IBM Resilient

Incident Response Platform Integrations

Microsoft Security Graph Integration V1.0.0

Release Date: December 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 Microsoft Security Graph Integration.

Overview

The Microsoft Security Graph functions contains the ability to call multiple security endpoints within the Microsoft Graph, while the Alert Polling Integration allows for creation of new incidents in the Resilient platform from alerts.

This document describes the Microsoft Security Graph integration, its customization options, and how to configure them in custom workflows.

Installation

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

- Resilient platform is version 31 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 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_microsoft_security_graph-1.0.0.zip
```

If this is a zip package with a tar.gz file inside, change step 2 to this:

To install the package, you must first unzip it then install the package as follows:

```
sudo pip install --upgrade fn_microsoft_security_graph-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.

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_microsoft_security_graph] section, edit the settings as follows:

```
# Graph URL with version number
microsoft_graph_url=https://graph.microsoft.com/v1.0/
tenant_id=<Tenant directory id>
client_id=<App client id>
client_secret=<App client secret>

## Polling options
# How often polling should happen. Value is in seconds. To disable
polling, set this to zero.
msg_polling_interval=0
#incident_template=<location_of_template_file>  # If not set uses default
template.

# String query to apply to the alert polling component. This will be
added to the end of the url
# when searching for alerts. The example shown below would make the whole
search url equal to
# https://graph.microsoft.com/v1.0/security/alerts/?$filter=assignedTo eq
'analyst@m365x594651.onmicrosoft.com' and severity eq 'high'
#alert_query=filter=assignedTo%20eq%20'analyst@m365x594651.onmicrosoft.co
m'%20and%20severity%20eq%20'high'

# Alert Time range sec - Optional value in seconds to set the start
dateTime values for the createdDateTime field when filtering alerts.
# This is calculated by adding to the filter 'createdDateTime ge
(current_dateTime - alert_time_range_sec)
#alert_time_range_sec=3600
```

Deploy customizations to the Resilient platform

The package contains function definitions that you can use in workflows, and includes example workflows and rules that show how to use these functions. Also includes a custom incident field.

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 and rules. The following data will be imported.

```
Incident fields:
  microsoft_security_graph_alert_id
Action fields:
  microsoft_security_graph_alert_assignedto
  microsoft_security_graph_alert_closeddatetime
  microsoft_security_graph_alert_comment
```

```
  microsoft_security_graph_alert_feedback
  microsoft_security_graph_alert_status
  microsoft_security_graph_alert_tags
  microsoft_security_graph_query_end_datetime
  microsoft_security_graph_query_start_datetime
Function inputs:
  microsoft_security_graph_alert_data
  microsoft_security_graph_alert_id
  microsoft_security_graph_alert_search_query
Message Destinations:
  microsoft_security_graph_message_destination
Functions:
  microsoft_security_graph_alert_search
  microsoft_security_graph_get_alert_details
  microsoft_security_graph_update_alert
Workflows:
  example_microsoft_security_graph_alert_search
  example_microsoft_security_graph_get_alert_details
  example_microsoft_security_graph_resolve_alert
  example_microsoft_security_graph_update_alert
Rules:
  Example Microsoft Security Graph Update Alert
  Example: Microsoft Security Graph Alert Search
  Example: Microsoft Security Graph Get Details
  Example: Microsoft Security Graph Resolve Alert
```

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
   ```

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:

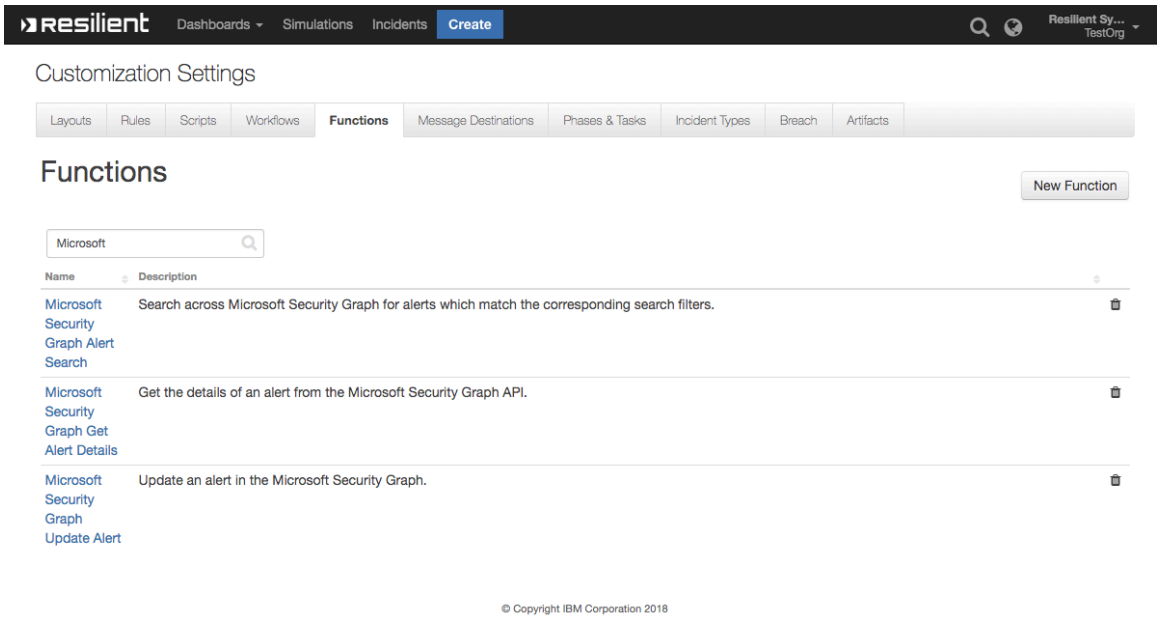
```
sudo systemctl resilient_circuits [start|stop|restart|status]
```

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"
```

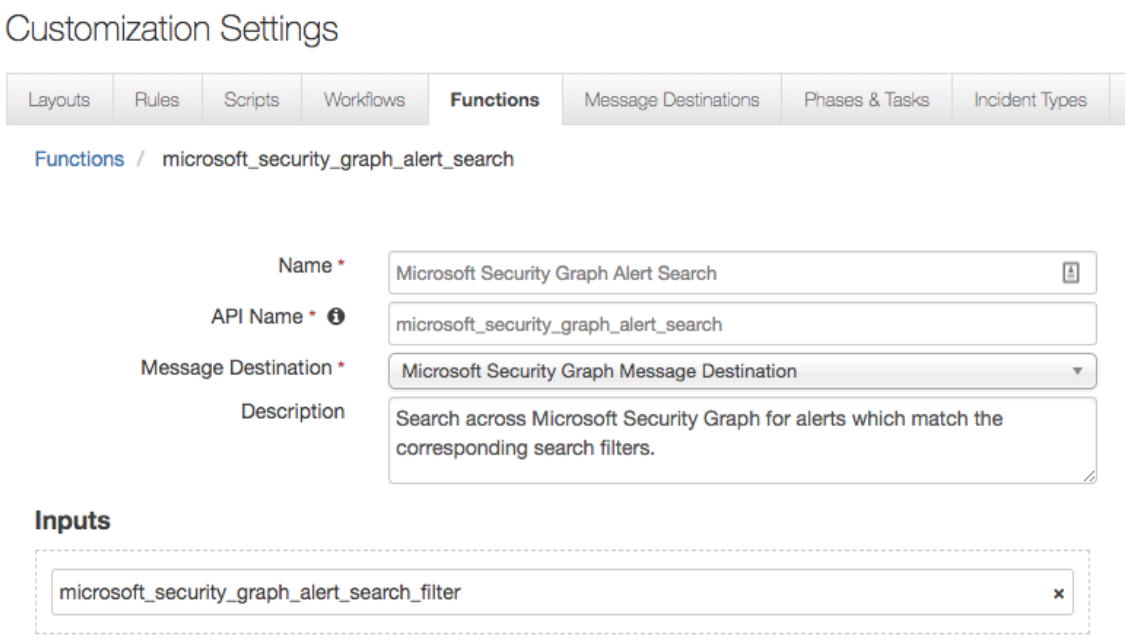
Function Descriptions

Once the function package deploys the 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.



Microsoft Security Graph Alert Search

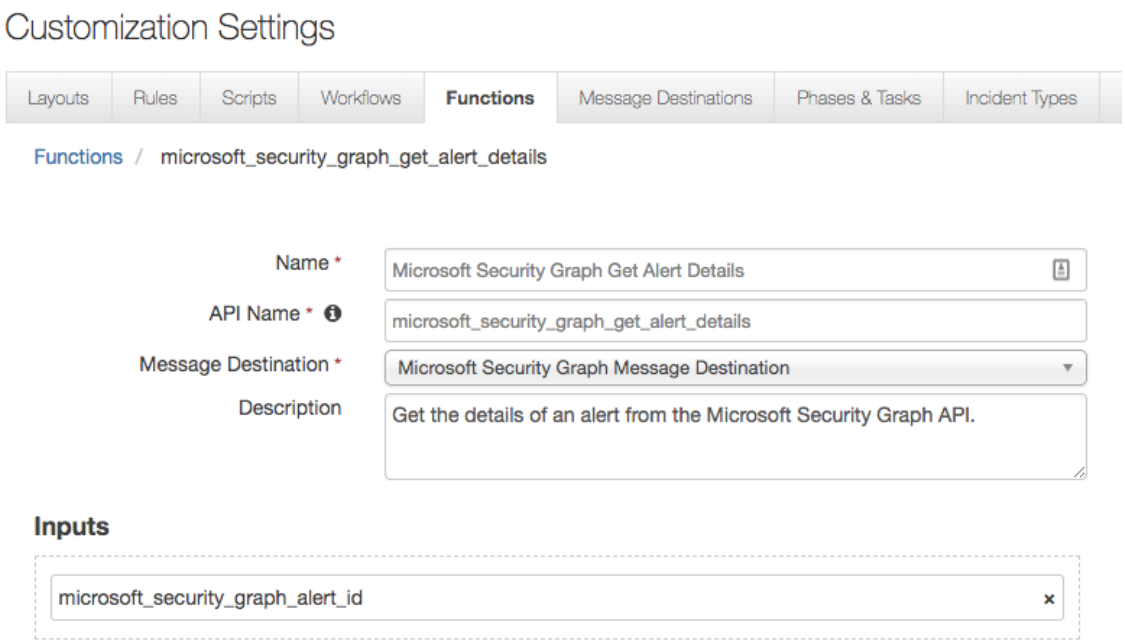
The Microsoft Security Graph Alert Search function allows for alerts to be searched for across the tenant's data in the Microsoft Graph. The function accepts one optional input `microsoft_security_graph_alert_search_query` which is an OData query string to return alerts back.



The default workflow for this function is run against an artifact and searches for all other alerts with a similar artifact value in a specified time range.

Microsoft Security Graph Get Alert Details

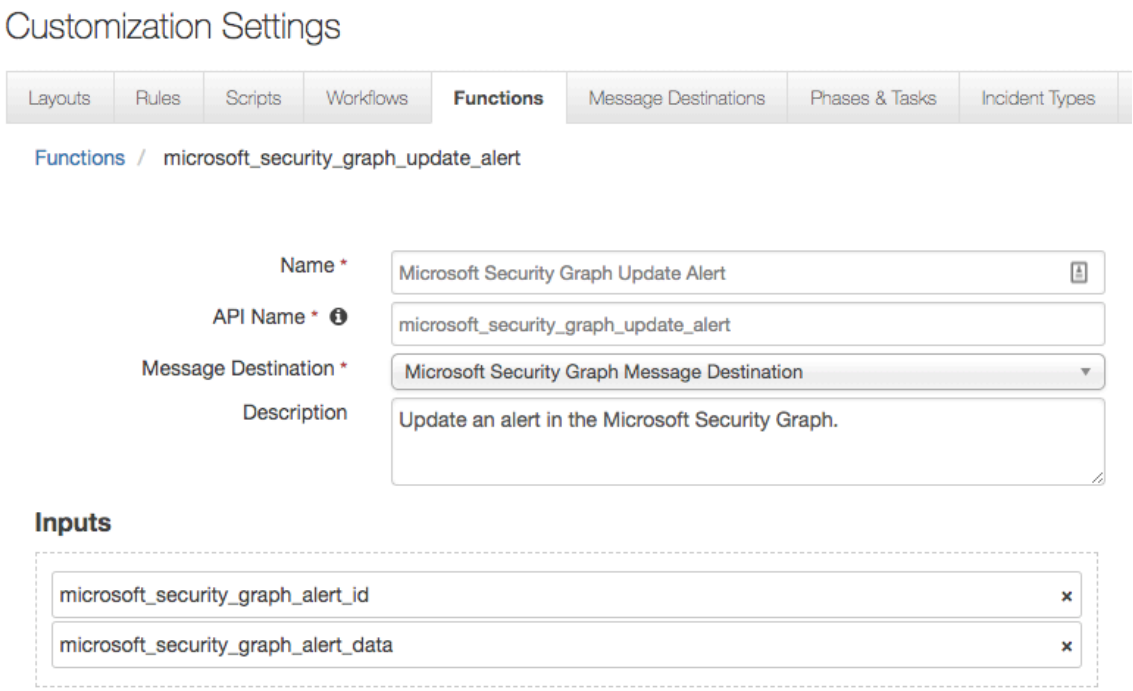
The Microsoft Security Graph Get Alert Details function returns back all details for a specific alert. This function takes one input `microsoft_security_graph_alert_id` which is the id specific of the alert to return details on.



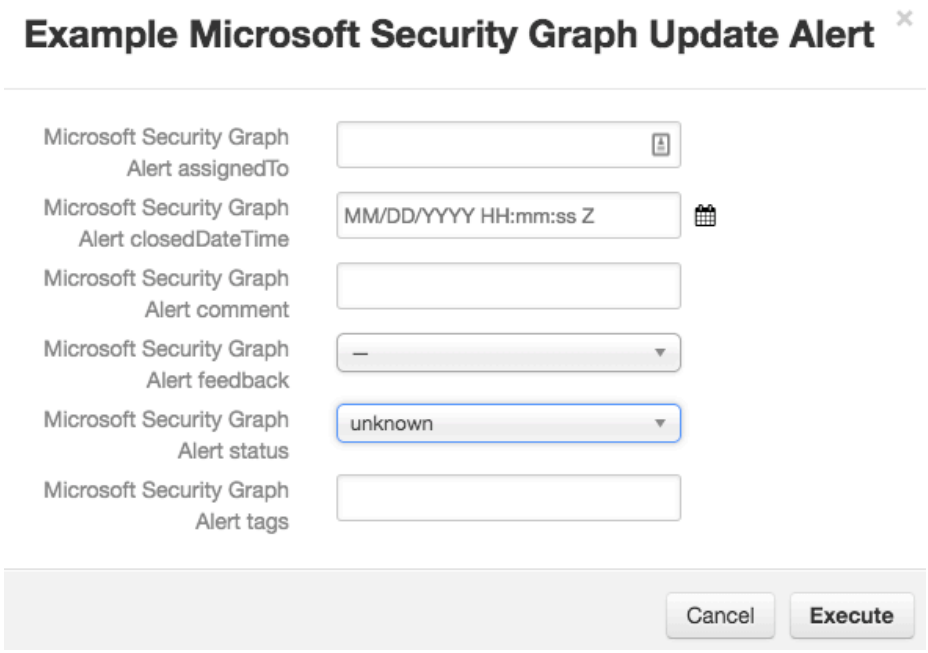
The default workflow for this function is run against an incident which has the custom incident field `microsoft_security_graph_alert_id` set and potential artifacts to the incident.

Microsoft Security Graph Update Alert

The Microsoft Security Graph Update Alert function is used to update an alert in the Microsoft Graph from the Resilient platform. The function accepts two inputs, the alert's unique id microsoft_security_graph_alert_id and a JSON string of the data to update an alert with microsoft_security_graph_alert_data.



The default workflows for this function include being triggered from an incident to update the alert based on a number of fields from a pop up shown below, and another one is triggered when the incident is closed and sets the alert status to resolved.



Alert Polling Integration Description

When loaded and set to poll, the alert polling integration spawns a new thread to handle all the polling and creation of incidents. For this to happen, set `msg_polling_interval` to a positive integer within the app.config file. This is enable Microsoft Security Graph polling and set the polling interval in second, to disable polling set this to an integer less than 1. `incident_template` can be set to the location of a jinja template used to create incident data, if this is not set, the integration will default to using the default packaged template.

When the component is loaded and `msg_polling_interval` is a positive integer, a new thread is created. This thread reaches out to the Microsoft Graph and returns a list of alerts, narrowing results down when `alert_query` is set. Since `alert_query` is full OData this can start with `filter=`, `top=`, `skip=`, etc; just do not start it with `$` as that character is reserved for environment variables. From here, this list is cross-referenced with active incident within the Resilient platform. If the alert already exists as an active incident in the Resilient platform, it moved on to the next alert; otherwise, a new incident is created in the Resilient platform based on the alert data.

When the incident is created, the Microsoft Security Graph Alert ID custom field is set to the ID of the alert in the Microsoft Graph to ensure the connection between alerts and incidents.

The incident template field can be edited to meet custom needs. The suggested way of accomplishing this is copying the default template that comes with the integration to a new directory and editing it from there. This template can be found at: `<python_env>/lib/<python_version>/site-packages/fn_microsoft_security_graph/data/templates/`. The template utilizes jinja. More documentation can be found at something. Once the custom template is finished, set its location in the config file at: `incident_template=<location_of_template>`.

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.