

MISP Functions

Craig @ Resilient Labs

The MISP functions allows integration with MISP Threat Intelligence Platform from Resilient.








New for v3.0.1

- App Host support.
- Proxy support.

Use Cases

The following 6 use cases are supported:

- Create a MISP "Event" from a Resilient incident.
- Add attributes to the incident "Event" in MISP from incident artifacts incident.
- Mark any artifact in Resilient as "Sighted" if they exist in MISP.
- Search all MISP events for a match on a given attribute, this will also return tags for an attribute.
- Return all MISP sightings for a given event.
- Create Tag on an event or attribute in MISP - such as TLP, Att&ck or threat actor.

MISP Create Attribute	Create a MISP attribute from an incident artifact	
MISP Create Event	Create a MISP event from an incident	
MISP Create Sighting	Create a MISP sighting from an incident artifact	
MISP Create Tag	Creates a Tag	
MISP Search Attribute	Search MISP event attributes for a given match on an artifact	
MISP Sighting List	List all sightings associated with an event	
Scan with urlscan.io	Analyze a URL with urlscan.io	

Installation

App Host

All the components for running this integration in a container already exist when using the App Host app.

To install,

- Navigate to Administrative Settings and then the Apps tab.
- Click the Install button and select the downloaded file: app-fn_misp-x.x.x.zip.
- Go to the Configuration tab and edit the app.config file, editing the API key for MISP and making any additional setting changes.

Config	Required	Example	Description
misp_url	Yes	10.10.10.10:5000	IP or URL of the MISP instance along with the http port
misp_key	Yes	someAPIkey	API key to access the MISP API
verify_cert	Yes	True	Secure connection
https_proxy	No	https://your.proxy.com	https proxy for connecting to MISP
http_proxy	No	http://your.proxy.com	http proxy for connecting to MISP

Integration Server

Python2 vs. Python3

MISP has deprecated support for Python2 and does not guarantee functionality in a Python2 environment. Bearing this in mind, we recommend that you use `fn_misp` in Python3. However, the `fn_misp` is designed to function in both Python2 and Python3.

Installation

This function is packaged as a zip, so you do not need to extract the zip to install it.

```
pip install fn_misp-<version>.zip
```

You will then need to add the configuration section to your `app.config` file.

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

This will add the MISP key and url properties. Update the URL for your instance of MISP and add the API key found in "Event Actions -> Automation". Set the verify cert to false if you do not want to verify the certificate.

```
[fn_misp]
misp_url=http://localhost
misp_key=<your key>
# used to bypass cerification validation for self signed instances of MISP
verify_cert=true
# Optional: access MISP via an http/https proxy
#http_proxy=<http_proxy_server>
#https_proxy=<https_proxy_server>
```

To add all the configuration settings for functions, workflows, runs, etc. to Resilient, run the following command:

```
resilient-circuits customize -l fn-misp
```

Configuration

Sample workflows are included to demonstrate how to execute a function and how to parse the returned results.

For each workflow, edit each function (see the pencil icon) and visit the Post-Process Script for processing hints. Your use of the functions may need different data formatting or different feedback such as notes etc.

The screenshot shows the 'Customization Settings' page in the Resilient interface. The top navigation bar includes 'Dashboards', 'Inbox', 'Incidents', and 'Create'. The main content area is titled 'Customization Settings' and has tabs for 'Layouts', 'Rules', 'Scripts', 'Workflows', 'Functions', 'Message Destinations', 'Phases & Tasks', 'Incident Types', 'Breach', and 'Artifacts'. The 'Workflows' tab is selected, showing 'Example: MISP Create Event'. The workflow details include: Name (Example: MISP Create Event), API Name (example_misp_create_event), Description (Create a MISP event from an incident), and Object Type (Incident). A sidebar on the right shows the Creator (Integrations), Last Modified (09/22/2020 11:38), Last Modified By (Integrations), and Associated Rules (Example: Create MISP Event). Below the details is a visual workflow diagram showing an 'Input' node leading to a 'MISP Create Event' function, which then leads to an 'Output' node. A note indicates that the function returns the 'misp_event_id' for ongoing reference. At the bottom, there is a table for 'Input Parameter' and 'Value' with three rows: 'misp_event_name', 'misp_distribution', and 'misp_analysis_level'.

Input Parameter	Value
misp_event_name *	
misp_distribution ⓘ	
misp_analysis_level ⓘ	

An incident field `misp_event_id` is used to track event creation. This field can remain hidden or added to your layout through the [Customization Settings](#) section. You can pair this with a rich text field and in-product script to give a direct Resilient UI link to the event in MISP.

Rules

This app is not packaged with rules to prevent adding needed configuration to your system.

Some recommended rules are below:

- Auto - Create MISP Event if incident is created.
- Auto - Create MISP Attribute if artifact is created and incident.misp_event_id has a value.
- Auto - Search MISP Attribute if artifact is created and incident.misp_event_id has a value.
- Menu Item - Create TLP Tag on an artifact when misp_event_id has a value.

Using MISP for Att&ck

The following assumes you have installed the Resilient Att&ck function.

When you run a search for an attribute with event or attribute level artifacts. The response includes Technique IDs from MITRE Att&ck.

You will see this is in the default search workflow, the Att&ck information comes back in the tags field in the artifact description.

We can use our postprocessing script to extract these to use in the Techniques DataTable for Att&ck.

Clone the example workflow to customize it.

```
resilient-circuits clone --workflow example_misp_search_attribute misp_search_attack
```

Now we can edit the post process script of the cloned workflow.

First we need to loop through the tags and check they have **mitre-attack** in, add to the bottom of the existing post processing script.

```
for tag in results.tags:  
    if "mitre-attack" in tag:
```

Next we need to parse out the Technique ID to put in our MITRE Att&ck table (from the MITRE function).

```
for tag in results.tags:  
    if "mitre-attack" in tag:  
        tag.split("=")
```

This will give us a list now where the second (position 1) item is "Bootkit - T1067", we can now use simple regex to pull this out.

```
import re  
  
regex_pattern = "(T\d+)"  
  
for tag in results.tags:  
    if "mitre-attack" in tag:  
        split_tag = tag.split("=")  
        technique = split_tag[1]  
        technique_id = re.findall(regex_pattern, technique)  
        dt = incident.addRow("mitre_attack_techniques")  
        dt.technique_id = technique_id[0]
```

Now create a rule and run the workflow to check it pulls out the Technique ID into the table.

Now this is all working you can chain with the MITRE Lookup function to lookup the Technique ID when a row is added to the datatable. Remember you can also use the "Create Tag" function to tag artifacts with Att&ck.