IBM Resilient

resilient

Incident Response Platform Integrations

McAfee TIE Function V1.0.0

Release Date: April 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 McAfee TIE Function.

Overview

The McAfee TIE function contains the ability to search McAfee TIE for information on a specific file hash

This document describes the McAfee TIE function, 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_tie-<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_tie] section, edit the settings as required.

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

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_tie_hash, mcafee_tie_hash_type
Message Destination: McAfee TIE Message Destination
Function: McAfee TIE search hash
Workflow: (Example) McAfee hash search workflow
Rule: (Example) McAfee artifact hash search
Data Table: TIE Results
```

Once the customizations are added, you have to manually add the data table to a layout to see the results. You do this in the Resilient platform as follows:

1. In Customization Settings, select the **Layouts** tab.
2. Click **Incident Tabs** then select the specific tab where you want to place the data table.
3. Drag and drop the TIE Results Data Table to the preferred spot in the tab layout.

For a more detailed procedure, see the *Resilient Incident Response Platform Playbook Designer Guide*.

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.

```
2018-04-10 16:55:49,218 INFO [app] Resilient server: 9.108.163.130
2018-04-10 16:55:49,219 INFO [app] Resilient org: TestOrg
2018-04-10 16:55:49,219 INFO [app] Logging Level: INFO
2018-04-10 16:55:49,220 WARNING [co3] Unverified HTTPS requests
(cafile=false).
2018-04-10 16:55:49,785 INFO [app] Components auto-load directory: (none)
2018-04-10 16:55:49,973 INFO [component_loader] Loading 1 components
2018-04-10 16:55:49,983 INFO [component_loader]
'fn_mcafee_tie.components.mcafee_tie_search_hash.FunctionComponent'
loading
2018-04-10 16:55:49,987 INFO [client] Waiting for broker list...
2018-04-10 16:55:50,016 INFO [client] Trying to connect...
2018-04-10 16:55:50,017 INFO [client] Trying to connect to broker {Unique
id: {<yourID>}, Host name: tieserver.resilientsystems, IP address:
<TIE_IP>, Port: 8883}...
2018-04-10 16:55:50,056 INFO [client] Connected to broker {<yourID>}
2018-04-10 16:55:50,104 WARNING [actions_component] Unverified STOMP TLS
certificate (cafile=false)
2018-04-10 16:55:50,105 INFO [stomp_component] Connect to
9.108.163.130:65001
```

```
2018-04-10 16:55:50,106 INFO [actions_component]
'fn_mcafee_tie.components.mcafee_tie_search_hash.FunctionComponent'
function 'mcafee_tie_search_hash' registered to 'mcafee_tie_md'
2018-04-10 16:55:50,107 INFO [app] Components loaded
2018-04-10 16:55:50,109 INFO [app] App Started
2018-04-10 16:55:50,212 INFO [actions_component] STOMP attempting to
connect
2018-04-10 16:55:50,213 INFO [stomp_component] Connect to Stomp...
2018-04-10 16:55:50,213 INFO [client] Connecting to 9.108.163.130:65001
...
2018-04-10 16:55:50,232 INFO [client] Connection established
2018-04-10 16:55:50,338 INFO [client] Connected to stomp broker
[session=ID:resilient.localdomain-45666-1523378546811-5:2, version=1.2]
2018-04-10 16:55:50,339 INFO [stomp_component] Connected to
failover:(ssl://9.108.163.130:65001)?maxReconnectAttempts=1,startupMaxReco
nnectAttempts=1
2018-04-10 16:55:50,339 INFO [stomp_component] Client HB: 0  Server HB:
15000
2018-04-10 16:55:50,339 INFO [stomp_component] No Client heartbeats will
be sent
2018-04-10 16:55:50,339 INFO [stomp_component] Requested heartbeats from
server.
2018-04-10 16:55:50,340 INFO [actions_component] STOMP connected.
2018-04-10 16:55:50,442 INFO [actions_component] Subscribe to message
destination 'mcafee_tie_md'
2018-04-10 16:55:50,442 INFO [stomp_component] Subscribe to message
destination actions.<orgID>.mcafee_tie_md
```

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:

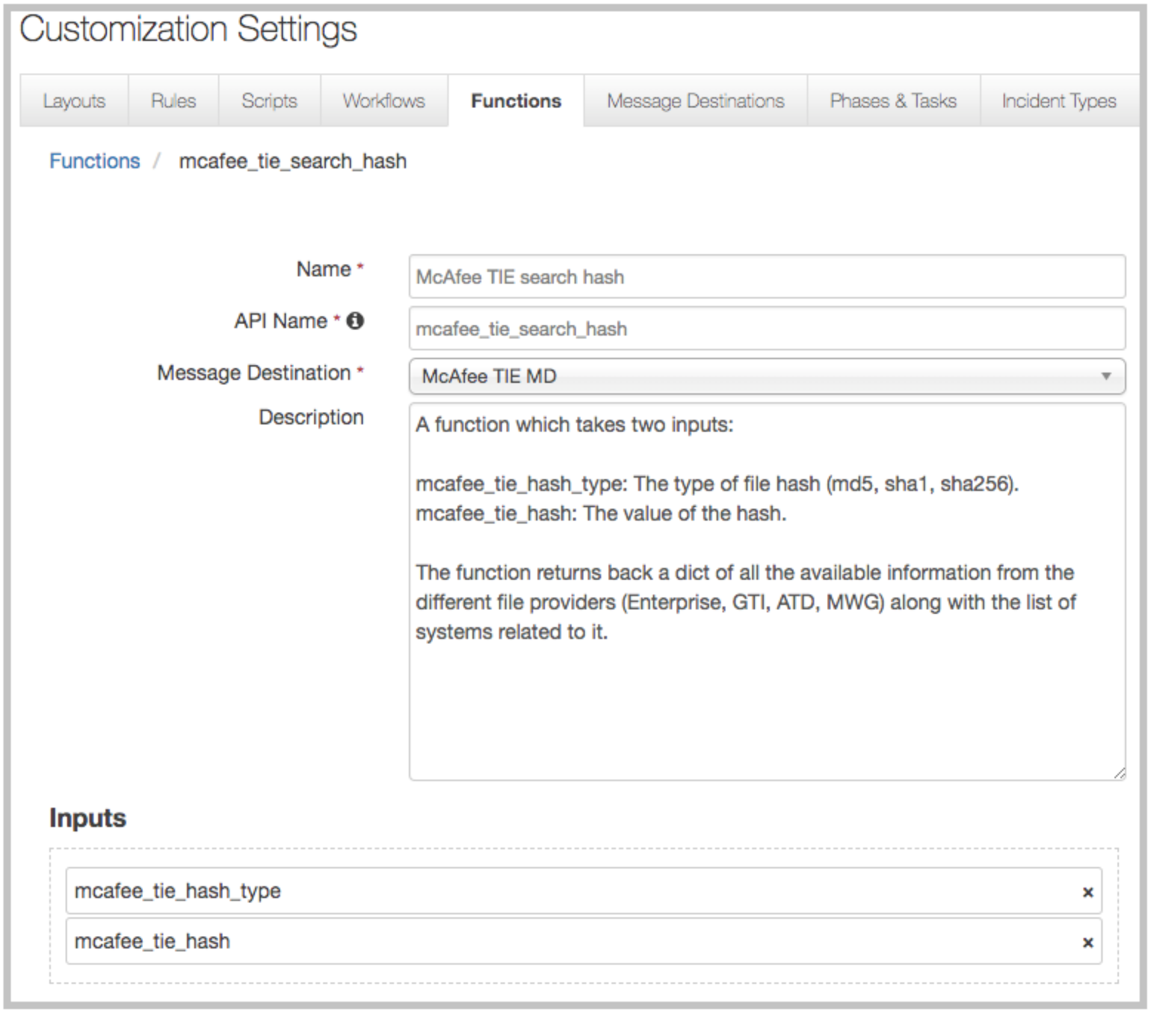
```
sudo systemctl resilient_circuits [start|stop|restart|status]
```

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.



Name * McAfee TIE search hash

API Name * mcafee_tie_search_hash

Message Destination * McAfee TIE MD

Description A function which takes two inputs:

mcafee_tie_hash_type: The type of file hash (md5, sha1, sha256).
mcafee_tie_hash: The value of the hash.

The function returns back a dict of all the available information from the different file providers (Enterprise, GTI, ATD, MWG) along with the list of systems related to it.

Inputs

mcafee_tie_hash_type

mcafee_tie_hash

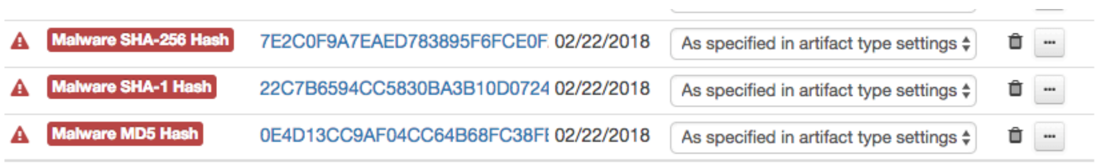
It also includes an example workflow and rule that show how the function can be used. You can copy and modify these workflows and rules for your own needs.

Fn_mcafee_tie: McAfee TIE search hash

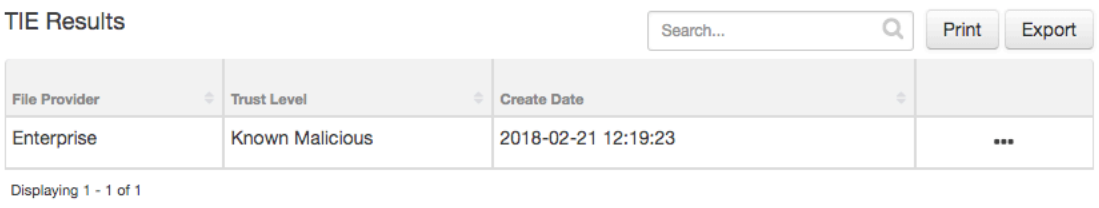
This function searches for information on a file hash within TIE. The function is provided a file hash along with the type of hash, and returns information relative to the file. This information can come from any of the providers (Enterprise, GTI, ATD, MWG) in addition to a system list. The following is an example of a response:

```
{
   "GTI":{
      "File Provider":"GTI",
      "Attributes":{
      },
      "Create Date":"2018-02-21 12:17:10",
      "Trust Level":"Known Malicious"
   },
   "ATD":{
      "File Provider":"ATD",
      "Create Date":"2018-03-14 11:53:09",
      "Trust Level":"Most Likely Malicious"
   },
   "MWG":{
      "File Provider":"MWG",
      "Create Date":"2018-03-14 11:53:55",
      "Trust Level":"Most Likely Malicious"
   },
   "Enterprise":{
      "File Provider":"Enterprise",
      "Attributes":{
         "Average Local Rep":"Most Likely Malicious",
         "First Contact":"2018-02-21 12:17:10",
         "Min Local Rep":"Most Likely Malicious",
         "Is Prevalent":"0",
         "File Name Count":"1",
         "Max Local Rep":"Most Likely Malicious"
      },
      "Create Date":"2018-02-21 12:17:10",
      "Trust Level":"Most Likely Malicious"
   }
   "system_list":[{
     "date": 1519233563,
     "agentGuid": {a00728ff-3187-46c1-97d2-8e0f26ea940b}
   }]
}
```

The out of the box example includes a manual rule which is triggered off of an artifact of type of Malware MD5 Hash, Malware SHA-1 Hash, or Malware SHA-256 Hash. A search is then performed on the hash in TIE and the TIE Results data table is populated.



Malware SHA-256 Hash	7E2C0F9A7EAED783895F6FCE0F	02/22/2018	As specified in artifact type settings	
Malware SHA-1 Hash	22C7B6594CC5830BA3B10D0724	02/22/2018	As specified in artifact type settings	
Malware MD5 Hash	0E4D13CC9AF04CC64B68FC38FI	02/22/2018	As specified in artifact type settings	



TIE Results

File Provider	Trust Level	Create Date	
Enterprise	Known Malicious	2018-02-21 12:19:23	•••

Displaying 1 - 1 of 1

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.