



# Incident Response Platform Integrations

# MITRE Integration Function V1.0.0

Release Date: March 2019

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 MITRE integration Function.

Overview

MITRE’s TAXIITM 2.0 Server provides Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK or ATTACK) content. This content is critical for cyber security industry in order to make a holistic approach to detection and mitigation of Advanced Persistent Threats (APTs).

MITRE Integration Function enables Resilient users to gather ATT&CK information on cyber intrusion once a tactic or technique has been identified. This information can help security analyst response quickly to a (potential) breach.

MITRE Integration includes 2 functions:

* Query ATT&CK information for a MITRE tactic
* Query ATT&CK information for a MITRE technique

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 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\_mitre\_integration-<*version*>.<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

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:

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

Deploy customizations to the Resilient platform

This package contains two functions, two custom fields, two data tables, three example workflows, and three rules.

|  |  |  |
| --- | --- | --- |
| **Function** | **Example Workflow** | **Rule** |
| MITRE tactic information | Example of getting MITRE tactic information | Get MITRE tactic information |
| MITRE technique information | Example of getting MITRE technique information | Get MITRE technique information |
| Example of adding MITRE technique task | Create Task for Technique |

The “Example of getting tactic information” updates both the “MITRE ATTACK of Incident” and “MITRE ATTACK techniques” data tables, while the “Example of getting technique information” populates the “MITRE ATTACK techniques” data table.

The “Create Task for Technique” is a data table rule for the “MITRE ATTACK techniques” data table. It creates a task for a selected technique.

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:

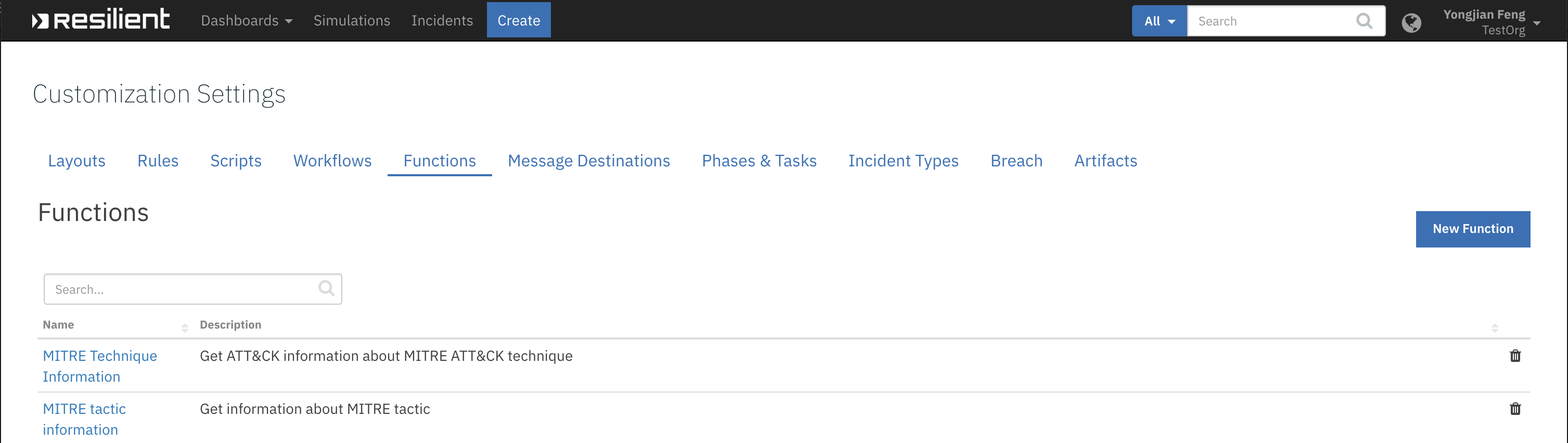
sudo systemctl resilient\_circuits [start|stop|restart|status]

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

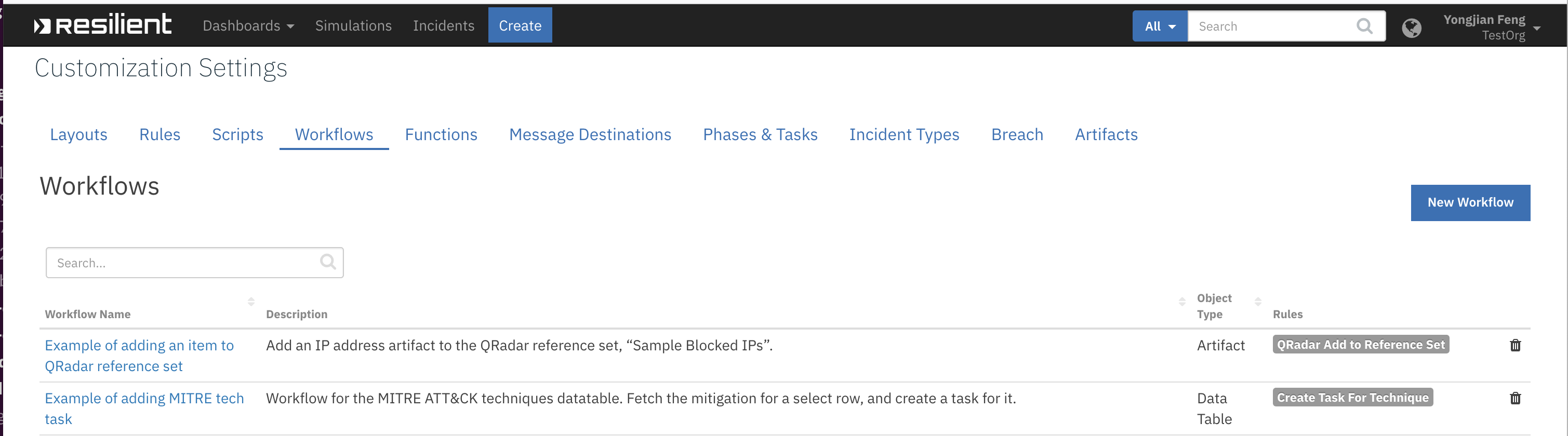
sudo journalctl -u resilient\_circuits --since "2 hours ago"

Function Descriptions

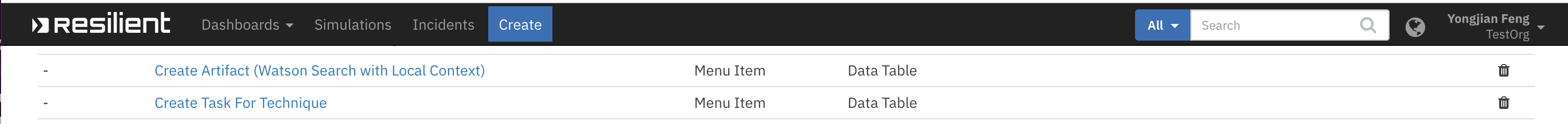
Once the function package deploys the function(s), you can view them in the Resilient platform Functions tab, as shown below.



The workflow can be viewed from the Workflows page. [Need to update screenshot with clean wfs]



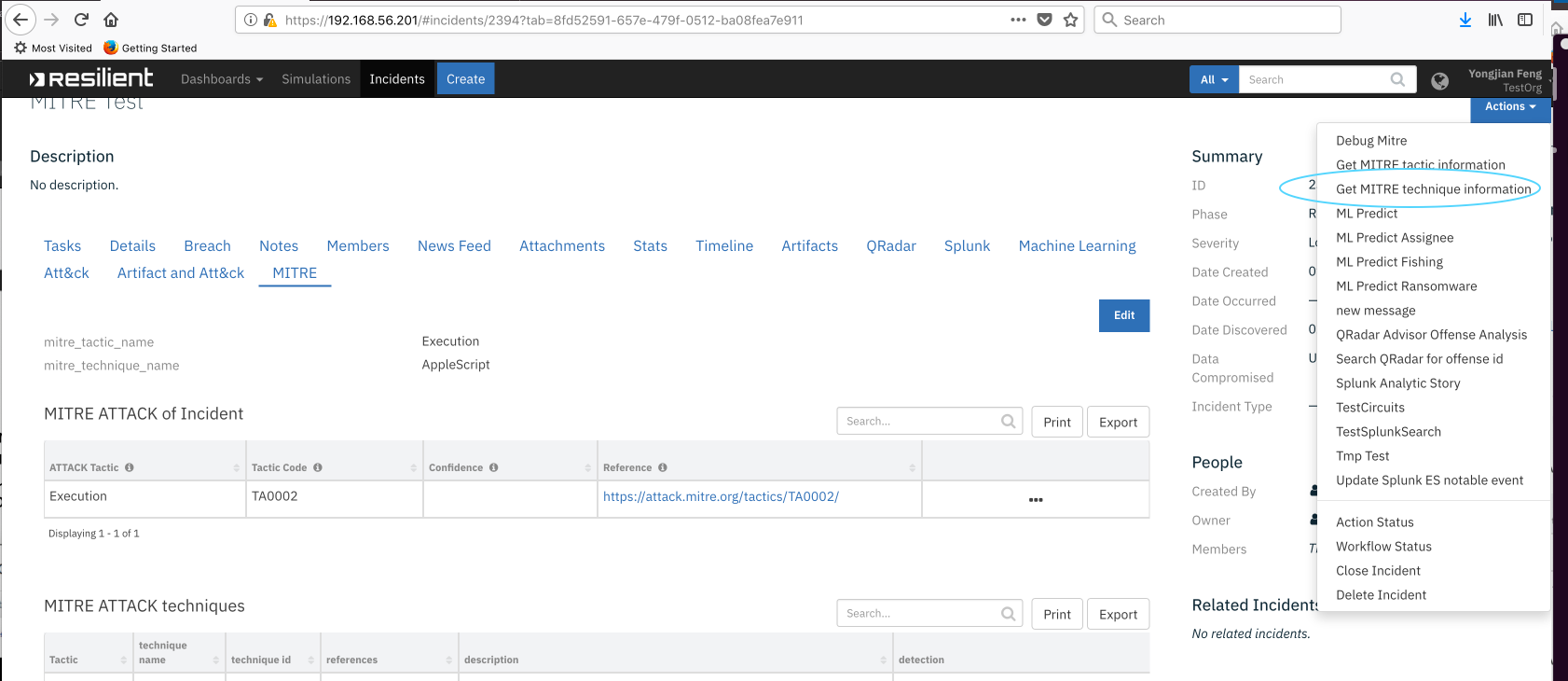
The rule that calls this workflow can be viewed from the Rules page. [Need to update screenshot with clean rules]



mitre\_technique\_information:MITRE technique information

This function retrieves ATT&CK information of the given MITRE technique from MITRE STIX TAXII server. Assume that the “MITRE” tab was created as shown above, enter a valid MITRE technique. For example, “AppleScript” can be used.

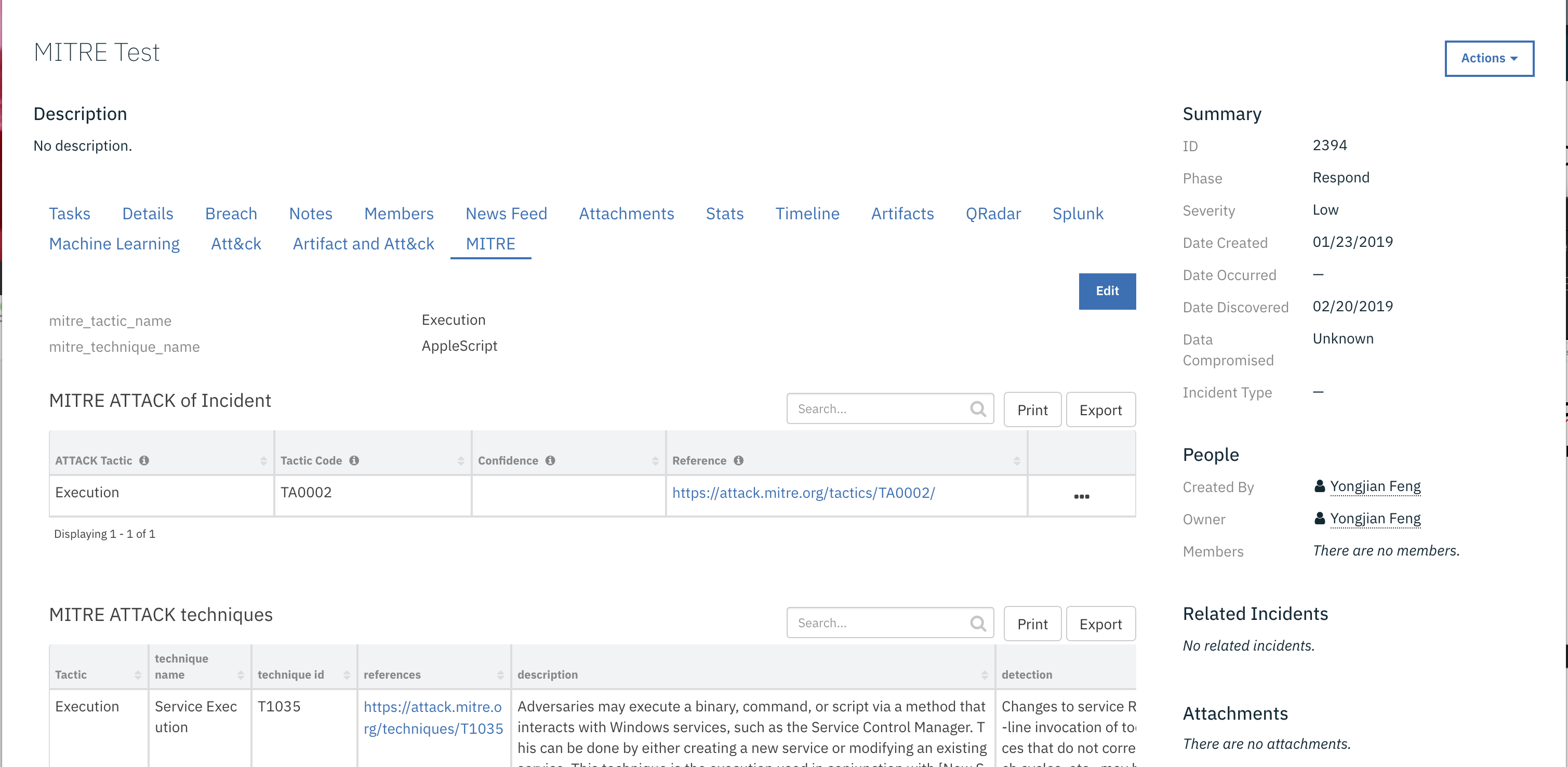
Now click Actions->Get MITRE technique information. This rule invokes the “Example of getting MITRE technique information” workflow, which calls the “MITRE technique information” function. The returned result is used by the workflow to populate the “MITRE ATTACK technique” data table.



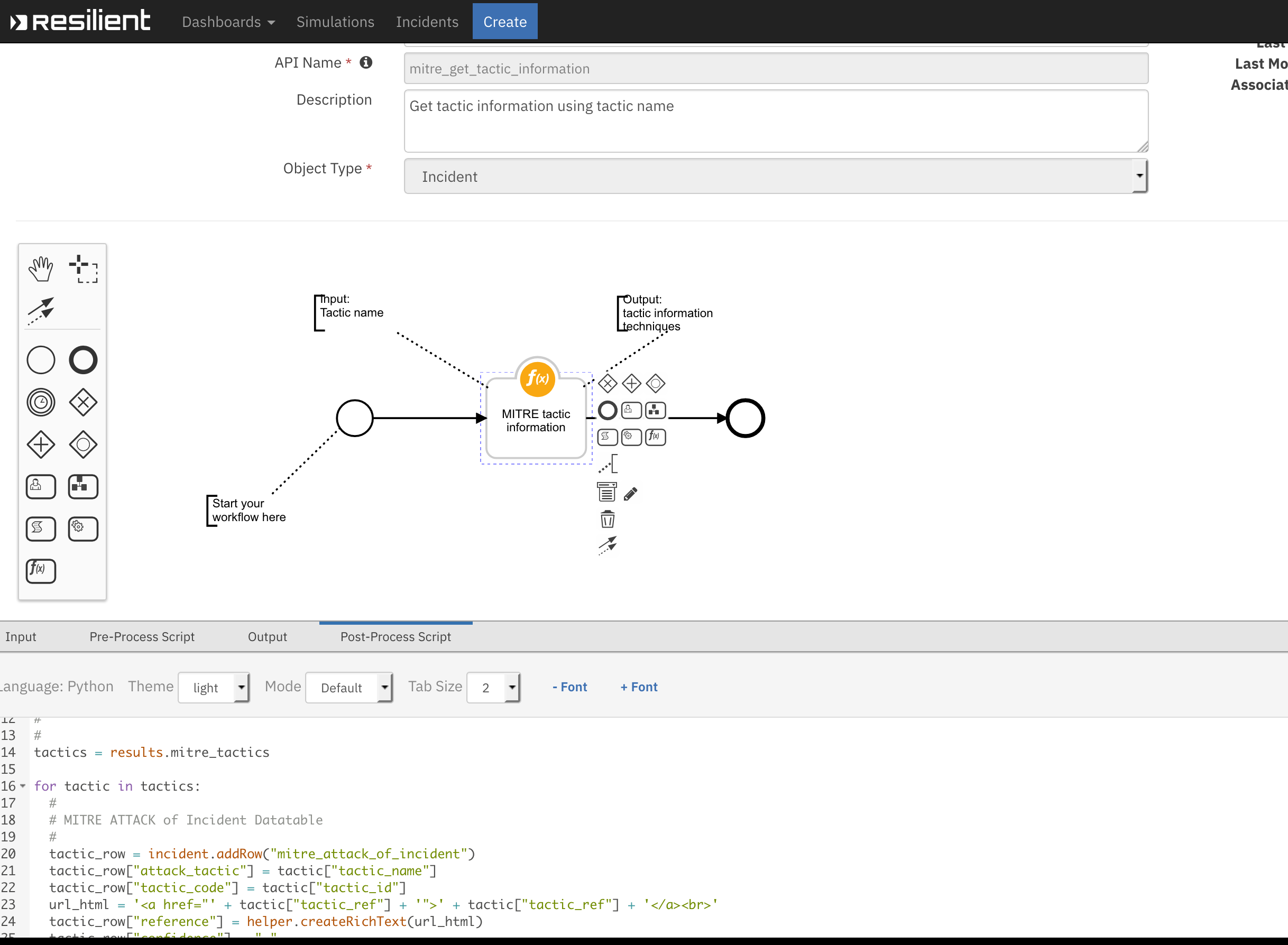
Note the “MITRE technique information” takes three inputs. Either the technique name or the technique id can be used to lookup the technique. If the flag “mitre\_technique\_mitigation\_only” is set to true, this function retrieves the mitigation information only. For some special cases as shown in the next function, only the mitigation is needed.

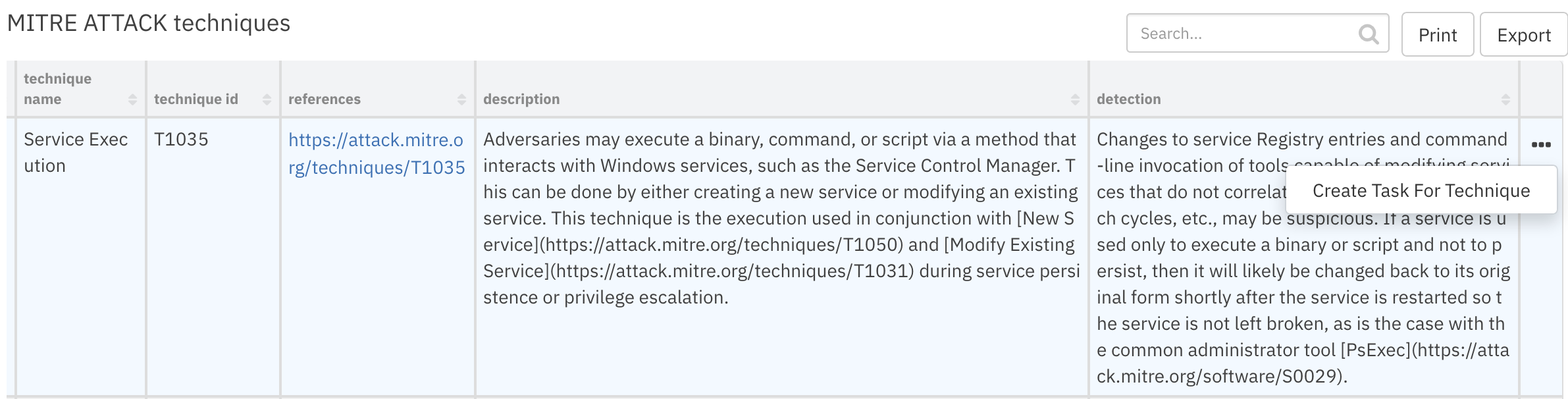
mitre\_tactic\_information:MITRE tactic information

This function retrieves ATT&CK information for give MITRE tactic(s). To use this function, user can create a “MITRE” tab, and put the two custom fields (mitre\_tactic\_name and mitre\_technique\_name) and the two data tables into this tab. Enter a valid MITRE tactic name. Here “Execution” is used.

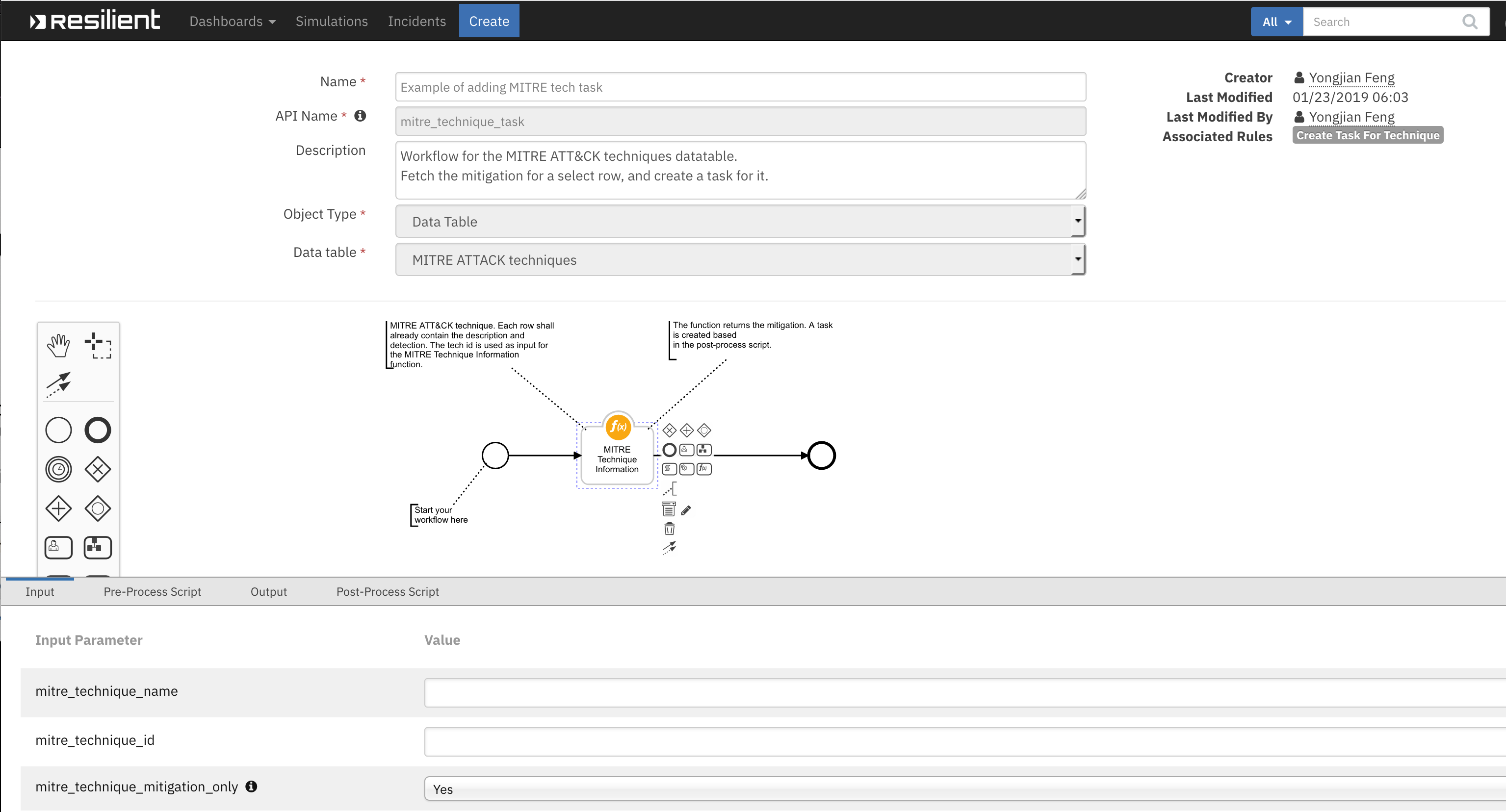


Now click Actions->Get MITRE tactic information. This rule invokes the “Example of getting MITRE tactic information” workflow, which calls the “MITRE tactic information” function. The function fetch information from the MITRE STIX TAXII server, and the returned result was used by the workflow to populate two data tables.

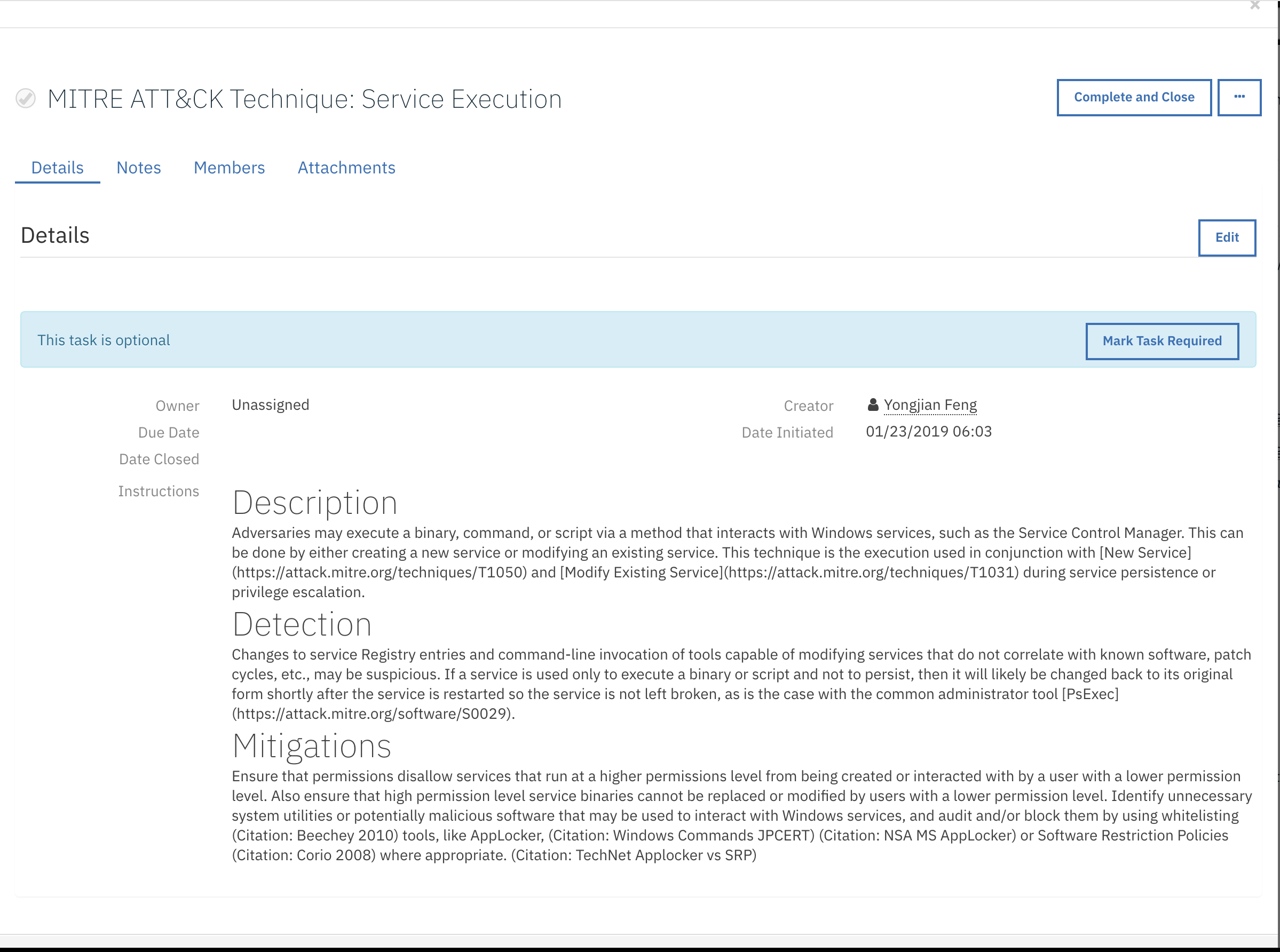


Note once a technique is added to the “MITRE ATTACK techniques” data table, user can further create a task to further investigate/mitigate the technique. 

The “Create Task For Technique” rule invokes the “Example of adding MITRE tech task” workflow. Note here most of the information of this technique is in the data table except the mitigation. The example workflow invokes the “MITRE technique information” function and set the “mitre\_technique\_mitigation\_only” to Yes. Thus the function retrieves only the mitigation for this technique.



The information about this technique is then used to create a task.



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.