IBM Resilient

Incident Response Platform Integrations

Floss Function V1.0.0

Release Date: July 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 Floss Function.

Overview

This Resilient Function package provides a function fn_floss that takes a binary file as input and returns a list of decoded obfuscated strings from the file.

Included in the package are two example workflows that use the fn_floss function:

- Example: Floss: Artifact Input

- Example: Floss: Attachment Input

Both example workflows create a task or incident note containing the list of decoded strings extracted from the file.

Also included in the package are example rules for creating the floss function menu items.

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.”x”, where “x” is 10 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. To use the Floss function package, you must first install 3rd party python packages vivisect and Floss:

```
pip install https://github.com/williballenthin/vivisect/zipball/master
pip install https://github.com/fireeye/flare-floss/zipball/master
```

3. Run the following command to install the Floss Function package:

```
sudo pip install --upgrade fn_floss-1.0.0.zip
```

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_floss] section, edit the settings as follows:

```
[fn_floss]

# Floss Function

# Use the following floss_options variable to specify the commandline
options to be used by

# the floss package to define the behavior for extracting strings.

# Each commandline parameter should be separated by a comma.

# The defaults here are: -q quiet mode, -s shellcode, -n minimum string
length
```

```
# See https://github.com/fireeye/flare-floss/blob/master/doc/usage.md for
all possible commandline options.

floss_options=-q,-s,-n 5
```

Deploy customizations to the Resilient platform

This Resilient Function package provides a function fn_floss, two example workflows that invoke the fn_floss function, a message queue and rules for creating the the fn_floss menu item.

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
```

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:

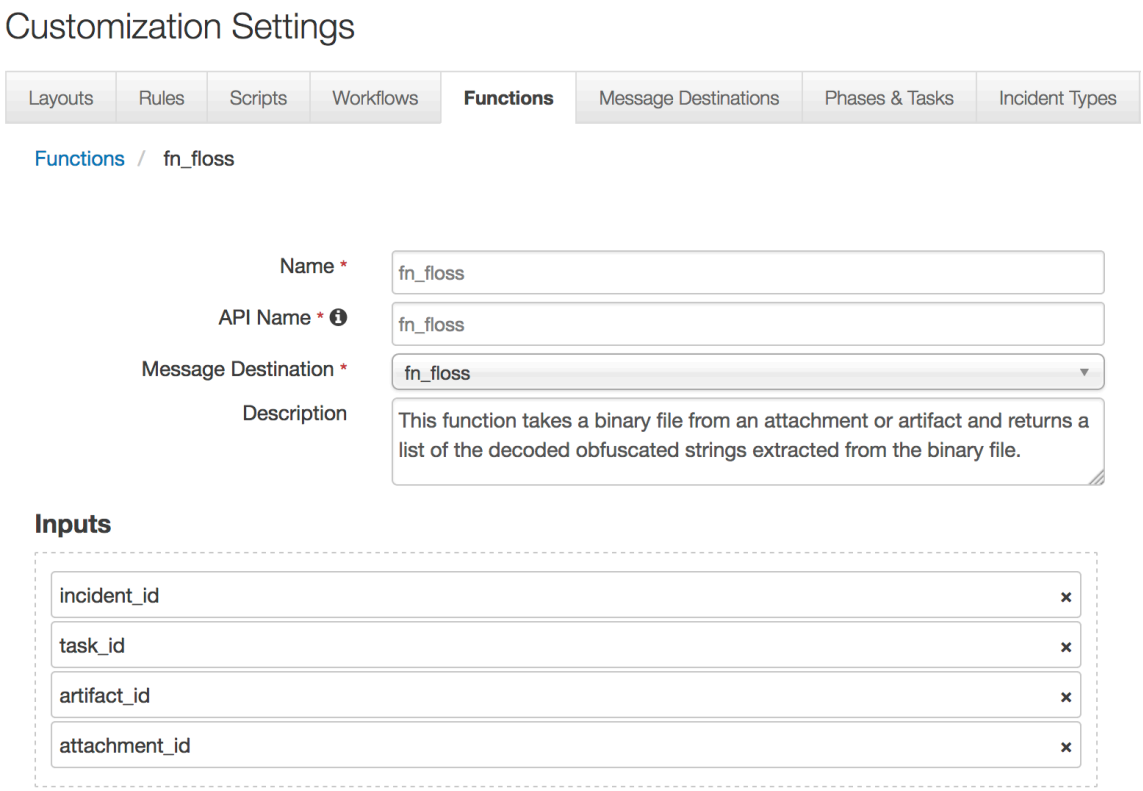
```
sudo journalctl -u resilient_circuits --since "2 hours ago"
```

Function Descriptions

Once the function package deploys the function, 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.

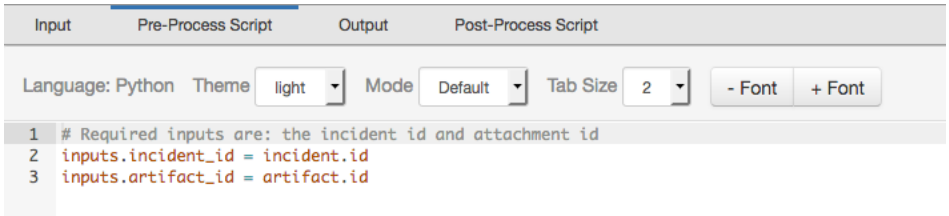
Fn_floss :

The Resilient Function fn_floss takes a binary file from an attachment or artifact and returns a list of the decoded obfuscated strings extracted from the binary file. A user may want to use fn_floss to search for strings encoded in possible malware files.

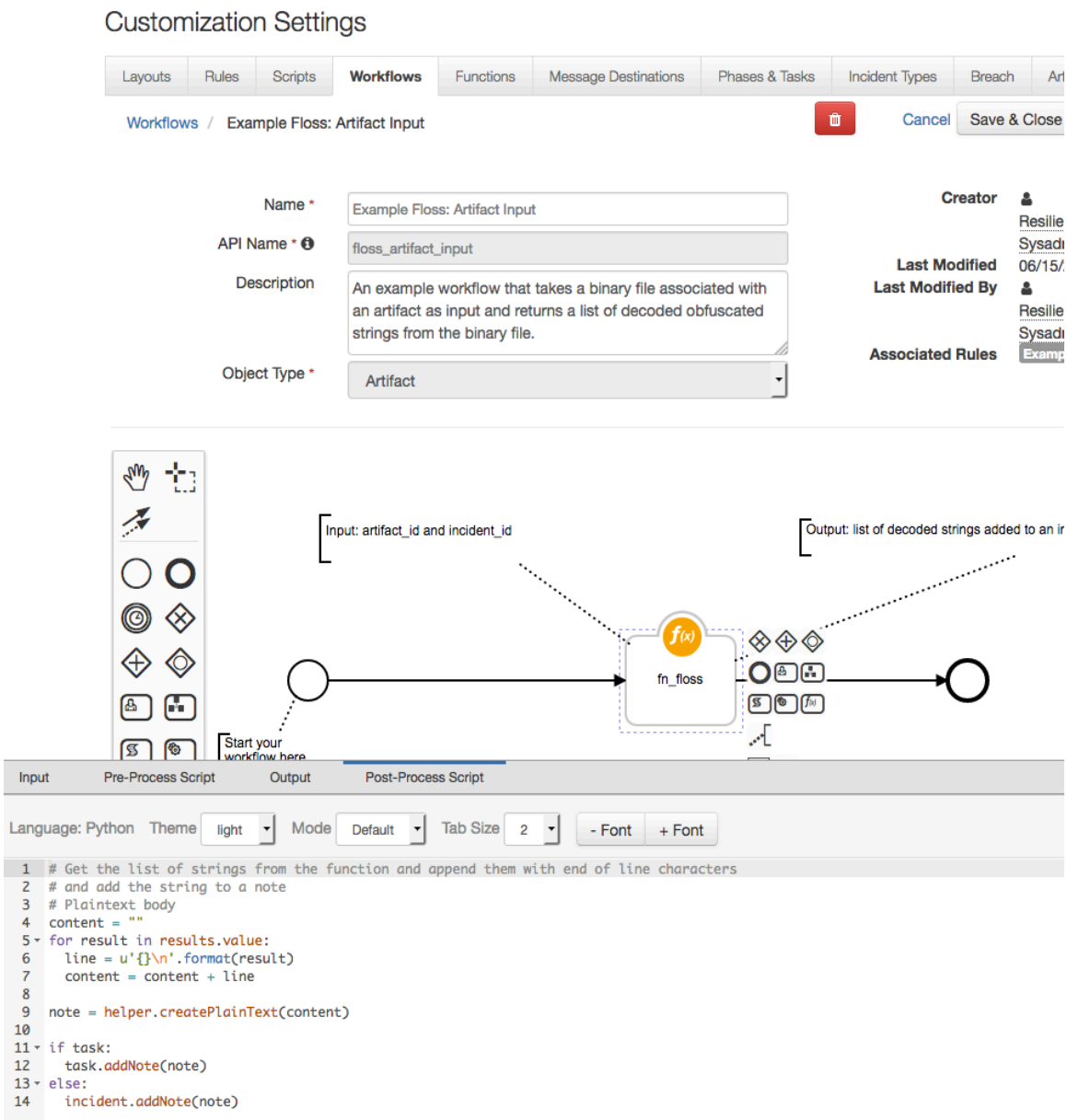


Example: Floss: Artifact Input Workflow:

An artifact_id and the associated incident_id are passed to the workflow in the pre-processor script.

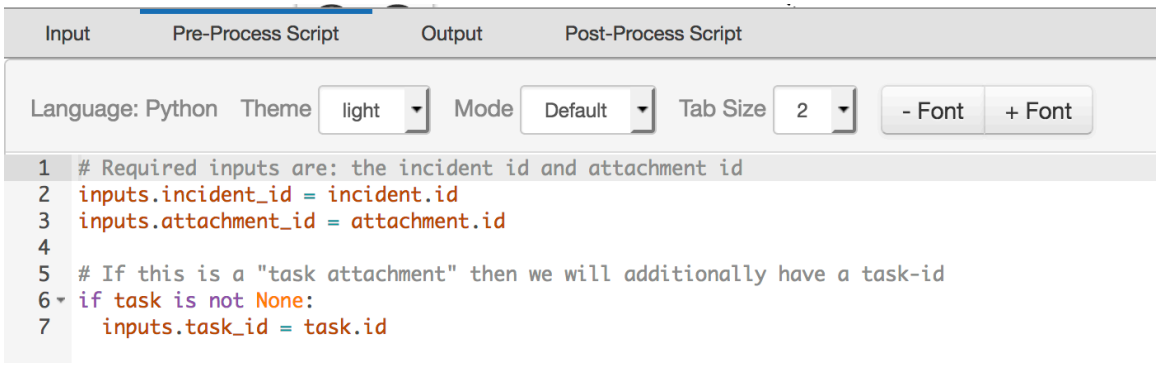


The screen shot below shows an example workflow with an artifact as input and the post-processor script that retrieves the list of strings from the fn_floss function and adds them to a note associated with the incident:

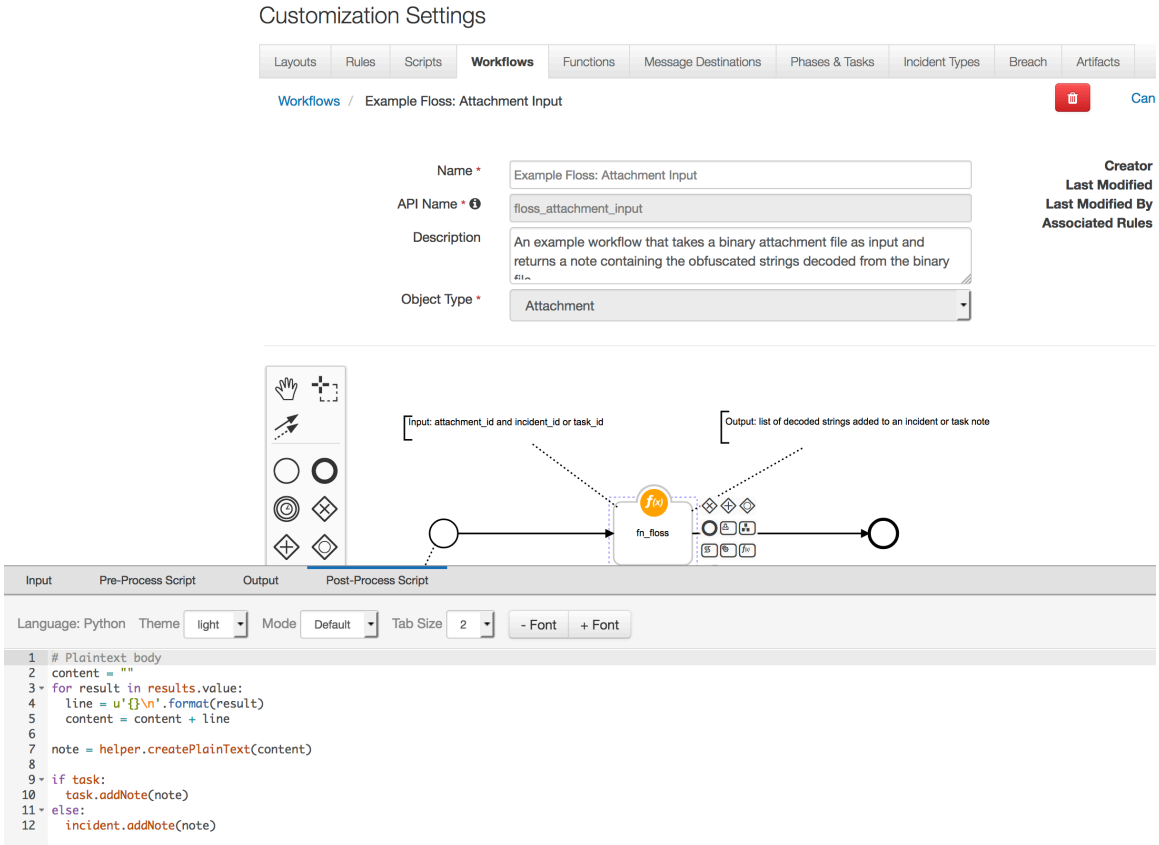


Example: Floss: Attachment Input Workflow:

An attachment_id and the associated incident_id or task_id are passed to the example workflow in the pre-processor script.



The screen shot below shows the example workflow with an attachment as input and the post-processor script that retrieves the list of strings from the fn_floss function and adds them to a note associated with the incident or task:



Resilient Platform Configuration

The Resilient fn_floss package uses an open source package called FLOSS to decode the obfuscated strings from a binary file. Refer to the FLOSS usage documentation: https://github.com/fireeye/flare-floss/blob/master/doc/usage.md for all possible command line options that can be configured in the Resilient app.config file.

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.