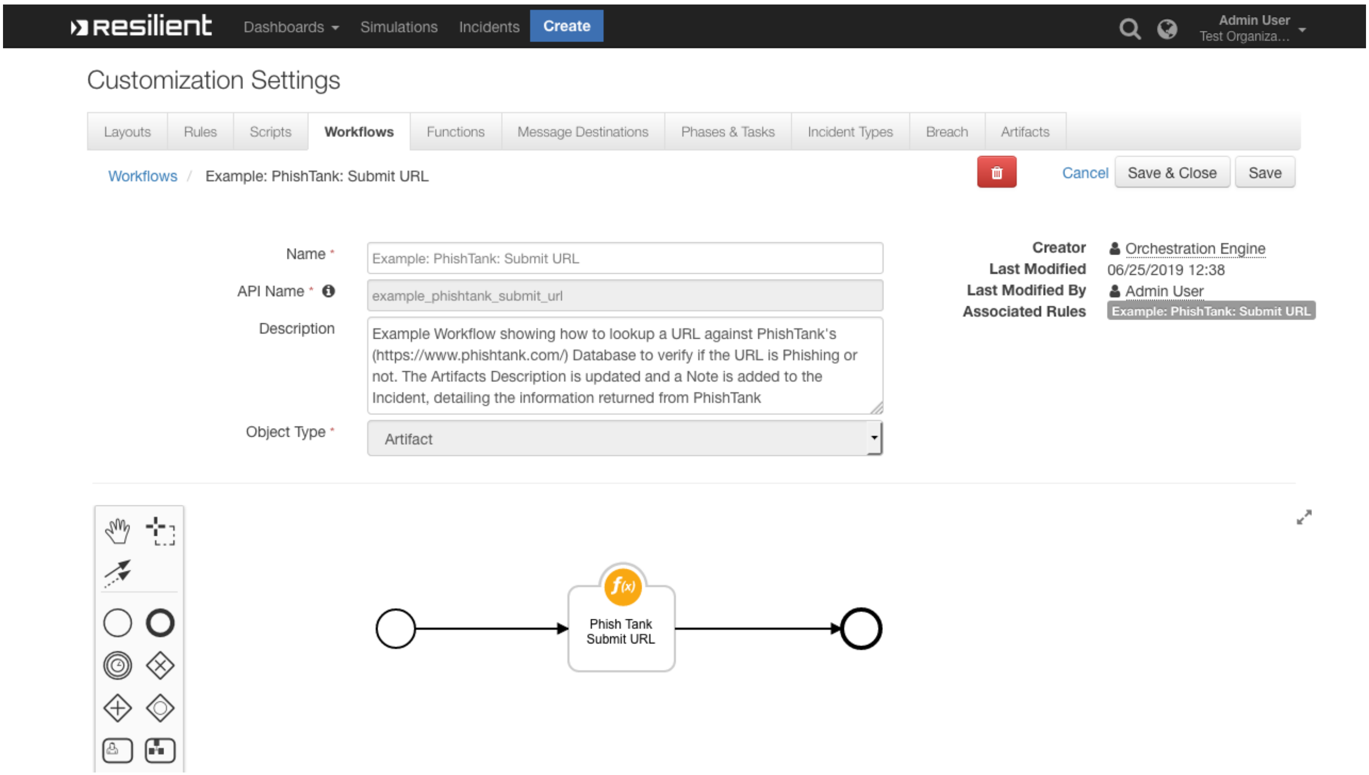
PhishTank Lookup Function for IBM Resilient

Overview

PhishTank Lookup URL Function for IBM Resilient



Customization Settings

Workflows / Example: PhishTank: Submit URL

Name * Example: PhishTank: Submit URL

API Name * example_phishtank_submit_url

Description Example Workflow showing how to lookup a URL against PhishTank's (https://www.phishtank.com/) Database to verify if the URL is Phishing or not. The Artifacts Description is updated and a Note is added to the Incident, detailing the information returned from PhishTank

Object Type * Artifact

Creator Orchestration Engine
Last Modified 06/25/2019 12:38
Last Modified By Admin User
Associated Rules Example: PhishTank: Submit URL

Lookup a URL against PhishTank's (https://www.phishtank.com/) Database to verify if the URL is Phishing or not. The Artifacts Description is updated and a Note is added to the Incident, detailing the information returned from PhishTank

Note on Partial URL Support

Due to the differences in implementation of PhishTank's API and the functionality behind their online User Interface, partial URLs are not supported with this Integration

- The UI matches part of the provided URL
- Whereas the API only matches the full URL

For example:

URL to Check	PhishTank UI Result	PhishTank API Result
`https://safrainternet-br.com/Atualizar.html`	`https://safrainternet-br.com/Atualizar.html is a Valid Phish`	`https://safrainternet-br.com/Atualizar.html is a Valid Phish`
`https://safrainternet-br.com/`	`https://safrainternet-br.com/Atualizar.html is a Valid Phish`	`https://safrainternet-br.com/ not found in Database`

Requirements

- IBM Resilient >= `v31.0.4254`
- An Integration Server running `resilient_circuits>=30.0.0`
  - To setup an Integration Server see: ibm.biz/res-int-server-guide

Installation

- Download the `fn_phish_tank.zip`
- Copy the `.zip` to your Integration Server and SSH into it.
- **Unzip** the package:

```
$ unzip fn_phish_tank-x.x.x.zip
```

- **Change Directory** into the unzipped Directory:

```
$ cd fn_phish_tank-x.x.x
```

- **Install** the package:

```
$ pip install fn_phish_tank-x.x.x.tar.gz
```

- Import the **configurations** into your app.config file:

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

- Import the fn_phish_tank **customizations** into the Resilient Appliance:

```
$ resilient-circuits customize -y -l fn-phish-tank
```

- Open the config file, scroll to the bottom and edit your fn_phish_tank **configurations**:

```
$ nano ~/.resilient/app.config
```

Config	Required	Example	Description
phish_tank_api_url	Yes	`http://checkurl.phishtank.com/checkurl/`	PhishTank API Access URL
phish_tank_api_key	Yes	-	PhishTank API Key
proxy	Yes	`127.0.0.1`	Proxy Server Address. By default it will be `None`

- **Save** and **Close** the app.config file.

- [Optional]: Run selftest to test the Integration you configured:

```
$ resilient-circuits selftest -l fn-phish-tank
```

- **Run** resilient-circuits or restart the Service on Windows/Linux:

```
$ resilient-circuits run
```

Uninstall

- SSH into your Integration Server
- **Uninstall** the package:

```
$ pip uninstall fn-phish-tank
```

- Open the config file, scroll to the [fn_phish_tank] section and remove the section or prefix `#` to comment out the section.
- **Save** and **Close** the 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

Name	Version	Author	Support URL
fn_phish_tank	1.0.0	Resilient Labs	http://ibm.biz/resilientcommunity

User Guide: fn_phish_tank_v1.0.0

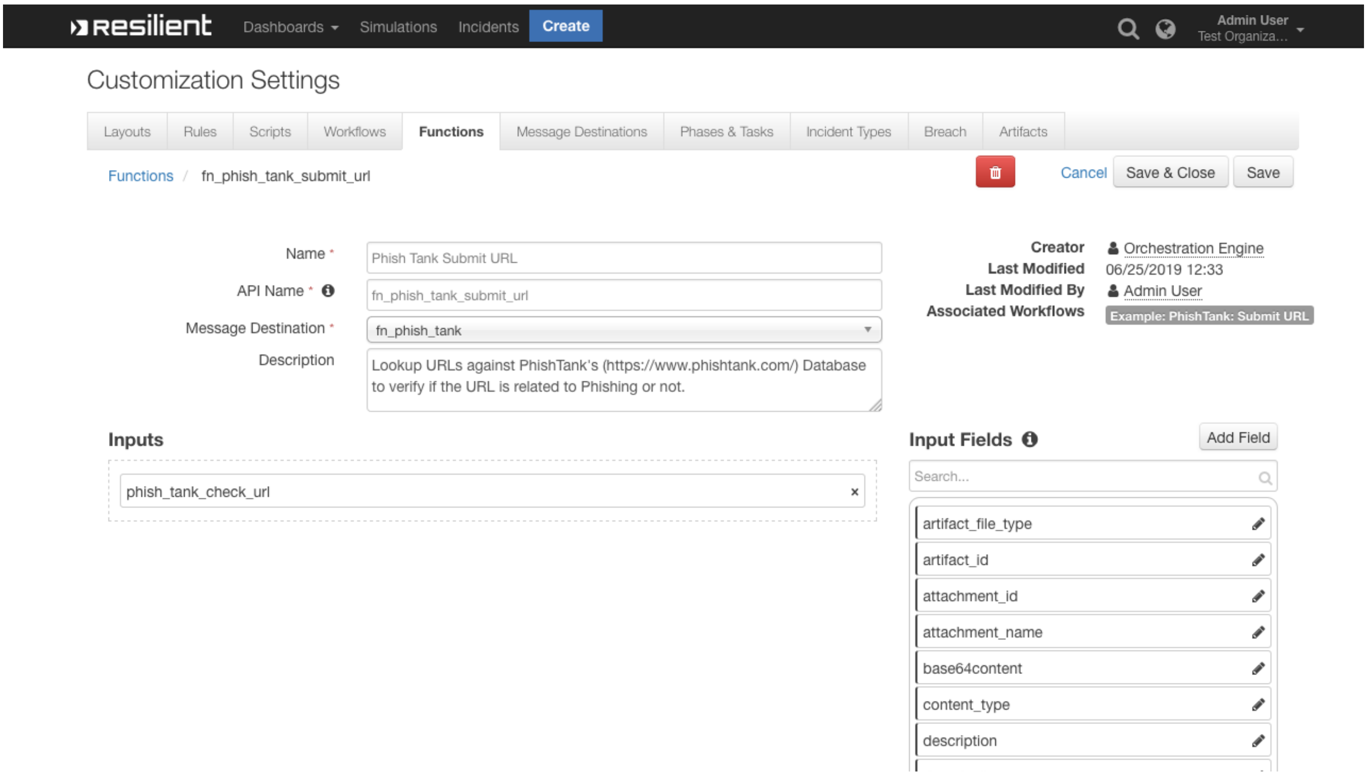
Table of Contents

Key Features

- Verify if a URL is related to Phishing using PhishTank's Database
- Add a Note to the Incident and append to the Artifact's Description information detailing the results received from PhishTank

Function - Phish Tank Submit URL

Lookup URLs against PhishTank's (https://www.phishtank.com/) Database to verify if the URL is related to Phishing or not.



Name * Phish Tank Submit URL

API Name * fn_phish_tank_submit_url

Message Destination * fn_phish_tank

Description Lookup URLs against PhishTank's (https://www.phishtank.com/) Database to verify if the URL is related to Phishing or not.

Creator Orchestration Engine

Last Modified 06/25/2019 12:33

Last Modified By Admin User

Associated Workflows Example: PhishTank: Submit URL

Inputs

phish_tank_check_url

Input Fields

- artifact_file_type
- artifact_id
- attachment_id
- attachment_name
- base64content
- content_type
- description

▶ Inputs:

Name	Type	Required	Example	Tooltip
`phish_tank_check_url`	`text`	Yes	`http://www.example.com`	URL to lookup in PhishTank's Database

▶ Outputs:

```
results = {
    'inputs': {
        u 'phish_tank_check_url': u 'http://barea-v02.ga/df/'
    },
    'metrics': {
        'package': 'fn-phish-tank',
        'timestamp': '2019-06-25 09:53:11',
        'package_version': '1.0.0',
        'host': 'shanes-mbp.galway.ie.ibm.com',
        'version': '1.0',
        'execution_time_ms': 192
    },
    'success': True,
    'content': {
        u 'meta': {
            u 'status': u 'success',
            u 'timestamp': u '2019-06-25T08:53:11+00:00',
            u 'serverid': u 'df0ef05',
            'timestamp_modified': 1561452791000,
            u 'requestid': u '172.31.97.117.5d11e0f7b69ce3.54218678'
        },
        u 'results': {
            u 'verified': True,
            'verified_at_modified': 1561452674000,
            u 'phish_detail_page': u
'http://www.phishtank.com/phish_detail.php?phish_id=6093832',
            u 'url': u 'http://barea-v02.ga/df/',
            u 'verified_at': u '2019-06-25T08:51:14+00:00',
            u 'phish_id': u '6093832',
            u 'valid': True,
            u 'in_database': True
        }
    },
    'raw': '{"meta": {"status": "success", "timestamp": "2019-06-
25T08:53:11+00:00", "serverid": "df0ef05", "timestamp_modified":
1561452791000, "requestid": "172.31.97.117.5d11e0f7b69ce3.54218678"},
"results": {"verified": true, "verified_at_modified": 1561452674000,
"phish_detail_page": "http://www.phishtank.com/phish_detail.php?
phish_id=6093832", "url": "http://barea-v02.ga/df/", "verified_at": "2019-
06-25T08:51:14+00:00", "phish_id": "6093832", "valid": true,
"in_database": true}}',
    'reason': None,
    'version': '1.0'
}
```

▶ Example Pre-Process Script:

```
# Get the url from the Artifact's Value
inputs.phish_tank_check_url = artifact.value
```

▶ Example Post-Process Script:

```
def append_artifact_description(the_artifact, the_text):
  """Appends the_text to the_artifact.description safely
  handling unicode"""

  new_description = u""

  if the_artifact.description is None:
    current_description = None
  else:
    current_description = the_artifact.description.get("content", None)

  if current_description is not None:
    new_description = u"{0}<br>---<br>
{1}".format(unicode(current_description), unicode(the_text))

  else:
    new_description = u"{0}".format(unicode(the_text))

  the_artifact.description = helper.createRichText(new_description)


if results.success:

  # Get the PhishTank Results
  phish_tank_results = results.content.get("results", {})
  url = phish_tank_results.get("url", u"")
  in_database = phish_tank_results.get("in_database", False)
  is_verified = phish_tank_results.get("verified", False)
  is_valid = phish_tank_results.get("valid", False)

  # Define the comment and msg to be appended to the Artifact's
Description
  comment = u""
  msg = u"""<b>PhishTank Lookup</b> has complete
          <br><b>URL:</b> {0}</b>
          <br><b>Found in Database:</b> {1}""".format(url,
unicode(in_database))

  if not in_database:
    comment = u"Nothing known about this url"

  else:
    phish_id = phish_tank_results.get("phish_id")
    phish_detail_page_url = phish_tank_results.get("phish_detail_page")

    msg = u"""{0}
```

```
          <br><b>Phish ID:</b> {1}
          <br><b>Valid Phish:</b> {2}
          <br><b>Verified:</b> {3}
          <br><b>Link to PhishTank: <a href={4}>{4}</a></b>""".format(msg, phish_id, u"Yes" if is_valid else u"No", u"Yes" if is_verified else "No", phish_detail_page_url)

    if is_verified and is_valid:
      comment = u"Verified: Is a phishing site"

    elif is_verified and not is_valid:
      comment = u"This site is not a phishing site"

    elif not is_verified:
      comment = u"This url has not been verified"

  msg = u"""{0}<br><br><b>Comment:</b> {1}""".format(msg, comment)

  append_artifact_description(artifact, msg)
  incident.addNote(helper.createRichText(msg))
```

Rules

Rule Name	Object	Workflow Triggered
Example: PhishTank: Submit URL	artifact	`example_phishtank_submit_url`