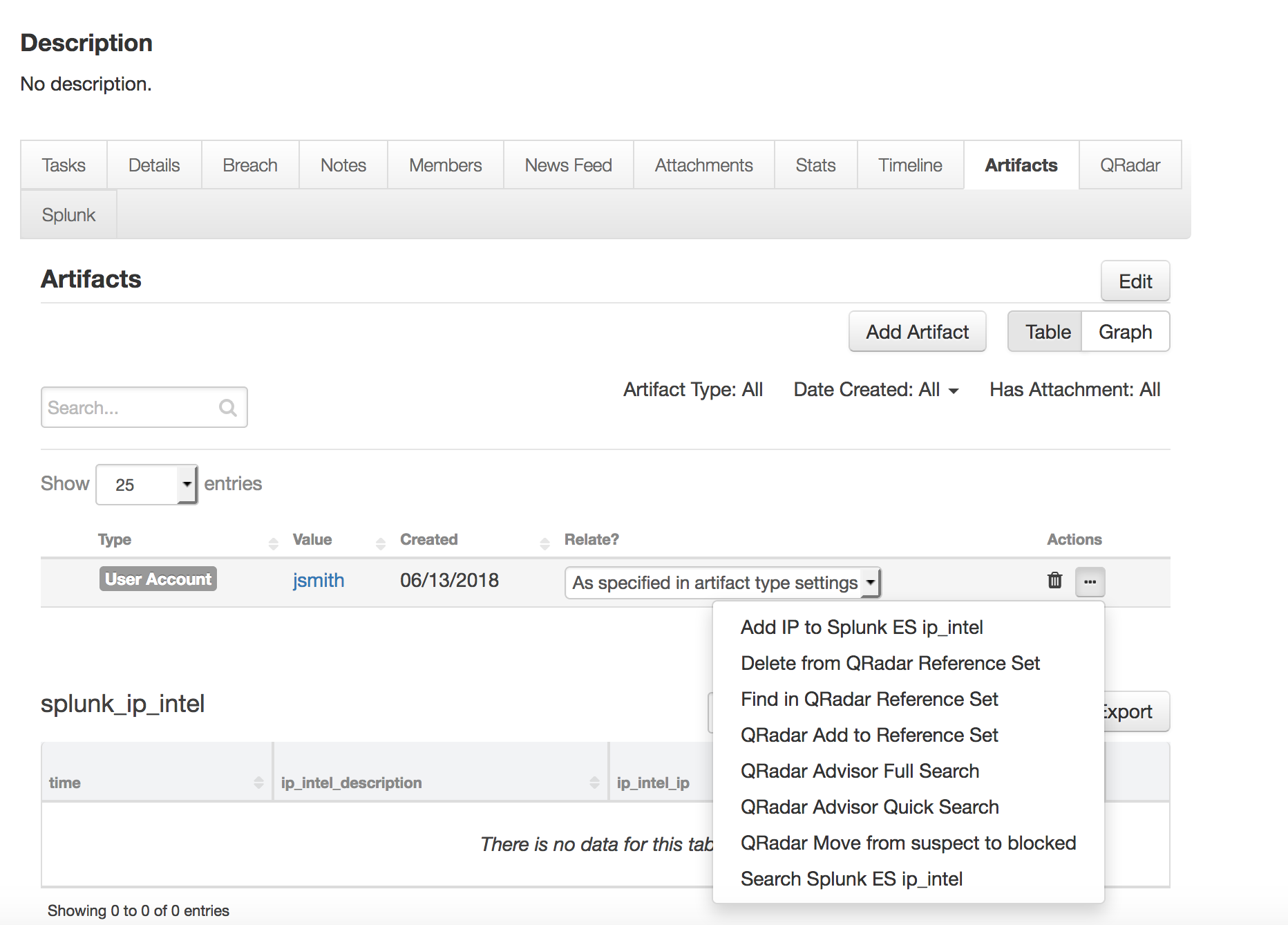
The full search REST API of QRadar Advisor returns CTI information in Structured Threat Information Expression (STIX 2.0) format. It is normally a STIX bundle with STIX objects. This function processes the STIX data and

* Generate a HTML representation of the STIX data
* Extract observables from the STIX objects
* Generate a summary from the STIX data

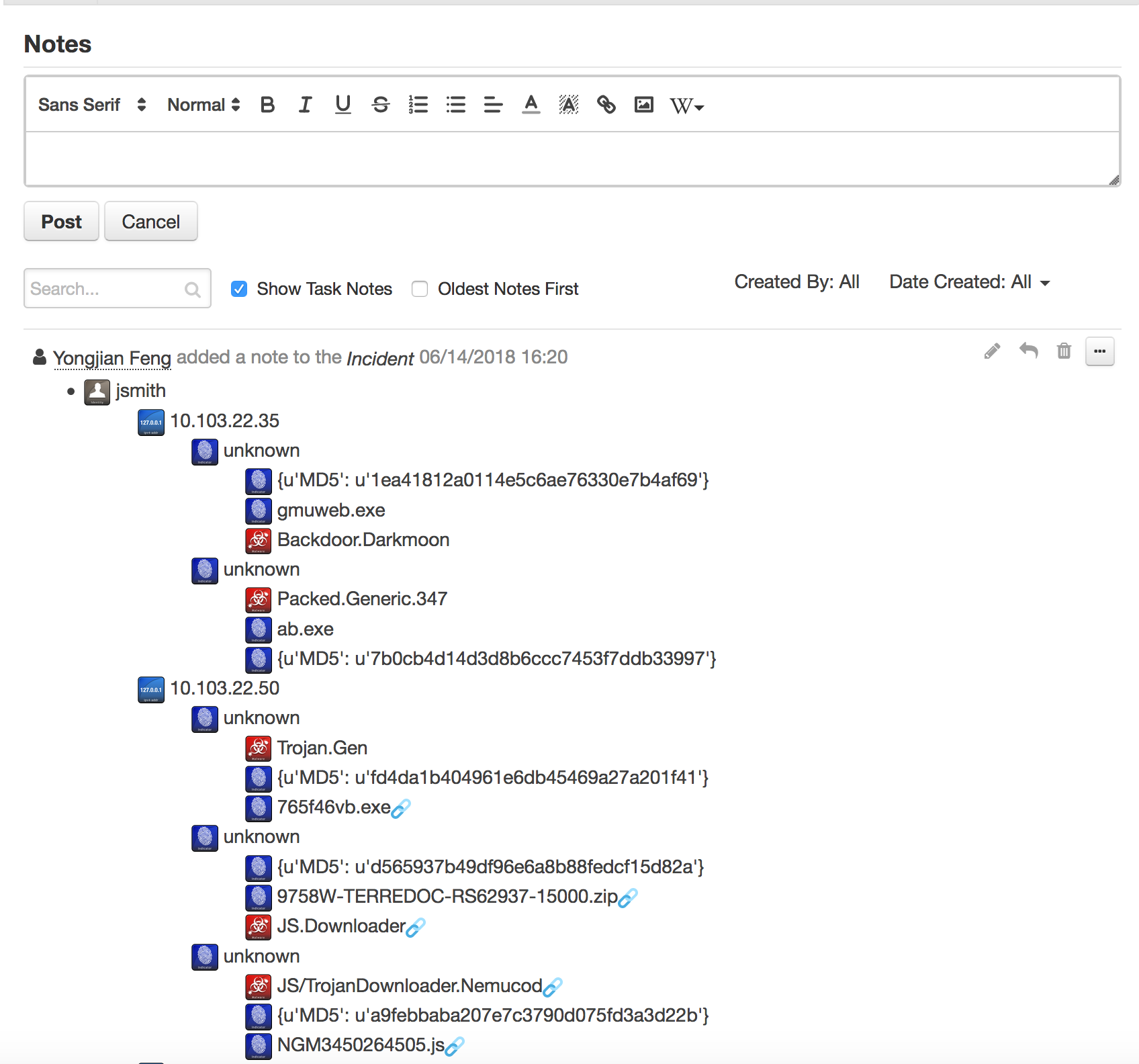
The return from this function includes the above three, together with the raw STIX data in json dictionary.

In the post-process script, the HTML representation is used to create a note. The observables are used to polish a custom data table called “QRadar Advisor Observable”. And the summary is used to create a task. 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.

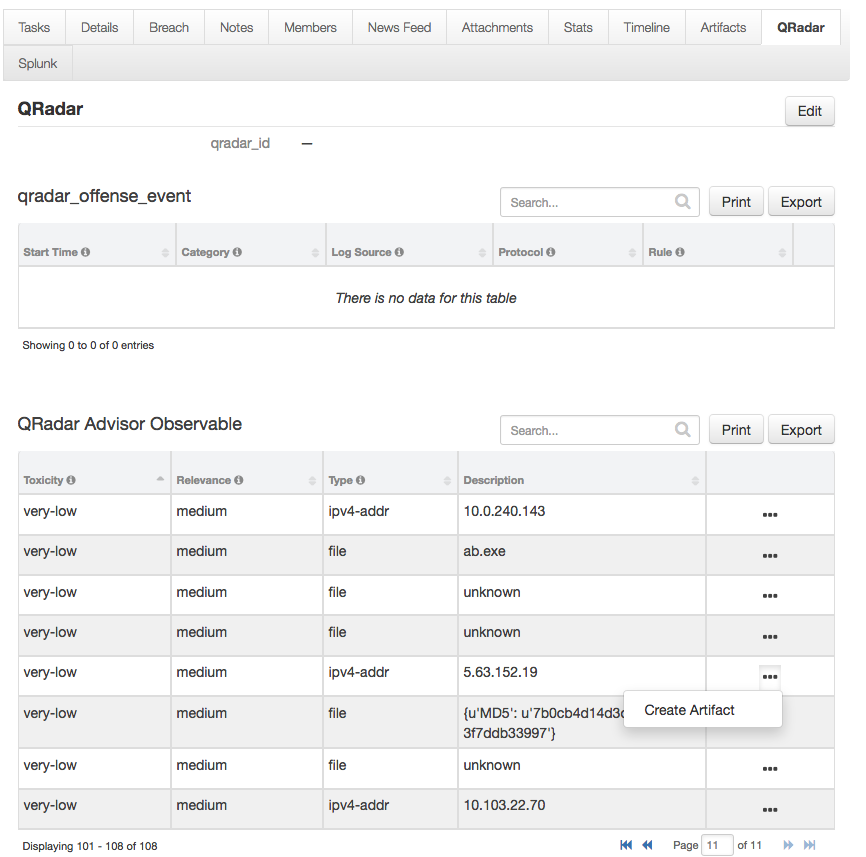


In the following example, a User Account artifact was added to an incident with value “jsmith”. User can then select QRadar Full Search from the artifact menu. 

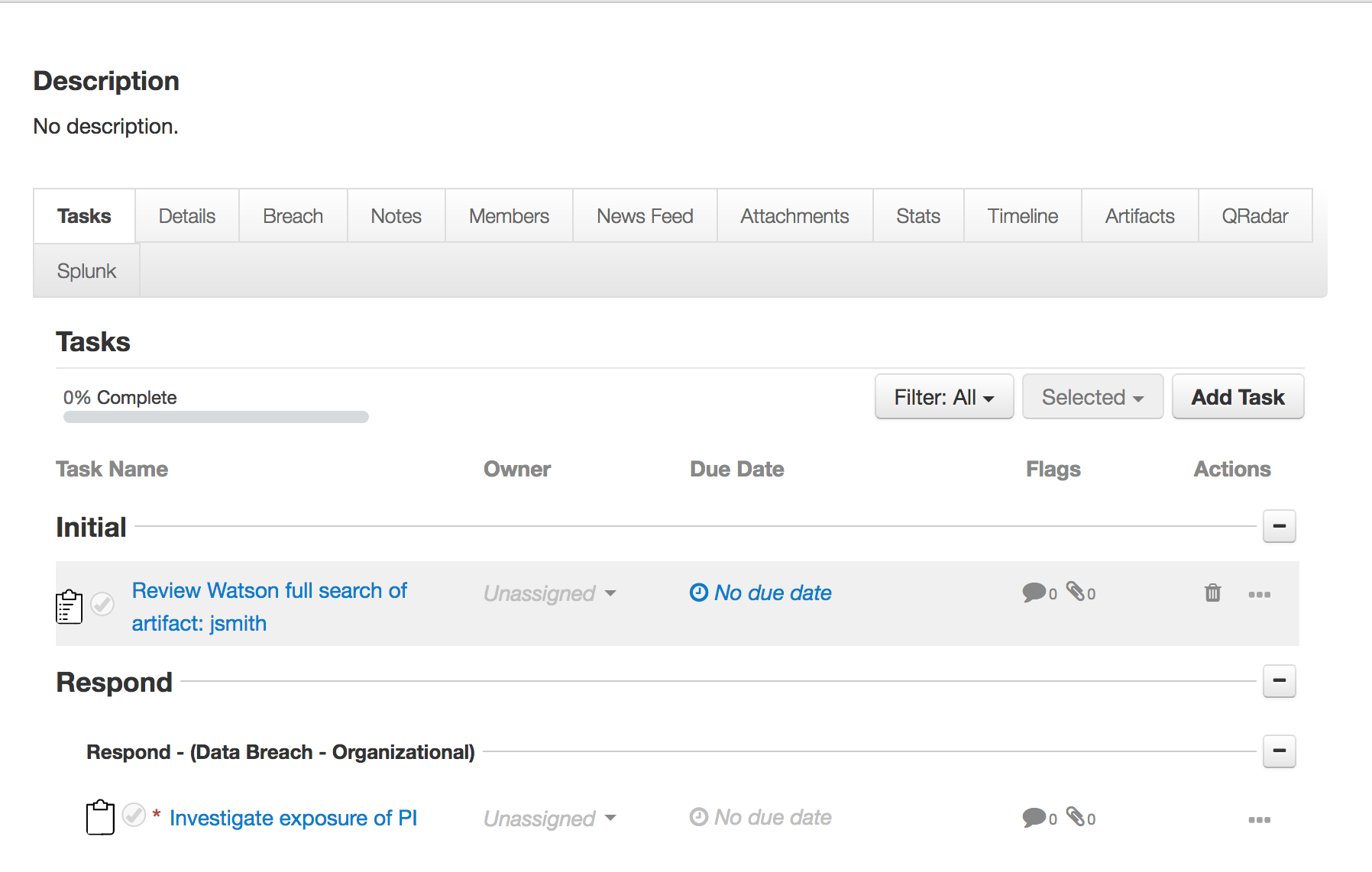
Note that a full search like this could take up to 15 minutes. Once it is completed, the note created for this indicator can be viewed from the Note tab of this incident.



The data table can be viewed if user adds the “QRadar Advisor Observable” data table into one tab of an incident. Note that this package includes a rule “Create Artifact”, which is added to the “QRadar Advisor Observable” data table. User can create an artifact based on a selected row from this datatable.



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



Since a full search 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: QRadar Advisor Quick Search

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

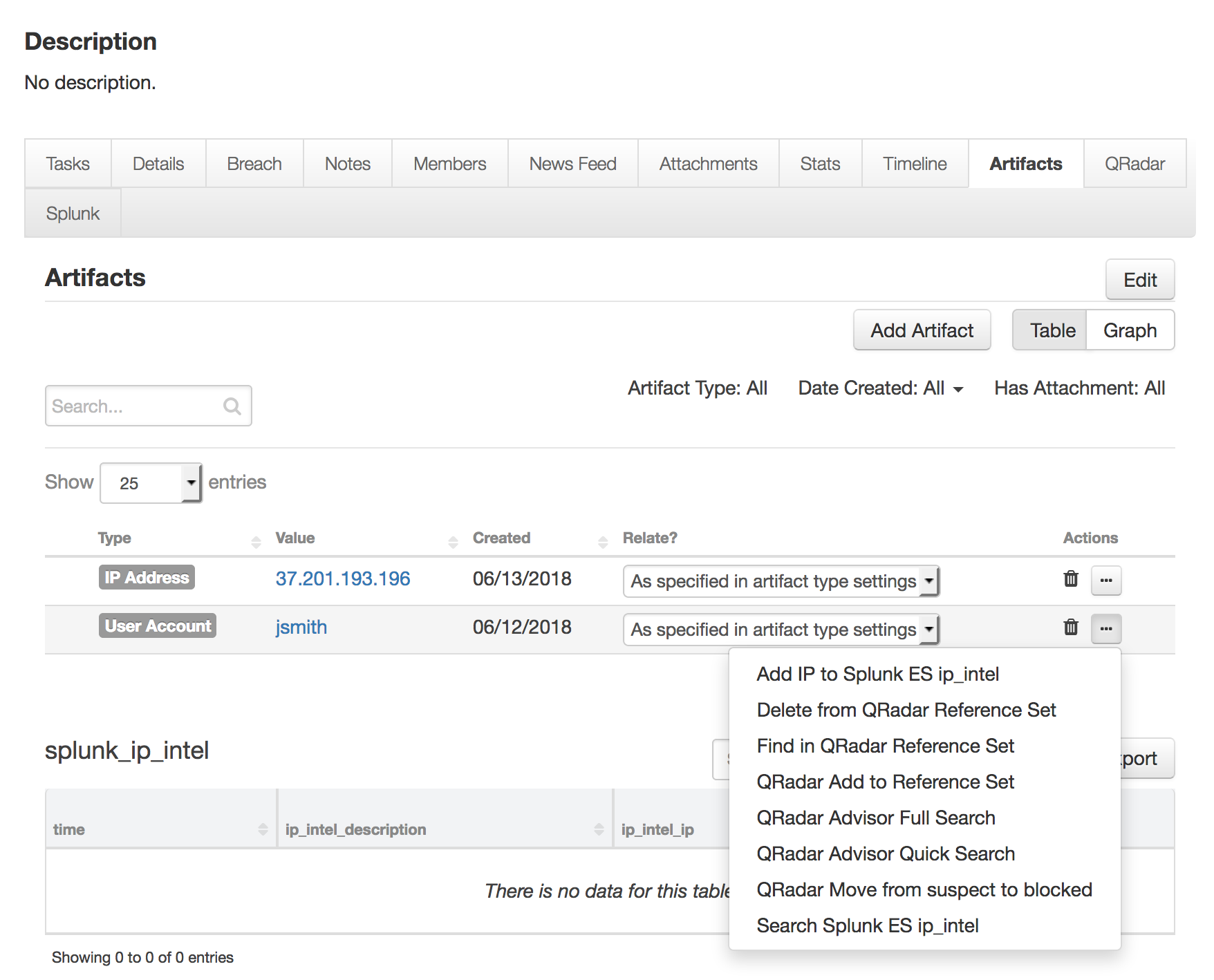
To use the example workflow for this function, user needs to create an incident and then adds an artifact with the proper type as shown above in the full search function.

The QRadar Advisor REST API for quick search returns result 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 incidents, using a dictionary defined there.

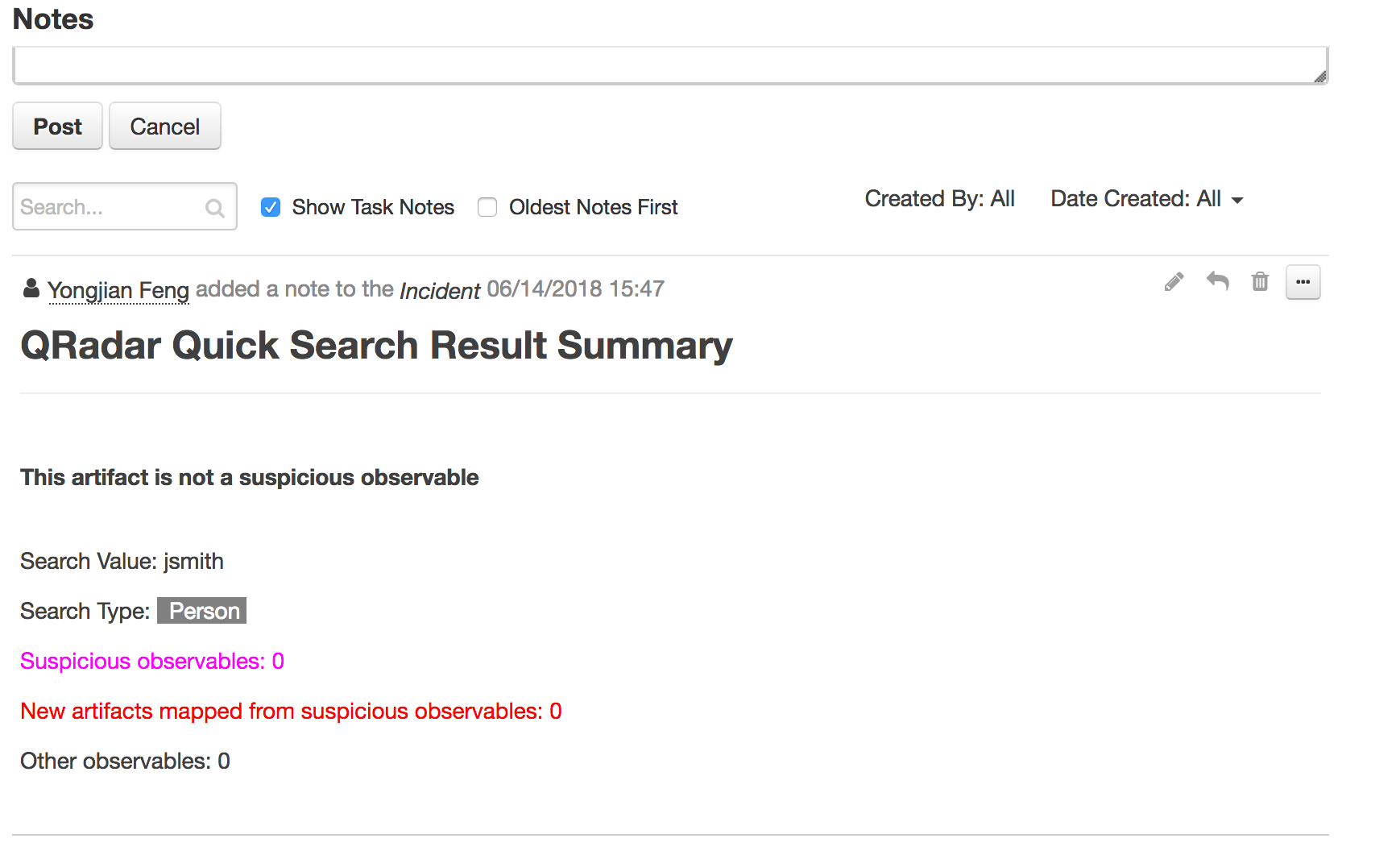
If user wants to map observables to custom artifacts, it can done easily by modifying the mapping dictionary there.

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

In the following example, a quick search of “jsmith” is shown. It starts from clicking QRadar Advisor Quick Search from artifact menu.



For this example, the quick 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 summary this.



qradar\_advisor\_offense\_analysis: QRadar Advisor Offense Analysis

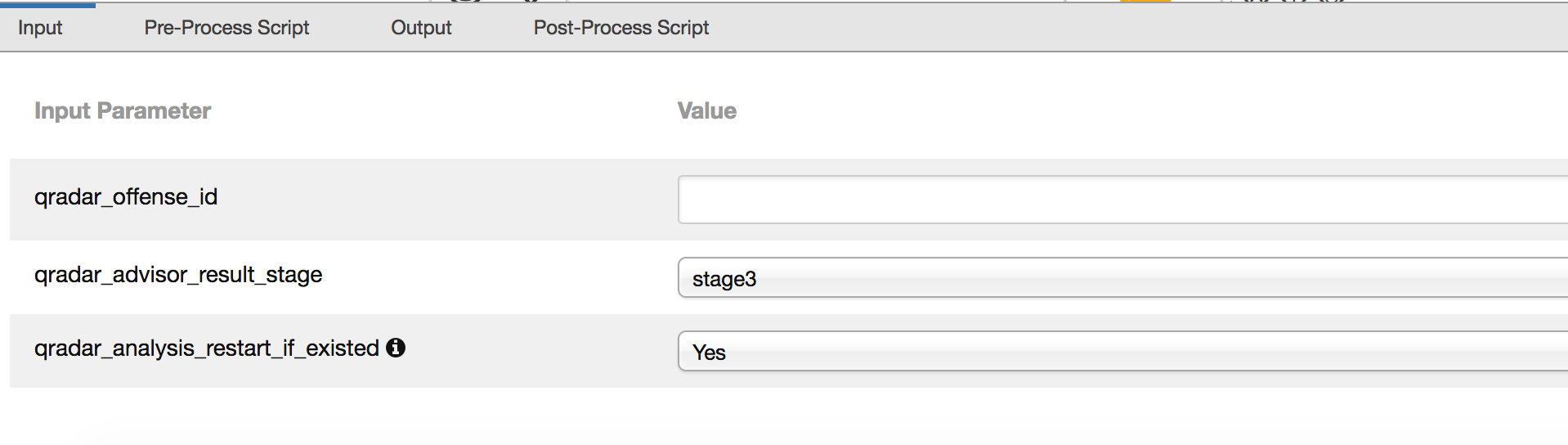
This function calls the QRadar Advisor REST API to

* get the insights of a QRadar offense
* perform analysis of it

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

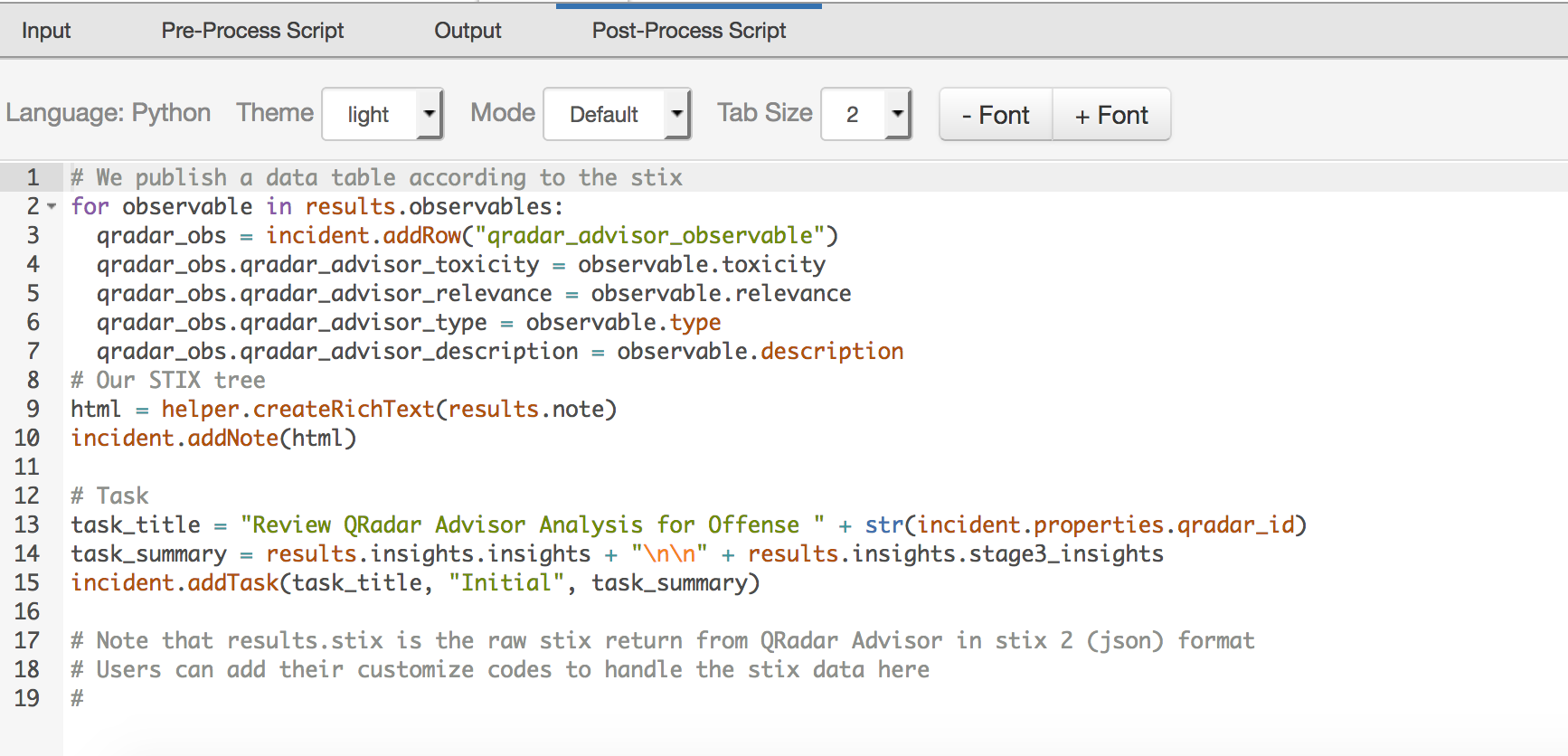
Similar to the QRadar Advisor Full Search, this function generates a HTML representation for the STIX data. It also extracts observables from the STIX objects.

Just like QRadar Advisor Full Search, here user can also specify the return stage from the pre-process script of the example workflow.



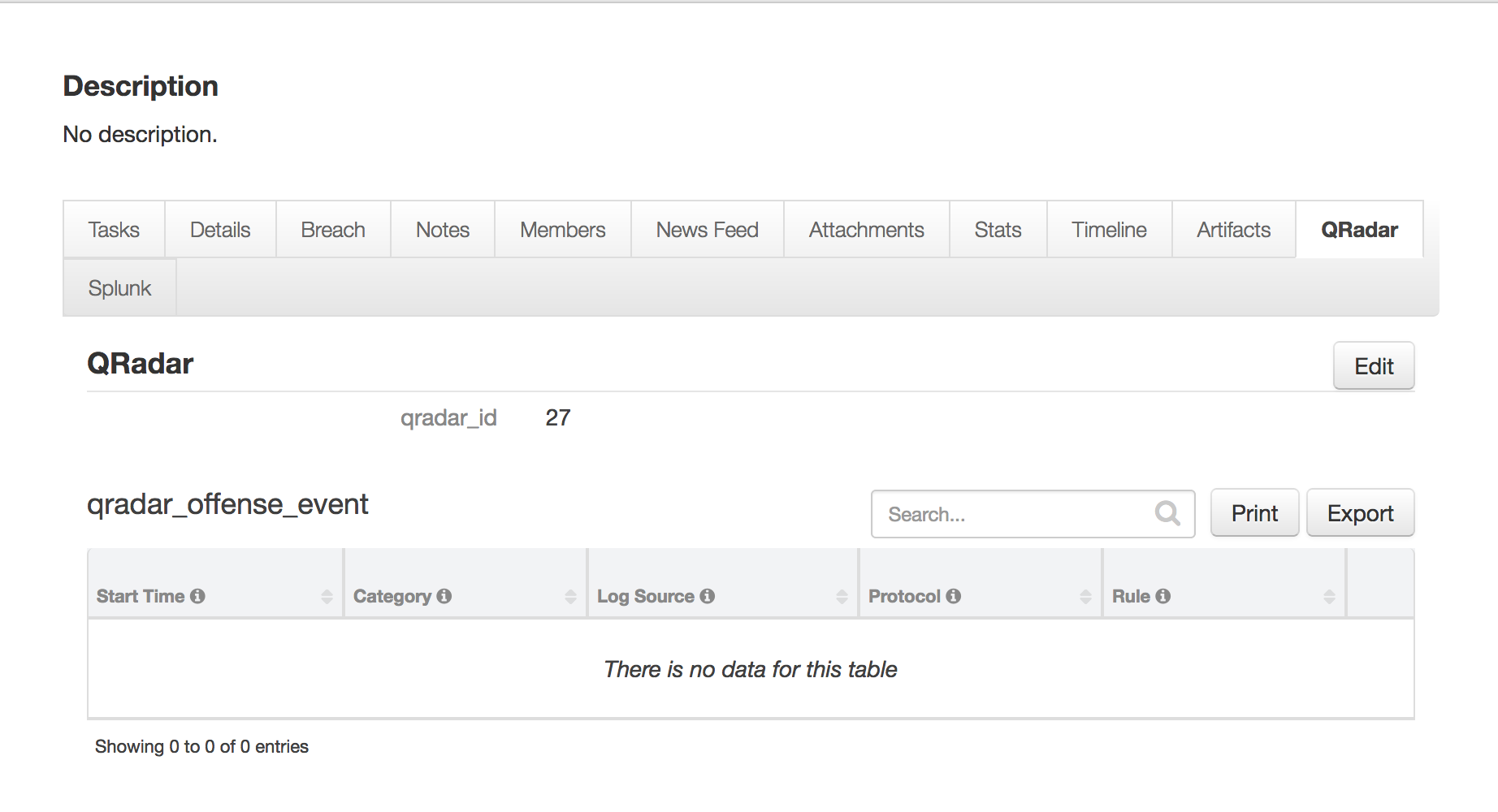
One more setting is qradar\_analysis\_restart\_if\_existed. If this flag is set to Yes, the function restarts a new analysis even if previous result exists for this offense.

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

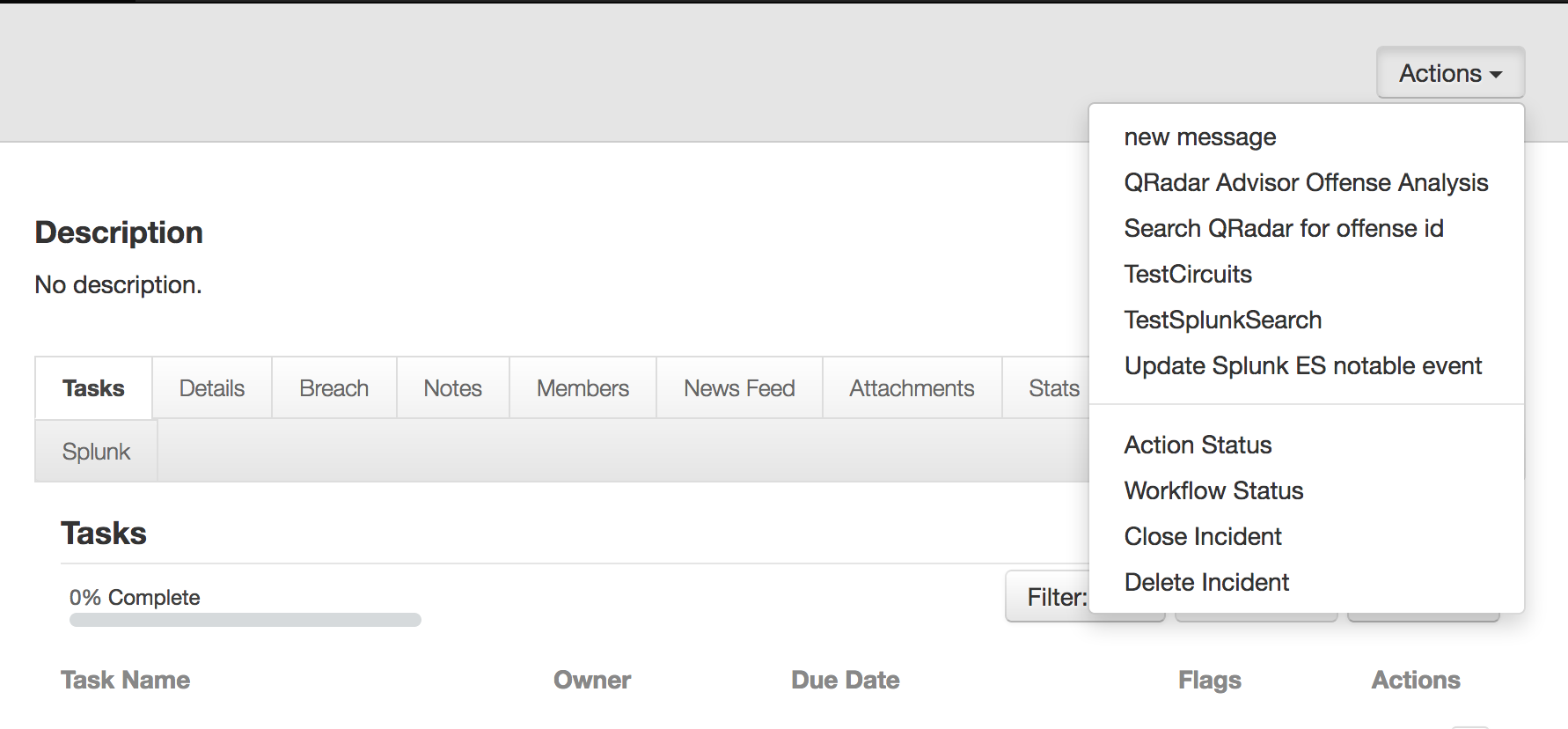


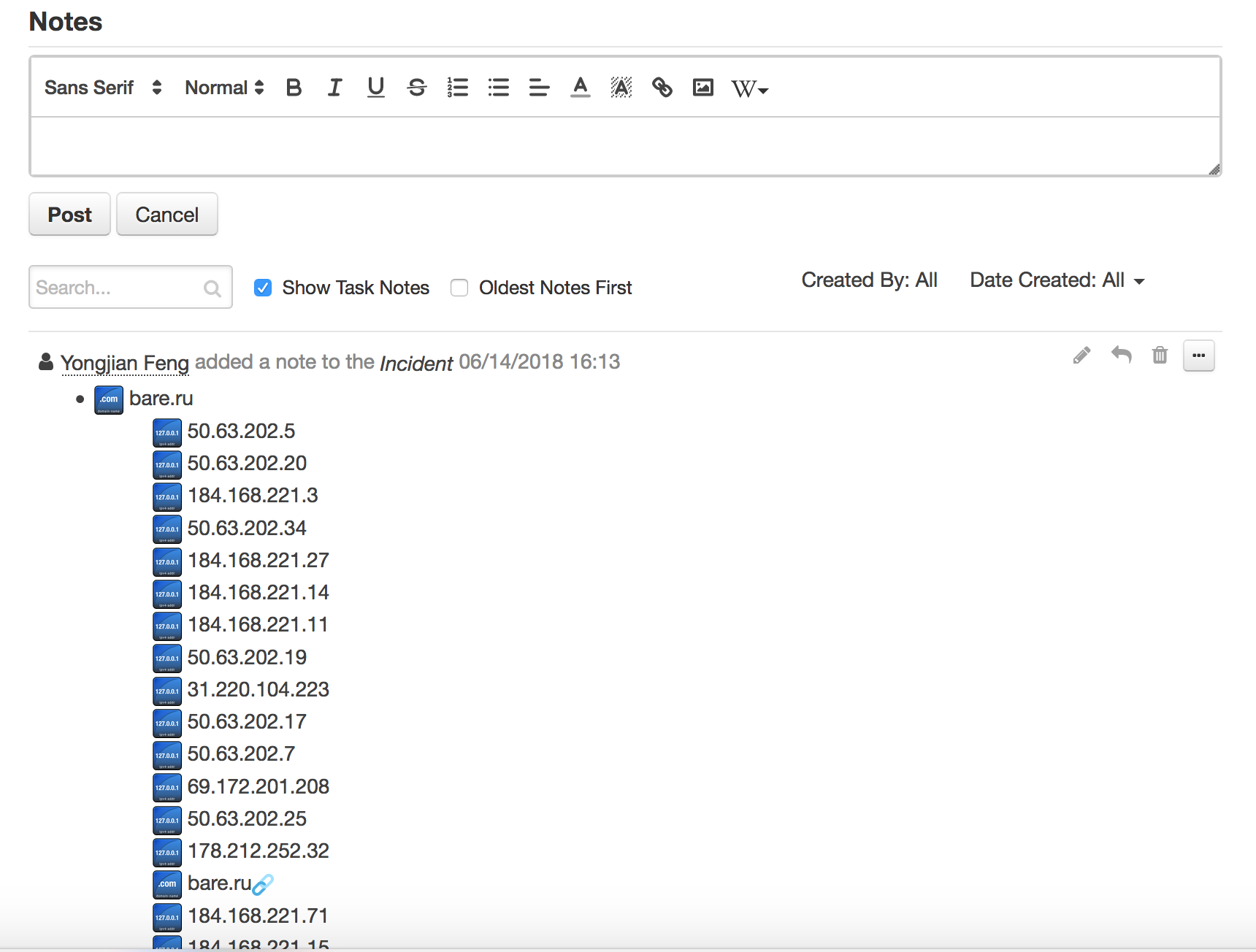
Note the raw STIX data from QRadar Advisor is accessible from the post-process script as results.stix, if the customer wants to create custom code to parse the STIX data.

To use the example workflow, a Resilient incident must be created with a valid QRadar offense id stored in the qradar\_id field. In the following example, this incident is linked to QRadar offense 27.



From the Action menu of this incident, user can select QRadar Advisor Offense Analysis.

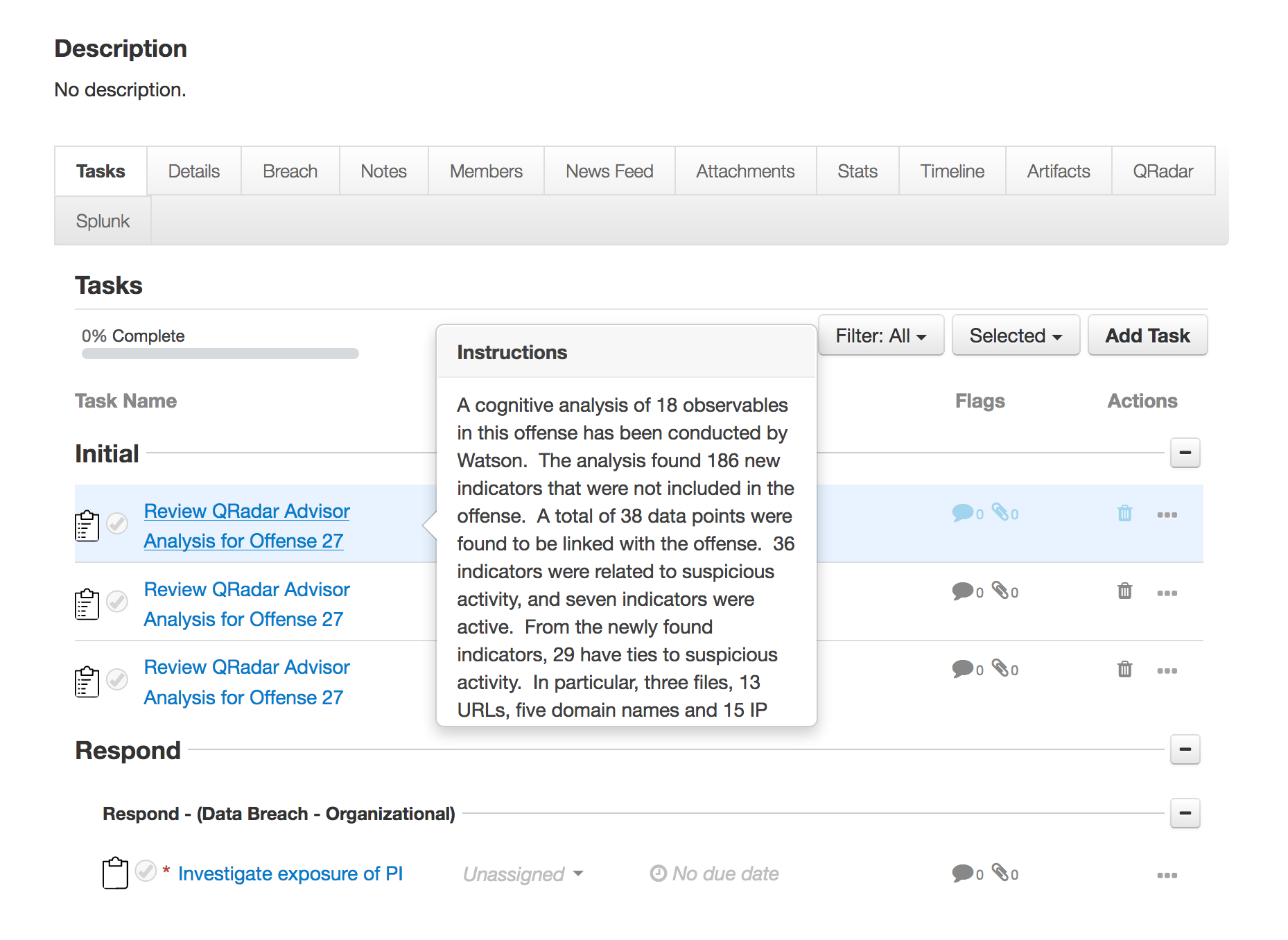


Note, a normal analysis could take up to 20 minutes. Once completed, the HTML representation is shown in the Note tab. 

The observables are added to the QRadar Advisor Observble data table. Note a data table menu rule is included in the package. User can use it to create a new artifact based on the selected row.



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



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](mailto:support@resilientsystems.com).

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