IBM Resilient

resilient

Incident Response Platform Integrations

Risk Fabric Function V1.0.0

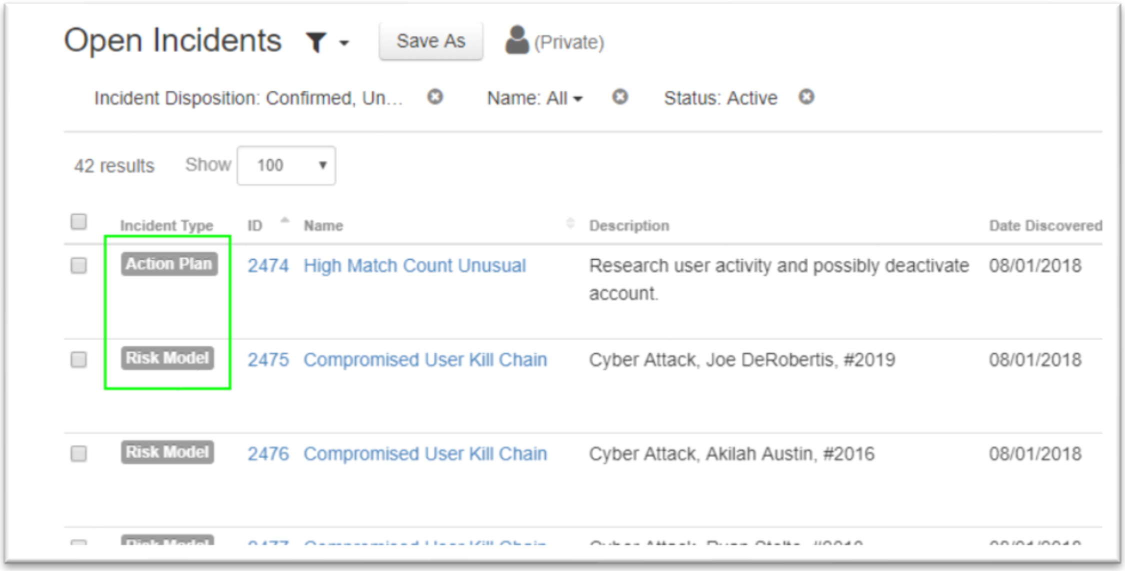
Release Date: September 2018

Resilient Functions simplify development of the integrations by sending data from the Resilient platform to a remote program that performs an activity then returns the results to the function. The results can be acted upon by a script and the result of that becomes a decision point in the Resilient workflow.

This guide describes the Risk Fabric Function.

Overview

The Risk Fabric integration with the Resilient platform allows for the querying of Risk Ratings for Artifacts like IPs, Computer Endpoints, and Users. Risk Models, Event Scenarios, and Action Plans can be pulled into Resilient and created as Incidents and then fully mitigated or classified.

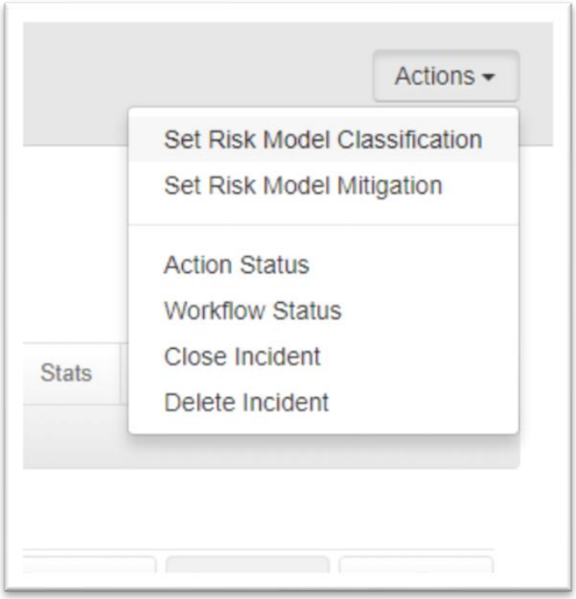


Incident Disposition: Confirmed, Un… Name: All Status: Active

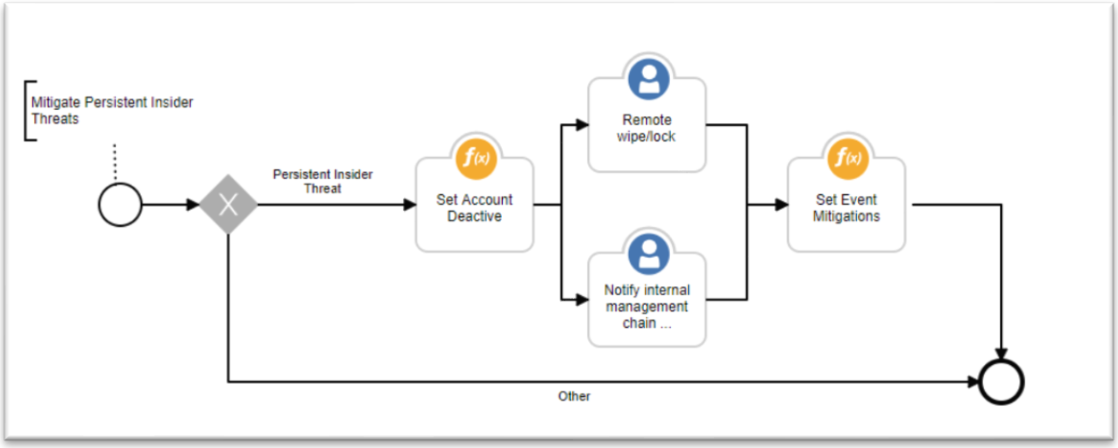
42 results Show 100

□	Incident Type	ID	Name	Description	Date Discovered
□	Action Plan	2474	High Match Count Unusual	Research user activity and possibly deactivate account.	08/01/2018
□	Risk Model	2475	Compromised User Kill Chain	Cyber Attack, Joe DeRobertis, #2019	08/01/2018
□	Risk Model	2476	Compromised User Kill Chain	Cyber Attack, Akilah Austin, #2016	08/01/2018

Manually perform Classifications and Mitigation actions on Risk Models, Event Scenarios, and Action Plans via rules:



Or automatically via advanced workflows:


Mitigate Persistent Insider Threats
Persistent Insider Threat
Set Account Deactive
Remote wipe/lock
Notify internal management chain ...
Set Event Mitigations
Other

Setup

The following lists the system requirements:

- Python version 2.7.10 or later, or version 3.6 or later
- Resilient Circuits and Resilient Python libraries version 30.0 or later
- Resilient platform version 30.0 or later
- Risk Fabric version 6.5.1 or later

Perform the following to install and configure the function:

1. Ensure the environment is up to date:
   ```
   sudo pip install --upgrade pip
   sudo pip install --upgrade setuptools
   sudo pip install --upgrade resilient-circuits
   ```
2. Install the required software for the function (if not already installed):
   ```
   sudo pip install fn_risk_fabric-<version>.tar.gz
   ```
3. Add the function to the Resilient platform:
   ```
   resilient-circuits customize
   ```
   You are prompted to answer prompts to import functions, message destinations, etc.
4. From the account used for Integrations, use one of the following commands to configure the Risk Fabric settings. Use –c for new environments or –u for existing environments.
   ```
   resilient-circuits config -c
   ```
   OR
   ```
   resilient-circuits config -u
   ```

5. Edit the .resilient/app.config file and section [fn_risk_fabric]:

```
server=<risk fabric url>
username=<risk fabric api user>
password=<risk fabric api password>
```

After completing the configuration steps, enter the `resilient-circuits run` command. The following is an example of the resulting messages indicating the successful connection to the Resilient platform and the loading of the Risk Fabric integration modules.

```
$ resilient-circuits run

2018-04-07 12:38:04,164 INFO [app] Configuration file:
/Users/Integration/.resilient/app.config

2018-04-07 12:38:04,165 INFO [app] Resilient server: <host>

2018-04-07 12:38:04,165 INFO [app] Resilient user: <acct>

2018-04-07 12:38:04,165 INFO [app] Resilient org: <org>

2018-04-07 12:38:04,165 INFO [app] Logging Level: INFO

…

2018-04-07 12:38:05,418 INFO [component_loader]
'fn_risk_fabric.components.get_host_risk.FunctionComponent' loading

2018-04-07 12:38:05,419 INFO [component_loader]
'fn_risk_fabric.components.get_ip_risk.FunctionComponent' loading

2018-04-07 12:38:05,420 INFO [component_loader]
'fn_risk_fabric.components.get_user_risk.FunctionComponent' loading

2018-04-07 12:38:05,421 INFO [component_loader]
'fn_risk_fabric.components.get_risk_model_instances.FunctionComponent' loading

2018-04-07 12:38:05,422 INFO [component_loader]
'fn_risk_fabric.components.get_risk_model_instance_details.FunctionComponent' loading

2018-04-07 12:38:05,423 INFO [component_loader]
'fn_risk_fabric.components.get_action_plans.FunctionComponent' loading

2018-04-07 12:38:05,424 INFO [component_loader]
'fn_risk_fabric.components.set_event_classifications.FunctionComponent' loading

2018-04-07 12:38:05,425 INFO [component_loader]
'fn_risk_fabric.components.set_event_mitigations.FunctionComponent' loading

…

2018-04-07 12:38:05,435 INFO [actions_component]
'fn_risk_fabric.components.get_host_risk.FunctionComponent' function 'get_host_risk '
registered to 'risk_fabric_integration_functions'

2018-04-07 12:38:05,436 INFO [actions_component]
'fn_risk_fabric.components.get_ip_risk.FunctionComponent' function 'get_ip_risk '
registered to 'risk_fabric_integration_functions'

2018-04-07 12:38:05,437 INFO [actions_component]
'fn_risk_fabric.components.get_user_risk.FunctionComponent' function 'get_user_risk '
registered to 'risk_fabric_integration_functions'

2018-04-07 12:38:05,438 INFO [actions_component]
'fn_risk_fabric.components.get_risk_model_instances.FunctionComponent' function
'get_risk_model_instances ' registered to 'risk_fabric_integration_functions'

2018-04-07 12:38:05,439 INFO [actions_component]
'fn_risk_fabric.components.get_risk_model_instance_details.FunctionComponent' function
'get_risk_model_instance_details ' registered to 'risk_fabric_integration_functions'

2018-04-07 12:38:05,440 INFO [actions_component]
'fn_risk_fabric.components.get_action_plans.FunctionComponent' function
'get_action_plans ' registered to 'risk_fabric_integration_functions'

2018-04-07 12:38:05,441 INFO [actions_component]
'fn_risk_fabric.components.set_event_classifications.FunctionComponent' function
'set_event_classifications ' registered to 'risk_fabric_integration_functions'
```

```
2018-04-07 12:38:05,442 INFO [actions_component]
'fn_risk_fabric.components.set_event_mitigations.FunctionComponent' function
'set_event_mitigations ' registered to 'risk_fabric_integration_functions'

…

2018-04-07 12:38:05,729 INFO [actions_component] Subscribe to message destination
'risk_fabric_integration_functions'

…

2018-04-07 12:38:05,731 INFO [stomp_component] Subscribe to message destination
actions.<org id>.risk_fabric_integration_functions

…
```

Resilient Platform Configuration

In the Customization Settings section of the Resilient platform, you can verify that the following Risk Fabric specific message destination, functions, workflows and rules are available in the Resilient platform by clicking their respective tabs.

Message Destination

- **Risk Fabric Integration Functions –** Default Message Destination for the Risk Fabric Integration Functions

Integration Functions

- **RF Get Host Risk** – Query the Risk Rating Information for a hostname.
  - Inputs
    - **rf_hostname**: Hostname for a computer endpoint
  - **Outputs**: Risk Score for a computer endpoint.
- **RF Get IP Risk** – Query the Risk Rating information for an IP address.
  - Inputs
    - **rf_ipaddress**: IP Address (ex. 123.123.123.123)
  - **Outputs**: Risk Score for an IP Address.
- **RF Get User Risk** – Query the Risk Rating information for a username.
  - Inputs
    - **rf_username**: Username for a user account.
  - **Outputs**: Risk Score for a user.
- **RF Get Action Plans** – Query the set of action plans for an account.
  - **Inputs**: *none*
  - **Outputs**: A list of Action Plans, including the **rf_actionplanguid** for performing other actions like adding comments or updating event classifications and mitigations.
- **RF Get Risk Model Instances** – Query the set of Risk Model Instances.
  - Inputs
    - **rf_limit**: For limited how many risk model instances to pull

    - **Outputs**: A list of Risk Model Instances, including the **rf_riskmodelinstanceid** for performing other actions like classifications and mitigations.

- **RF Get Risk Model Instance Details** – Get the set of Event Scenarios for a Risk Model Instance.
    - Inputs
        - **rf_riskmodelinstanceid**: ID for the Risk Model Instance being requested
    - **Outputs**: Additional Details for a Risk Model Instance, including Event Scenarios and Entity Collections with their **rf_cardinstanceid** and **rf_focusentityid** for performing other actions like classifications and mitigations.
- **RF Set Classifications** – Update Event Classifications.
    - Inputs
        - **rf_riskmodelinstanceid**: ID for the Risk Model Instance being classified.
        - **rf_cardinstanceid**: ID for the Card Instance being classified.
        - **rf_focusentityid**: ID for the Focus Entity being classified.
        - **rf_actionplanguid**: ID for the action plan being classified.
    - **Outputs**: *none*
- **RF Set Mitigations** – Update Mitigation statues.
    - Inputs
        - **rf_riskmodelinstanceid**: ID for the Risk Model Instance being classified.
        - **rf_cardinstanceid**: ID for the Card Instance being classified.
        - **rf_focusentityid**: ID for the Focus Entity being classified.
        - **rf_actionplanguid**: ID for the action plan being classified.
    - **Outputs**: *none*

Example Workflows

- **RF Example: Get IP Risk** – Example workflow for getting an IP Risk Score. Used by the example rule with the same name to automatically assign Risk scores to IP Artifacts.
- **RF Example: Mitigate Persistent Insider Threats** – Example workflow for mitigating persistent insider threats. Add other integration functions like disabling users in LDAP and notifying managers to create fully automated mitigation process.

Example Rules

- **RF Example: Get IP Risk** – Example rule for automatically updating an IP Artifact's description field with the IP Address's Risk Score. This rule calls the Get IP Risk Workflow which uses the RF Get IP Risk Integration Function.

Example Scripts

- **create_incidents_action_plans.py** – Example script to create Incidents from RF Action Plans. Requires creating and configuring an Incident Type, ex. "Action Plan".
- **create_incidents_risk_models.py** – Example script to create Incidents from RF Risk Models. Requires creating and configuring an Incident Type, ex. "Risk Model".

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 creates progress information. Failures show up as errors and may contain Python trace statements.

Support

For additional support, contact support@baydynamics.com.

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