

IBM Resilient



Incident Response Platform Integrations

Machine Learning Function V1.1.0

Release Date: March 2020

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 Machine Learning Function.

Overview

Resilient Machine Learning function provides a tool kit for users to train models using historical incident data. The models can then be used to make prediction for new incidents. This tool is integrated with Resilient platform and customized for Resilient users.

This integration provides the followings:

- A command line tool to build machine learning models
- A predict function that predicts a given field of an incident

Together with the above, this package also includes an example workflow that demonstrates how to call the function above, a rule that starts the workflow, and a custom field the function uses to write the prediction.

History

Resilient Incident Response Platform Machine Learning Function Reference Guide

Platform Version	Publication	Notes
1.1.0	March 2020	New ml.config file is used to build, train, and test machine learning models
1.0.0	December 2018	Initial publication.

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
```

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.

3. To install the package, you must first unzip it then install the package as follows:

```
sudo pip install --upgrade fn_machine_learning-<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
```

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, `app.config`, as follows:
 - a. In the `[resilient]` section, ensure that you provide all the information required to connect to the Resilient platform.
 - b. In the `[machine_learning_predict]` section, edit the settings as follows:

```
model_dir=path to the folder you are going to save your model files
```

4. Create the directory where the model files will be saved. The path to this directory will be used for `model_dir` in `app.config`.
5. From the directory specified by `model_dir`, use the `res-ml` command line tool to create two files: `ml.config` and `res-ml.log`

```
res-ml config
```

6. Edit the `ml.config` file, edit the `[machine_learning]` section:

```
#
#   Field   to predict
#
prediction=severity_code
#
#   Features to use
#       example:features=confirmed, incident_type_ids, negative_pr_likely
features=list_of_fields_for_features_separated_by_comma
#
#   Algorithms supported:
#       Logistic Regression, Decision Tree, Random Forest,
#       Dummy Classifier, SVM, SVM with Gaussian kernel, GaussianNB,
#       BernoulliNB, K-Nearest Neighbors
algorithm=Logistic Regression
#
#   Ensemble method is optional, it can be Bagging or
#   Adaptive Boosting (Optional)
#
method=None
#
#   Split samples for testing. 0.5 means 50% of the samples will be
#   used for testing
```

```

#
split=0.5

#
# Advanced options
#-----
#
# 1. Imbalance Class
#
# Predicted data could be imbalanced. Uncomment one of the following
# option to handle it
#
class_weight=balanced
#imbalance_upsampling=true
#
# 2. Data Preparation
#
# Some prediction values could be misleading for the machine
# learning model Put those values below. Samples with those
# values will be removed
#
unwanted_values=None
#
# 3. Filter (Optional)
#
# * Time filter: format YYYY-mm-dd
#
#time_start=2018-10-01
#time_end=2018-10-08
#
# * Count: Maximum number of samples to process
#
#max_count = 10000

```

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]
```

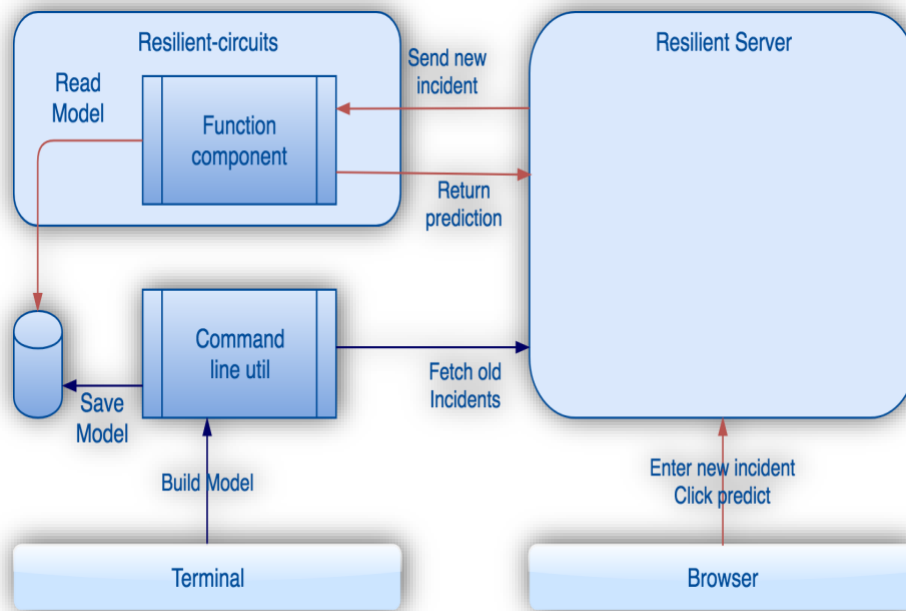
You can view log files for systemd and the resilient-circuits service using the journalctl command, as follows:

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

Function Descriptions

This integration package contains two components.

- A command line component to build machine models.
- A function component to predict using a saved machine model.



As shown in the above graph, users use the command line tool from a terminal to build a machine learning model. The command line tool reads configuration settings from the app.config file, and connects to the specified Resilient server to fetch incidents. These incidents are the samples used to train a machine model. The app.config file also contains other settings about the model to build, such as the features, the field to predict and algorithm. The command line tool can then build the model and saves it into a file.

The function component takes an incident ID and a file name. It retrieves the incident using the ID, and loads the saved model using the file name. It can then use the model to make predictions for this incident. This is triggered by a menu item in the incident page.

In summary, there are two processes, one to build a machine learning model, the other to make predictions using a model.

Build a Machine Learning Model

The command line tool is used to build machine models. It contains 5 subcommands.

Download

This subcommand is used to download incidents from a Resilient platform, and then save them into a CSV file. The Resilient platform is specified in the ml.config file. The subcommand has the following parameter:

-o Required. File path to the file to save the CSV file

Example:

```
res-ml download -o resilient_incidents.csv
```

Example output:

```
Saved 5 samples into resilient_incidents.csv
```

It shows in this example that 5 incidents have been downloaded and saved into resilient_incidents.csv.

Build

This subcommand is used to build a new model. The command line tool reads settings from ml.config regarding the features, predict, and algorithm. The default settings in ml.config shall be valid for a basic model. Users shall adjust them to get better performance.

This subcommand takes two flags:

- o Required. File path to the file to save the built model
- c Optional. File path to a CSV file that contains the samples.
If this flag is absent, the tool downloads incidents from the Resilient Server (specified in app.config) as samples. If this flag is given, the CSV file is used for samples.

Example:

```
res-ml build -c resilient_incidents.csv -o first_model.ml
```

If the model can be built successfully, you see a summary; for example:

```
-----
Summary:
-----
File:           /home/yj/git/resilient-community-apps/fn_machine_learning/fn_machine_learning/bin/first_model.ml
Build time:     2018-11-13 08:38:12
Num_samples:    15698
Algorithm:      Decision Tree
Method:         None
Prediction:     false_positive
Features:       incident_type_ids, category, confirmed, alert_source, involved_party
Class weight:   balanced
Upsampling:     False
Unwanted Values: None
Accuracy:       0.884571282966
F1:            0.880141116459
  Accuracy for false_positive value:
    False:      0.867940028753
    True:       0.911744966443
```

It shows the overall accuracy, and the accuracy for each value of the predicted field. Here the predicted field (false_positive) has two values: False and True. The accuracy for True is 91.17% and the accuracy for False is 86.79%. The overall accuracy is 88.45%. Please refer to Resilient Machine Learning Reference Guide for more details regarding the other fields.

In this example, the model is built and saved into a file called first_model.ml.

View

This subcommand is used to view a saved model. When more than one models are built for comparison, this subcommand is useful to figure out which model to use.

It takes one flag:

- i Required. File path to the saved model to be rebuilt

Example:

```
res-ml view -i first_model.ml
```

It shows the summary of the saved model. The output is the same as the build subcommand.

Rebuild

This subcommand is used to rebuild a saved model. After a model is built and used for a while, new incidents are accumulated. A model needs to be updated and this subcommand is for rebuilding the model. It takes two flags:

- i Required. File path to the saved model to be rebuilt
- c Optional. File path to a CSV file that contains the samples.
If this flag is absent, the tool downloads incidents from the Resilient Server (specified in ml.config) as samples. If this flag is given, the CSV file is used for samples.

Please note that when a model is rebuilt, the predict/features/algorithm/method information are taken from the saved file instead of from ml.config. This subcommand is intended for rebuilding/updating a successful model after new samples are available.

Example:

```
res-ml build -i lg_adaboost.ml
```

If the model can be rebuilt successfully, a summary is shown.

Count-value

This subcommand is used to detect imbalanced dataset. Please refer to Resilient Machine Learning Reference Guide for details about imbalanced dataset and the recommendations to optimize the performance of a model for handling it.

- i Required. File path to the CSV file that contains the models
- f Required. The field you want to check value count.

Example:

```
res-ml count_value -i resilient_incidents.csv -f false_positive
```

The output shows the count of each value of this field “false_positive”.

```
-----  
Value Counts  
-----  
Value counts for false_positive in resilient_incidents.csv:  
False    9738  
True     5960  
None     1443  
Name: false_positive, dtype: int64
```

In this example, there are three values. “None” means some incidents have this “false_positive” empty.

Use a model to predict

Once a machine model is built and saved, it can be used to make prediction by the function component. The function component locates the saved model to use by looking at the following:

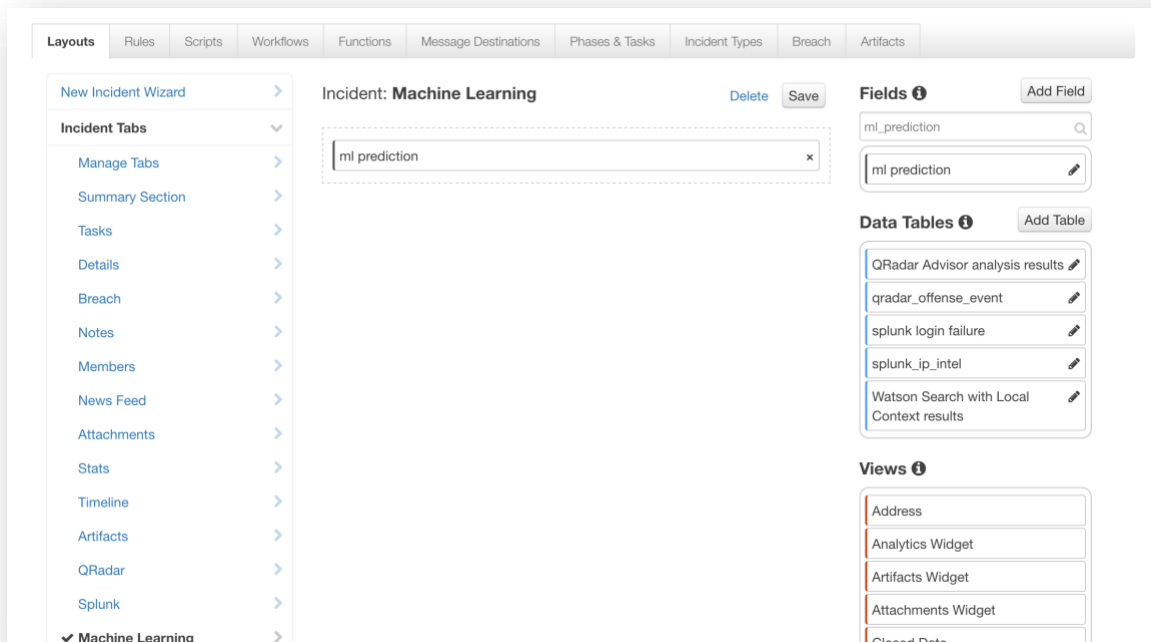
- Model folder: “model_folder” is set in app.config. A saved model file needs to be in this folder.
- Model name: The name of the model to be used is given as an input to the function. It can be set in the input tab of the workflow that calls the function. For the sample workflow included in this package, the default name is “incident_prediction.ml”. If you are using the sample workflow to make prediction, you need to copy the model file you want to use into the model folder as incident_prediction.ml.



The screenshot shows a configuration window with tabs: Input, Pre-Process Script, Output, and Post-Process Script. The 'Input' tab is active, displaying a table with two input parameters:

Input Parameter	Value
ml_incident_id ⓘ	<input type="text"/>
ml_model_file ⓘ	incident_prediction.ml

The sample workflow writes the prediction into a custom field call ml_prediction. To show this in the incident page, add this customer field to it. You can go to Customization Settings page and create a new tab call “Machine Learning”. Then add this custom field into the newly created tab.



The screenshot shows the 'Customization Settings' page with tabs: Layouts, Rules, Scripts, Workflows, Functions, Message Destinations, Phases & Tasks, Incident Types, Breach, and Artifacts. The 'Layouts' tab is active, showing the configuration for an incident named 'Machine Learning'.

Incident: Machine Learning [Delete] [Save]

Incident Tabs: A list of tabs including Manage Tabs, Summary Section, Tasks, Details, Breach, Notes, Members, News Feed, Attachments, Stats, Timeline, Artifacts, QRadar, Splunk, and Machine Learning (checked).

Fields: A search bar and a list of fields. The field 'ml_prediction' is added to the incident page layout.

Data Tables: A search bar and a list of data tables including QRadar Advisor analysis results, qradar_offense_event, splunk login failure, splunk_ip_intel, and Watson Search with Local Context results.

Views: A search bar and a list of views including Address, Analytics Widget, Artifacts Widget, Attachments Widget, and Closed Date.

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

New Test

Actions

Summary

ID 2126

Phase Engage

Severity Low

Date Created 08/10/2018

Date Occurr... —

Date Discov... 08/22/2018

Data Compr... Unknown

Incident Type Denial of Service

Description

No description.

Tasks

Details

Breach

Notes

Members

News Feed

Attachments

Stats

Timeline

Artifacts

QRadar

Splunk

Machine Learning

Machine Learning

ml prediction ⓘ —

Edit

People

Created By Yongjian Feng

Owner Yongjian Feng

Members There are no members.

Related Incidents

No related incidents.

<https://192.168.56.201/incidents/2126?tab=020aa43a-a2f5-c385-ede4-80d2c26b80bc>

Now we are ready to make a prediction. Please note that you need to ensure that all the features are non-empty. Go to the “Details” tab to check. For the default settings, features are “NIST Attack Vectors”, “Incident Type”, “Incident Disposition”, and “Negative PR”. As shown in the following screenshot, none of these fields are blank. This an incident that the function can predict.

Details

Basic Details

Name ⓘ	New Test
Description ⓘ	—
Incident Type ⓘ	Denial of Service
NIST Attack Vectors ⓘ	E-mail Web
Incident Disposition ⓘ	Unconfirmed
Phase ⓘ	Engage
Resolution ⓘ	—
Resolution Summary ⓘ	—
Owner	Yongjian Feng
Created By	Yongjian Feng

Date and Location

Date Created ⓘ	08/10/2018 08:28
Date Occurred ⓘ	—
Date Discovered ⓘ	08/22/2018 11:32:47

Address ⓘ	—
City	—
Country	United States
State	—
Zip Code	—

Implications

Criminal or nefarious activity? ⓘ	Unknown
Exposure Type ⓘ	Unknown
Department	—
Negative PR ⓘ	No
Reporting Individual ⓘ	—
Severity ⓘ	Low

From the incident Actions, select “ML Predict”.

The screenshot shows the 'New Test' page in the Splunk Security Console. The page has a header with 'New Test' and an 'Actions' dropdown menu. The 'Actions' menu is open, showing a list of actions including 'ML Predict', 'ML Predict Fishing', 'ML Predict Ransomware', 'new message', 'QRadar Advisor Offense Analysis', 'Search QRadar for offense id', 'TestCircuits', 'TestSplunkSearch', 'Update Splunk ES notable event', 'Action Status', 'Workflow Status', 'Close Incident', and 'Delete Incident'. The 'ML Predict' option is highlighted. The main content area is divided into two columns: 'Summary' and 'Description'. The 'Summary' column contains fields for ID (2126), Phase (Engage), Severity (Low), Date Created (08/10/2018), Date Occurred (—), Date Discovered (08/22/2018), Data Compromised (Unknown), and Incident Type (Denial of Service). The 'Description' column contains the text 'No description.' and a tabbed interface with tabs for Tasks, Details, Breach, Notes, Members, News Feed, Attachments, and Stats. The 'Machine Learning' tab is selected, showing a 'Machine Learning' section with a 'ml prediction' field and a minus sign. The 'People' section shows 'Created By' and 'Owner' as 'Yongjian Feng' and 'Members' as 'There are no members.' The 'Related Incidents' section shows 'No related incidents.'

Wait for the integration to finish its job. The prediction is shown.

The screenshot shows the 'New Test' page in the Splunk Security Console. The page has a header with 'New Test' and an 'Actions' dropdown menu. The main content area is divided into two columns: 'Summary' and 'Description'. The 'Summary' column contains fields for ID (2126), Phase (Engage), Severity (Low), Date Created (08/10/2018), Date Occurred (—), Date Discovered (08/22/2018), Data Compromised (Unknown), and Incident Type (Denial of Service). The 'Description' column contains the text 'No description.' and a tabbed interface with tabs for Tasks, Details, Breach, Notes, Members, News Feed, Attachments, Stats, Timeline, Artifacts, and QRadar. The 'Machine Learning' tab is selected, showing a 'Machine Learning' section with a 'ml prediction' field and a 'Medium' prediction result. The 'People' section shows 'Created By' and 'Owner' as 'Yongjian Feng' and 'Members' as 'There are no members.' The 'Related Incidents' section shows 'No related incidents.'

Build multiple models

When res-ml is executed, it looks for ml.config using this order:

1. ml.config file in the same folder where res-ml runs.
2. app.config in the .resilient folder of the home directory.

If a user wants to maintain multiple models, one way to do it is to keep each model in its own folder with its own ml.config.

Use multiple models to predict

If a user wants to predict more than one field, then multiple models are needed. For example, a user can use one model to predict “severity_code”, and another model to predict “owner”. The multiple models need to be built first. Also, multiple workflows are needed, one for each model. All workflows call the same function but provide different input regarding the model file to use.

Most likely, a user also needs to create multiple rules, one for each workflow.

Resilient Platform Configuration

There are two 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.

[machine_learning_predict]

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

There are two sections in ml.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.

[machine_learning]

This section contains settings for building a machine learning model. The default values for important fields given in the ml.config are listed below. It uses four built-in fields as features to predict severity.

Predict	severity_code
features	incident_type_ids, confirmed, negative_pr_likely, nist_attack_vectors
Algorithm	Logistic Regression
split	0.5
Class_weight	balanced

unwanted_features	None
-------------------	------

Please refer to the Resilient Machine Learning Reference Guide for guidelines to build a useful machine model and improve its performance.

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 additional support, contact support@resilientsystems.com.

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