IBM Security

IBM

Resilient SOAR Platform
Resilient Machine Learning
Natural Language Processing
Function User Guide
V1.0

Date: April 2020

Resilient SOAR Platform
Resilient Machine Learning Function
User Guide

Platform Version	Publication	Notes
1.0	April 2020	Initial publication.

Table of Contents

Overview

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 is a user guide for the Resilient Machine Learning Function (fn_machine_learning_nlp). It is a companion guide to the *Resilient Machine Learning Function Reference Guide*.

Resilient Machine Learning function supports Nature Language Processing (NLP). NLP is used to digest textual data of incidents and provide advanced predictions of incident relationships. This tool is integrated with Resilient platform and customized for Resilient users.

This integration provides the followings:

- A command line tool to build an NLP model
- A search function that ranks incidents according to similarity

Together with the above, this package also includes an example workflow that demonstrates how to call the function above, two rules that starts the workflow, and a custom datatable the function uses to write the results.

Installation

Before installing, verify that your environment meets the following prerequisites:

- Resilient platform is version 32 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 3.5 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. Run the following command to install pandas:

   ```
   sudo pip install pandas-0.23.4-cp27-cp27m-linux_x86_64.whl
   ```

   This step is necessary because there is no wheel for pandas on Redhat Enterprise 7 available from pip. Also Redhat Enterprise 7 does not have gcc/g++ installed, so pip cannot build it from source code. A wheel was built locally, using the source code of https://files.pythonhosted.org/packages/e9/ad/5e92ba493eff96055a23b0a1323a9a803af71ec859ae3243ced86fcbd0a4/pandas-0.23.4.tar.gz. This is the same source code as the "pip install pandas" downloads.

1. To install the package, you must first unzip it then install the package as follows:
2. sudo pip install --upgrade fn_machine_learning_nlp-1.0.0.tar.gz
3. Configure the Python components
4. 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.
5. Complete the following steps to configure and run the integration:
6. Using sudo, switch to the integration user, as follows:
7. sudo su - integration
8. 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.
9. resilient-circuits config -c or resilient-circuits config -u

1. 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_machine_learning_nlp] section, edit the settings as follows. Note that the first one is required. The other two already have default values.

```
model_path=path to the folder you are going to save your model files
num_features=50
```

Deploy customizations to the Resilient platform

This package contains one function definitions and includes one example workflow and a rule that runs this function.

1. Use the following command to deploy these customizations to the Resilient platform:

```
resilient-circuits customize
```

2. 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]
```

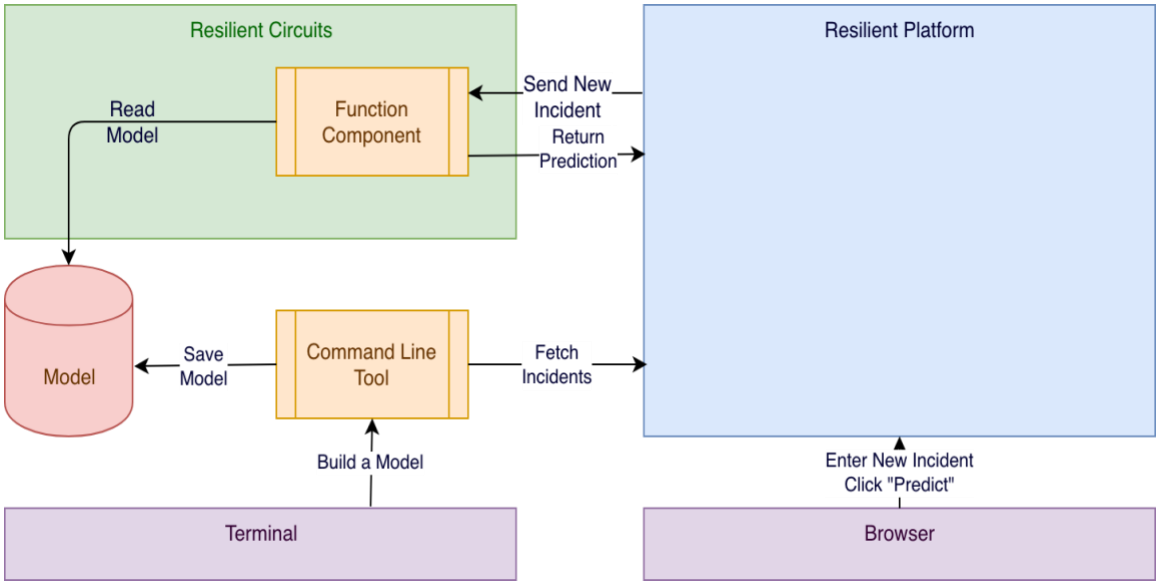
You can view log files for systemd and the resilient-circuits service using the journalctl command, as follows:

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

Function Descriptions

This integration package contains two components.

- A command line component to build an NLP model.
- A function component to compute and return incident similarity using the save NLP model above.



As shown in the above graph, users use the command line tool from a terminal to build an NLP model. The command line tool connects to the specified Resilient server to fetch incidents and artifacts. These are the samples used to train an NLP model. The model and saves it into files.

The function component takes an incident ID. It retrieves the incident using the ID, and loads the saved model from the folder specified in app.config. It can then use the model to compute similarities for this incident. This is triggered by a menu item in the incident page.

In summary, there are two processes, one to build an NLP model, the other to compute similarities using a model.

Build a Machine Learning Model

The command line tool is used to build an NLP model. It contains 2 subcommands.

build_nlp

This subcommand is used to build an NLP model. It downloads incidents and artifacts from a Resilient platform, and then save them into two CSV files. Then it uses the data to build an NLP model. Those two CSV files will be deleted once a model is built. The Resilient platform is specified in the app.config file. The subcommand has the following parameters:

-i	Optional. CSV file with incident data. Download incidents if this is absent
-a	Optional. CSV file with artifact data. Download artifacts if this is absent
-o	Optional. Model name. Default to be “resilient” if absent

Example:

```
res-ml build_nlp
```

Note, by default, an NLP model is saved into the following 4 files in the folder you run this command:

- resilient-w2v.txt
- resilient-sif.pkl
- resilient-pca.json
- resilient-vec.json

view

This subcommand is used to view basic information of one of those 4 files above.

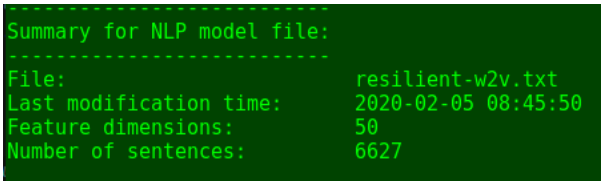
It takes one flag:

-i Required. File name

Example:

```
res-ml view -i resilient-w2v.txt
```

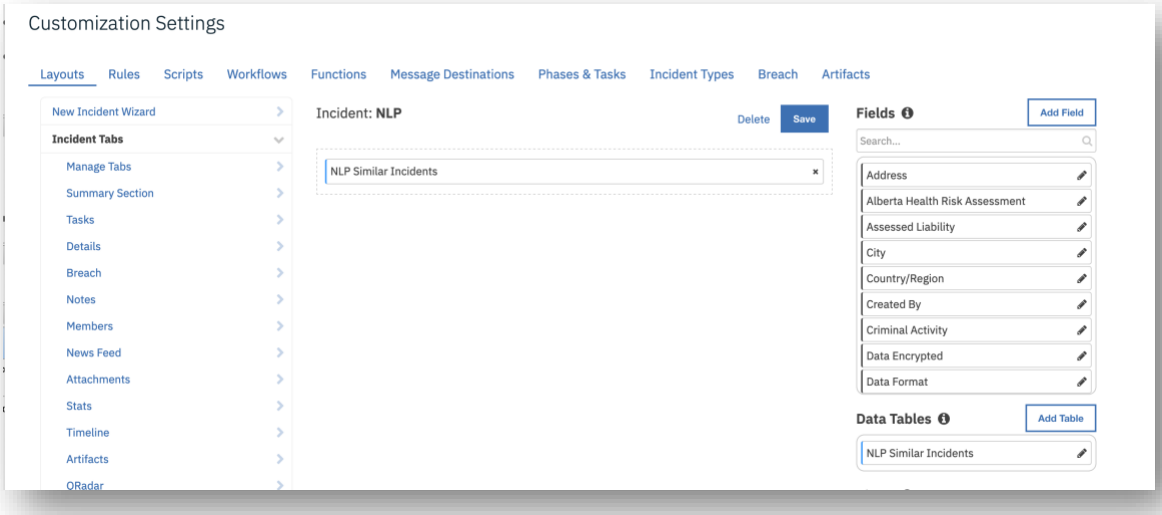
It shows the summary of the saved model, including last modification time, number of features (number of dimensions), and sample count.



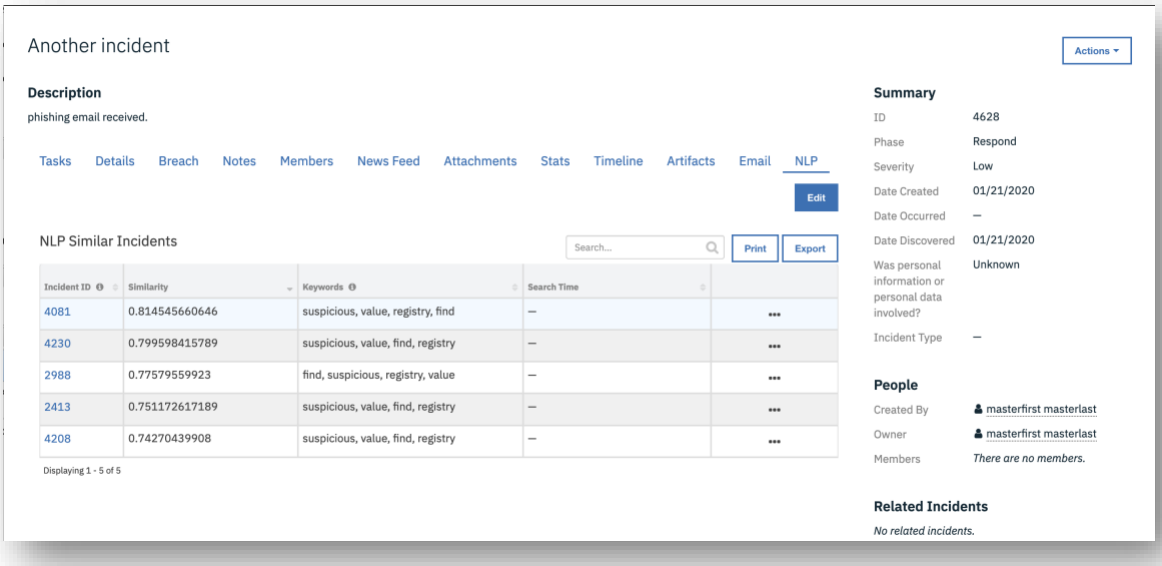
Use an NLP model to do search

Once a machine model is built and saved, it can be used to find similar incidents by the function component. The function component locates the saved model to use by looking at “model_path” is set in app.config. Those 4 files list above for a saved NLP model need to be in this folder.

The sample workflow writes the result into a custom datatable call “NLP Similar Incidents”. To show this in the incident page, add this customer datatable to it. You can go to Customization Settings page and create a new tab call “NLP”. Then add this custom datatable into the newly created tab.

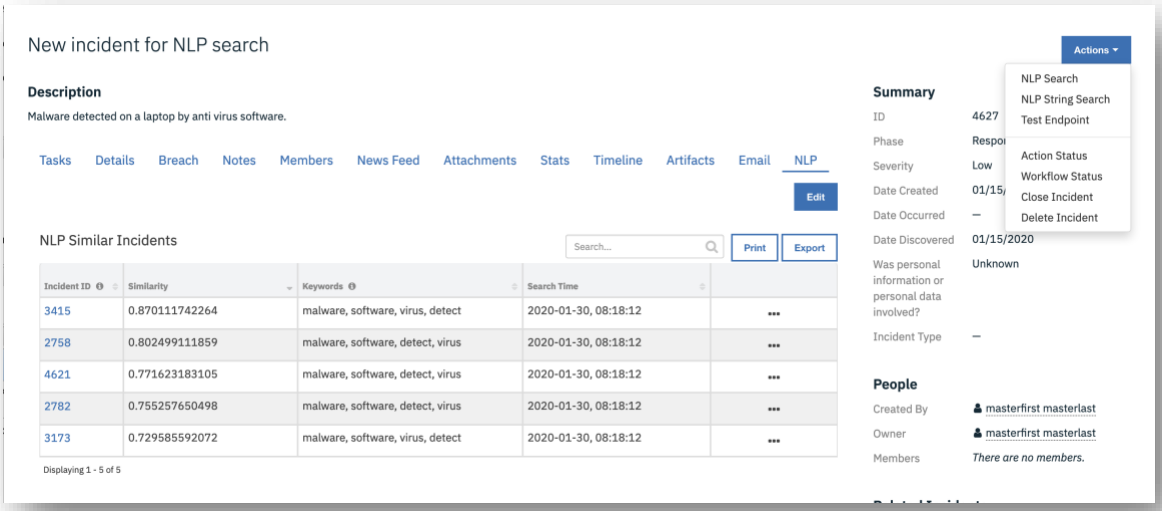


Then this new tab and the custom field is shown in the incident page.



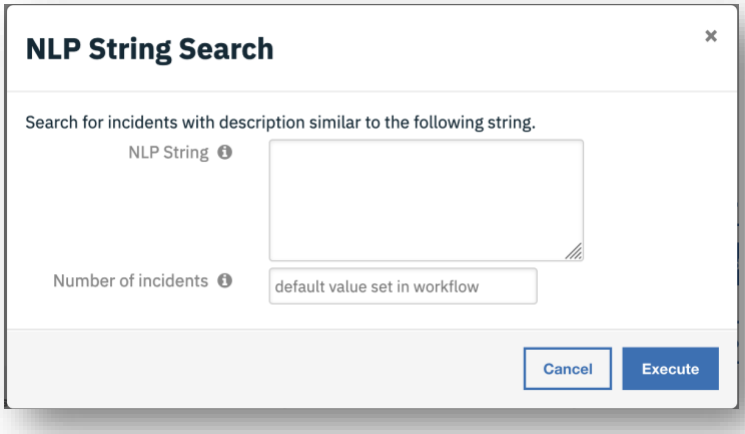
Assume that you already used the command line tool to build an NLP model. Now we are ready to use it to find similar incidents. Create a new incident and enter name and description. This information will be used to do NLP search.

From the incident Actions, select “NLP Search”.



Wait for the integration to finish its job. The results are shown. Note that you can click the “Similarity” column to force the table to sort according to similarity.

As you can see from the Actions menu, there is another menu item for “NLP String Search”. This is slightly different from the “NLP Search” above. For “NLP Search”, the function uses the name and description of the corresponding incident to do the NLP search. For “NLP String Search”, users can input what string to use to do NLP search. Thus, users can copy portion of the description into the following text field and do NLP search. This menu item can be useful if an incident contains some irrelevant information. Also shown in the image below, users can specify how many incidents to return.



Rebuild an NLP model

When more incidents are created after an NLP model is built, users shall rebuild the model making use of the newly available data. The process is the same as building a new model. In general, users can rebuild the NLP model once a week.

Resilient Platform Configuration

There are three sections in the app.config.

[resilient]

This section contains information regarding the Resilient platform. The command line component uses this to retrieve incidents from the Resilient platform, and the function component uses this to listen to command from the Resilient platform.

[fn_machine_learning_nlp]

This section is for the function component to locate the saved machine learning model used for prediction.

model_path	Absolute path to the folder that contains the saved NLP model
Num_features	Number of features for the word2vec model

Please refer to the Resilient Machine Learning Reference Guide for more information about Resilient Machine Learning.

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.

For errors related to building a machine learning model, please refer to <TBD> for more details.

Support

For support, visit https://ibm.com/mysupport.

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