IBM Resilient

Incident Response Platform Integrations

McAfee DXL Function and DXL Subscriber V1.1.0

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 McAfee Publish to DXL Function along with the DXL Subscriber integration.

Overview

The McAfee Publish to DXL function contains the ability to publish a message to an event or a service while the DXL subscriber listens on defined topics and maps the data to the Resilient platform to create incidents and artifacts.

This document describes the McAfee Publish to DXL function and McAfee DXL subscriber integration, how to configure it in custom workflows, and additional customization options.

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).

Install the Python components

The functions package contains Python components that will be 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.

Ensure that the environment is up to date,

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

To install the package:

```
sudo pip install --upgrade fn_mcafee_opendxl-<1.0.0>.tar.gz
```

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.

Perform the following to configure and run the integration:

1. Using 'sudo', become the integration user.

   ```
   sudo su - integration
   ```

2. Create or update the resilient-circuits configuration file.

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

   or

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

3. Edit the resilient-circuits configuration file.

   a. In the [resilient] section, ensure that you provide all the information needed to connect to the Resilient platform.

   b. In the [fn_mcafee_opendxl] section, edit the settings as required.

   ```
   dxlclient_config=<absolute path to dxl config file>

   topic_listener_on=[True|False]

   custom_template_dir=<*optional* path to directory which contains custom
   templates>
   ```

More information on the config file and provisioning the system can be found here:

https://opendxl.github.io/opendxl-client-python/pydoc/provisioningoverview.html

Deploy customizations to the Resilient platform

The package contains the function definition that you can use in workflows, and an example workflow and rule that show how to use the function.

Install these customizations to the Resilient platform with the following command:

```
resilient-circuits customize
```

Answer the prompts to import the function, message destination, workflow and rule. The following data will be imported.

```
Function inputs: mcafee_dxl_payload, mcafee_publish_method,
mcafee_wait_for_response, mcafee_topic_name
Message Destination: McAfee DXL Message Destination
Function: McAfee TIE search hash
Workflow: (Example) McAfee Publish to DXL (Set TIE Reputation), (Example)
McAfee Publish to DXL (Tag System)
Rule: (Example) McAfee Publish to DXL (Set TIE Reputation Known
Malicious), (Example) McAfee Publish to DXL (Tag System Shut Down)
```

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. The following shows a successful connection to the Resilient platform and loading of components.

```
2018-04-12 12:33:49,971 INFO [app] Resilient server: 9.108.163.117
2018-04-12 12:33:49,972 INFO [app] Resilient org: TestOrg
2018-04-12 12:33:49,972 INFO [app] Logging Level: INFO
2018-04-12 12:33:49,973 WARNING [co3] Unverified HTTPS requests
(cafile=false).
2018-04-12 12:33:50,317 INFO [app] Components auto-load directory: (none)
2018-04-12 12:33:50,479 INFO [component_loader] Loading 1 components
2018-04-12 12:33:50,480 INFO [component_loader]
'fn_mcafee_opendxl.components.mcafee_publish_to_dxl.FunctionComponent'
loading
2018-04-12 12:33:50,483 INFO [client] Waiting for broker list...
2018-04-12 12:33:50,521 INFO [client] Trying to connect...
2018-04-12 12:33:50,522 INFO [client] Trying to connect to broker {Unique
id: {brokerID}, Host name: tieserver.resilientsystems, IP address: <IP
Address>, Port: 8883}...
2018-04-12 12:33:50,558 INFO [client] Connected to broker {borkerID}
2018-04-12 12:33:50,606 WARNING [actions_component] Unverified STOMP TLS
certificate (cafile=false)
2018-04-12 12:33:50,607 INFO [stomp_component] Connect to
9.108.163.117:65001
2018-04-12 12:33:50,608 INFO [actions_component]
'fn_mcafee_opendxl.components.mcafee_publish_to_dxl.FunctionComponent'
function 'mcafee_publish_to_dxl' registered to
'mcafee_dxl_message_destination'
2018-04-12 12:33:50,609 INFO [app] Components loaded
2018-04-12 12:33:50,610 INFO [app] App Started
```

```
2018-04-12 12:33:50,716 INFO [actions_component] STOMP attempting to
connect
2018-04-12 12:33:50,717 INFO [stomp_component] Connect to Stomp...
2018-04-12 12:33:50,717 INFO [client] Connecting to 9.108.163.117:65001
...
2018-04-12 12:33:50,757 INFO [client] Connection established
2018-04-12 12:33:50,858 INFO [client] Connected to stomp broker
[session=ID:resilient.localdomain-45666-1523378546811-5:11, version=1.2]
2018-04-12 12:33:50,858 INFO [stomp_component] Connected to
failover:(ssl://9.108.163.117:65001)?maxReconnectAttempts=1,startupMaxReco
nnectAttempts=1
2018-04-12 12:33:50,858 INFO [stomp_component] Client HB: 0  Server HB:
15000
2018-04-12 12:33:50,858 INFO [stomp_component] No Client heartbeats will
be sent
2018-04-12 12:33:50,859 INFO [stomp_component] Requested heartbeats from
server.
2018-04-12 12:33:50,860 INFO [actions_component] STOMP connected.
2018-04-12 12:33:50,961 INFO [actions_component] Subscribe to message
destination 'mcafee_dxl_message_destination'
2018-04-12 12:33:50,962 INFO [stomp_component] Subscribe to message
destination actions.<orgID>.mcafee_dxl_message_destination
```

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.

The unit file should be named 'resilient_circuits.service':

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

The contents:

```
[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
```

Ensure that the service unit file is correctly permissioned:

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

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

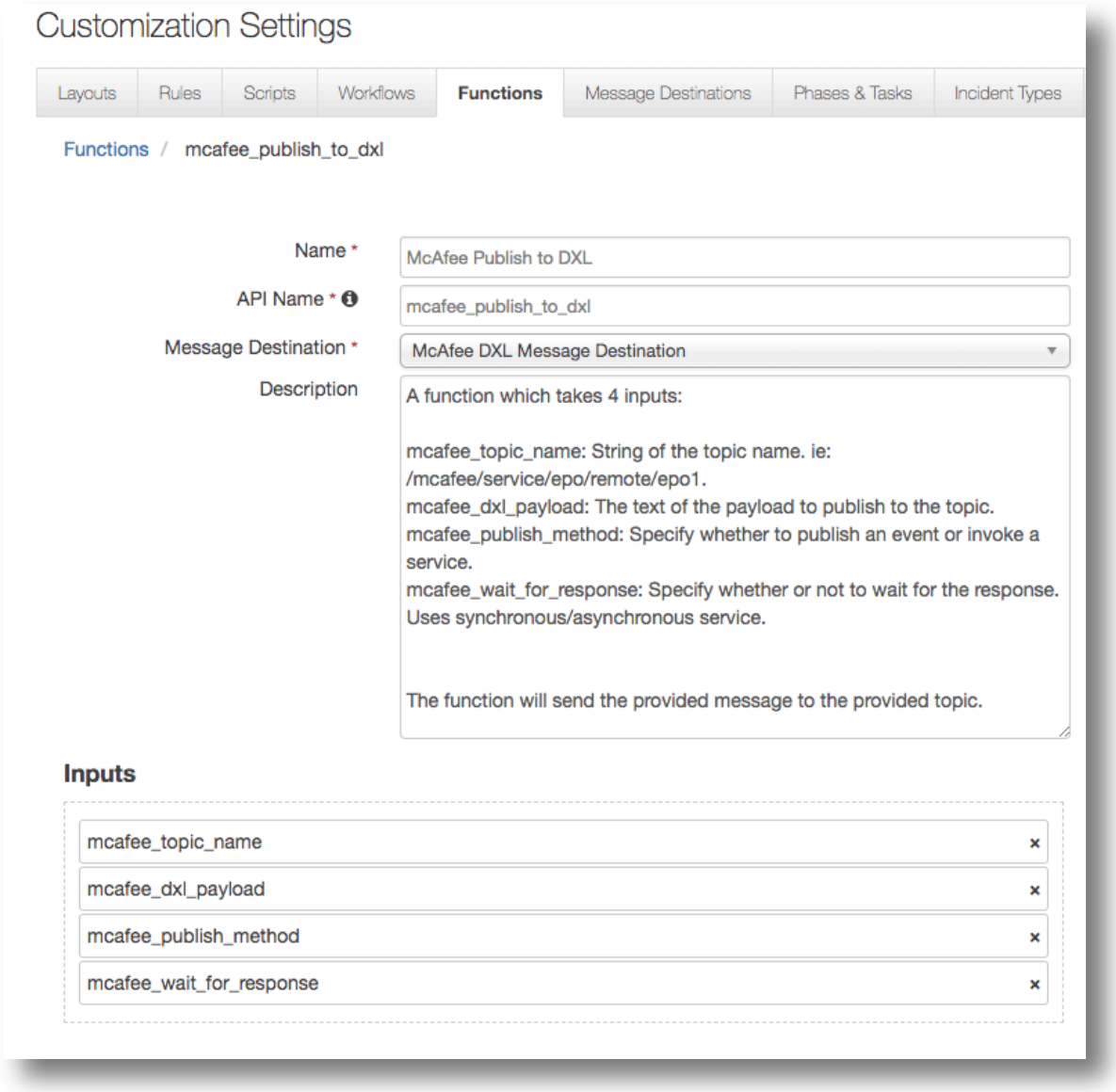
```
sudo systemctl resilient_circuits [start|stop|restart|status]
```

Log files for systemd and the resilient-circuits service can be viewed through the journalctl command:

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

Function Description

Once the function package deploys the function, you can view it in the Resilient platform Functions tab. You can see the function details by clicking its name, as shown in the following screenshot.



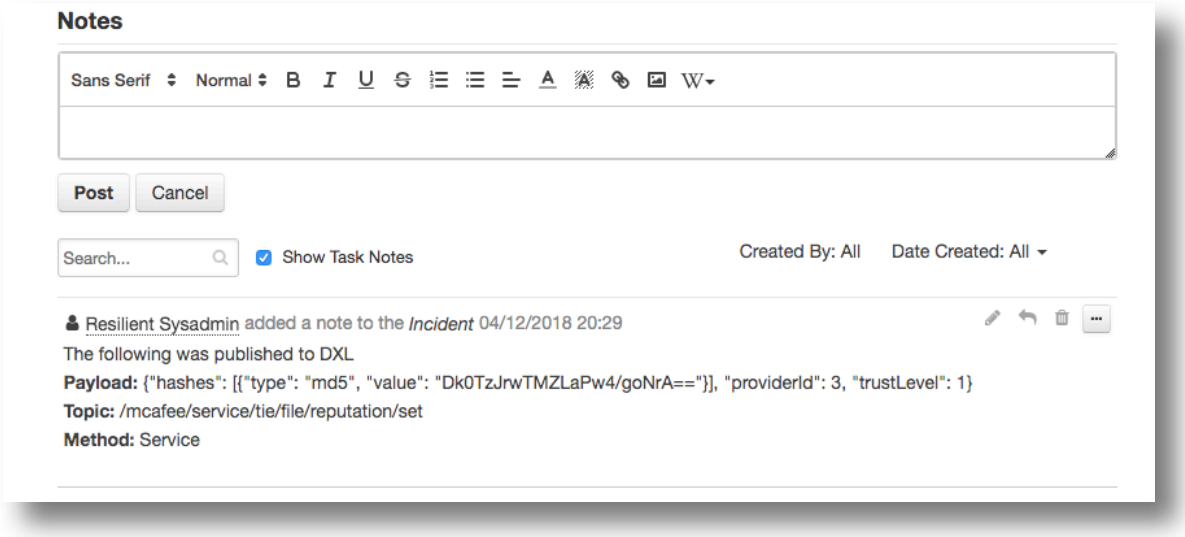
It also includes example workflows and rules that show how the function can be used. You can copy and modify these workflows and rules for your own needs.

Fn_mcafee_opendxl: McAfee Publish to DXL

This function takes four inputs:

```
mcafee_topic_name: Name of the topic to publish the payload to
mcafee_dxl_payload: The payload to be published.
mcafee_publish_method: Use a Service or Event
mcafee_wait_for_response: Wait for a response, only relevant if using a
Service
```

Based on the inputs, the function publishes the payload to the topic to an event or a service and then waits for a response or continues right away. The packaged examples include setting a file reputation for a provider in TIE and tagging a system in ePO. The examples when triggered will then create an incident note showing the inputs as shown below. Note to run the "(Example) McAfee Publish to DXL (Tag System)" workflow your own topic will have to be run which can be done using the following https://github.com/opendxl/opendxl-epo-service-python.



DXL Subscriber Description

The DXL subscriber is designed using Resilient Circuits but does not rely on the functions capabilities. The subscriber connects to the Data Exchange Layer and listens on the topic specified topic(s). When a message is sent to the topic, the integration uses a mapping template to map the data into a Resilient incident DTO and create incidents/artifacts within the Resilient platform. To use the DXL Subscriber the `topic_listener` needs to be set to `True`; otherwise, the listener does not listen on any topics when Resilient Circuits is run.

When you run Resilient Circuits, the subscriber listens on the default topic, `/mcafee/event/epo/threat/response`, and uses the default provided template to map incident and artifact data into the Resilient Platform.

Customize DXL Subscriber

The DXL Subscriber can be configured to listen on any topic, and use any Jinja mapping template.

Create custom template

You can create a custom mapping template using any built-in Jinja2 formatting (refer to Jinja site http://jinja.pocoo.org/). The only additional requirement is the mapping template file type must be either `.jinja` or `.jinja2`.

Add custom templates to directory

You should organize custom templates and place them directly into a specified custom template directory (no subdirectories). This directory can live anywhere Resilient Circuits can access it, and the absolute path should be specified in the app.config:
`custom_template_dir=<absolute_path_to_custom_template_dir>`.

Built in Date-Time formatter

There is one built-in date-time formatting method which helps to convert a string date-time (i.e., 2017-05-17T17:07:59:114Z) into the acceptable millisecond epoch time the Resilient Platform expects. This method can be used in any template by calling:

```
{{ds_to_millis(<date-time_string)}}
```

Override default template

To override a default template, it is best practice to copy the template from the data directory location to the specified custom directory. You can find the default template directory here:

```
<python_env>/lib/<python_version>/site-
packages/fn_mcafee_opendxl/data/templates/
```

The template must be named the same in the custom template directory as it was in the default template directory; otherwise, it will not listen on the correct topic and it will override the default template.

Specify the topic to listen on

You can set the DXL subscriber to listen on any DXL topic name. This is done based on how the template file is named. The name of the template file must be the exact name of the topic that you wish to listen on, with the substitution of forward slashes in the topic name to underscores in the file name. For example, to listen on topic `/mcafee/event/epo/threat/response`, the template file name to listen on that topic must be `_mcafee_event_epo_threat_response.<jinja/jinja2>`.

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.