Resilient Machine Learning User Guide

Installation

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

1. Run the following command to install pandas:

sudo pip install pandas-0.23.4-cp27-cp27m-linux\_86\_64.whl

This step is necessary because of the following. 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 won’t be able to built 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 “pip install pandas” downloads.

1. Run the following command to install the package:

sudo pip install --upgrade fn\_machine\_learning-<*version*>.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

1. 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

1. Edit the resilient-circuits configuration file, as follows:
   1. In the [resilient] section, ensure that you provide all the information required to connect to the Resilient platform.
   2. In the [machine\_learning\_predict] section, edit the settings as follows:

active\_model=path\_to\_a\_saved\_model\_file

* 1. In the [machine\_learning] section, edit the settings as follows:

prediction=field\_to\_predict

features=fields\_to\_be\_used\_as\_features\_separated\_by\_comma

algorithm=algorithm\_for\_model

method=optional\_ensemble\_method

split=split\_train\_test\_default\_0.5

Deploy customizations to the Resilient platform

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

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

resilient-circuits customize

1. Respond to the prompts to deploy function, message destination, workflow, script, and rule.

Run the integration

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.

﻿Build a machine learning model

This package includes a command line tool to build machine model. It reads the setting from the [machine\_learning] section of the app.config to build a model. This command line tool has three subcommands.

## Build

This subcommand is used to build a new model. It 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, this tool will download incidents from the Resilient Server (specified in app.config) as samples.  If this flag is given, the CSV file is used for samples instead. |

Example:

res-ml build -o lg\_adaboost.ml

If the model can be built successfully, an output like this will be shown



## Rebuild

This subcommand is used to rebuild a saved 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, this tool will download incidents from the Resilient Server (specified in app.config) as samples.  If this flag is given, the CSV file is used for samples instead. |

Please note that when a model is rebuilt, the predict/features/algorithm/method information will be taken from the saved file, instead of from app.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, an output like this will be shown



## View

This subcommand is used to view a saved model. It takes one flag:

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

Example:

res-ml view -i lg\_adaboost.ml

It will show the summary of the saved model.



Use a ML model to predict

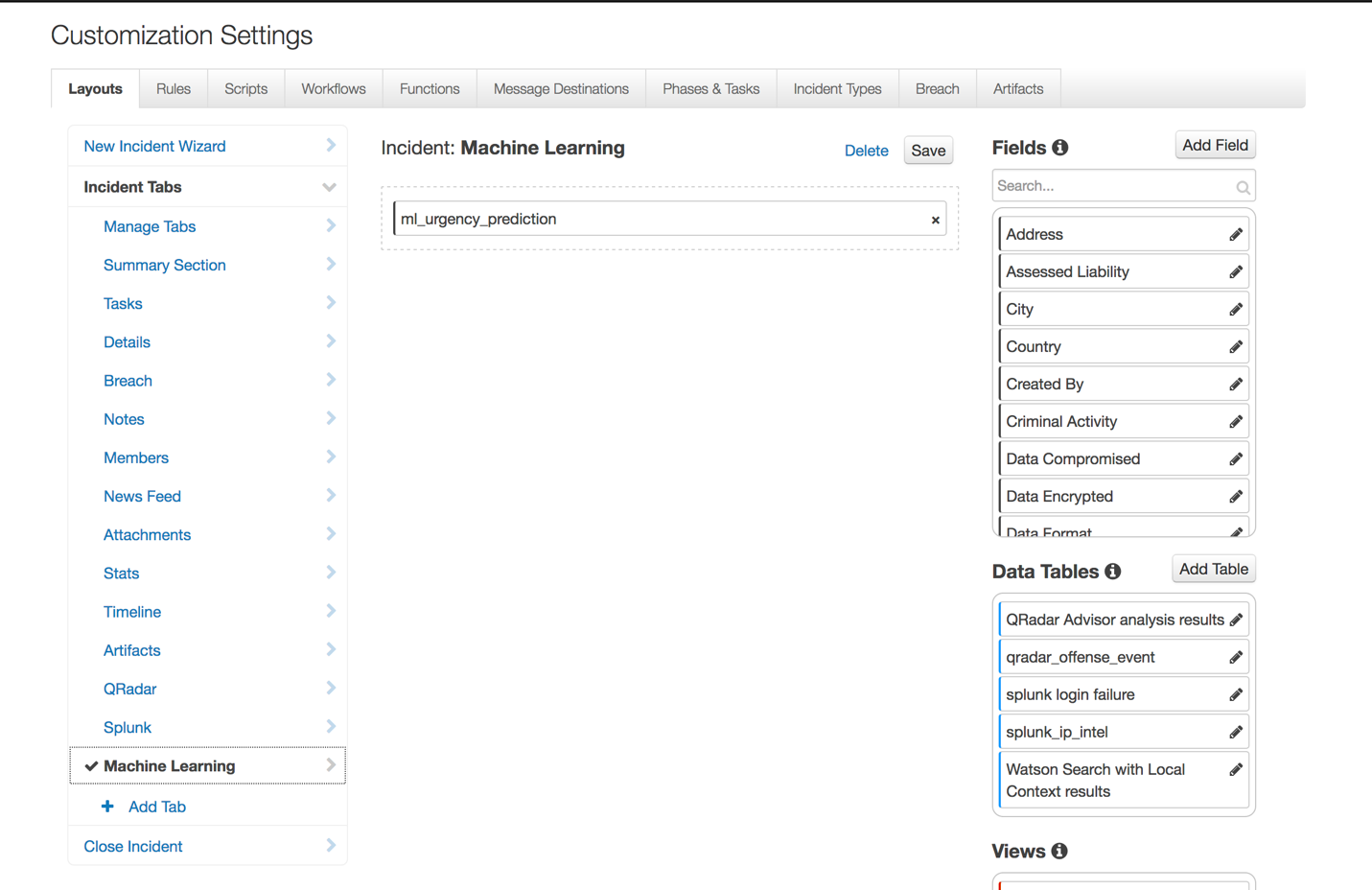
First of all, copy the model you want to use for prediction to the active\_model specified in the app.config. (This is easier than modifying the app.config to point to the model file you want to use, because you don’t need to restart resilient-circuits).

Back to Resilient Server, the sample workflow included in this integration is called “ML predict urgency”. A custom field called “ml\_urgency\_prediction” has been added to your Resilient Server when you ran

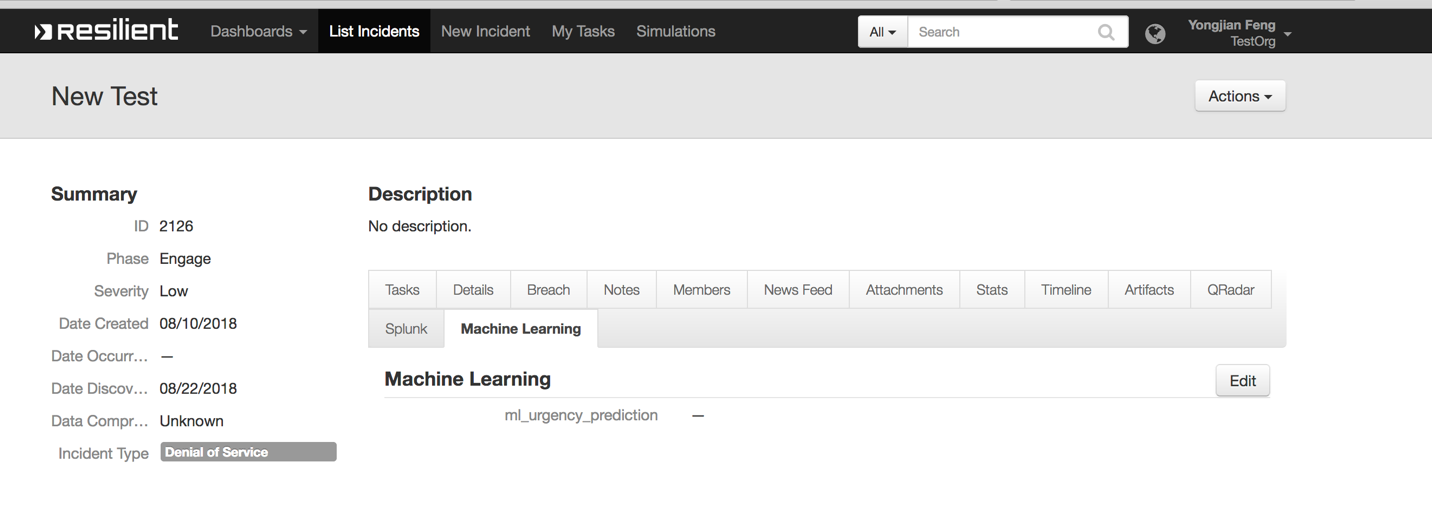
resilient-circuits customize

above.

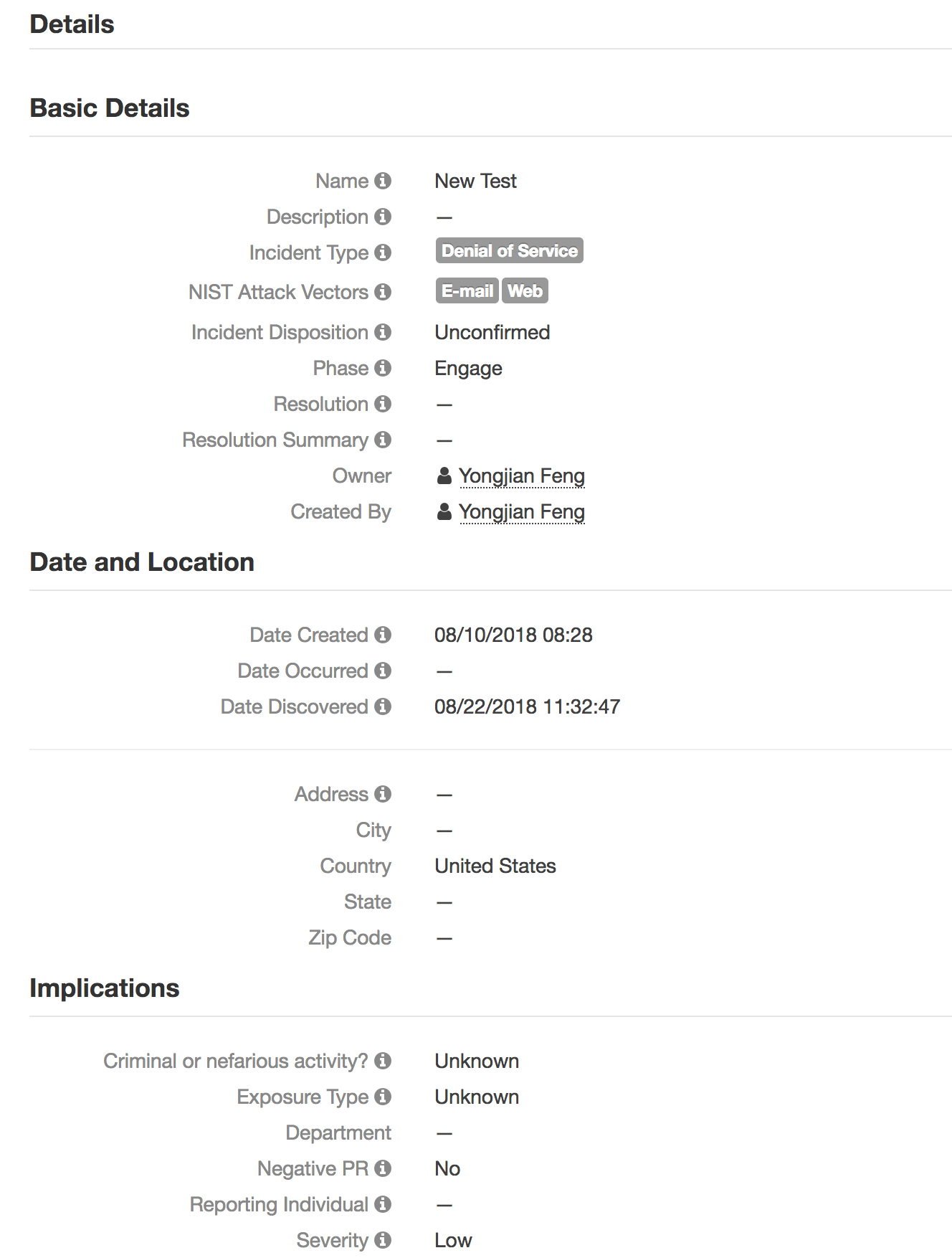
You can go to Customization Settings page, and create a new tab call “Machine Learning”. Then add this custom field into the newly created tab.

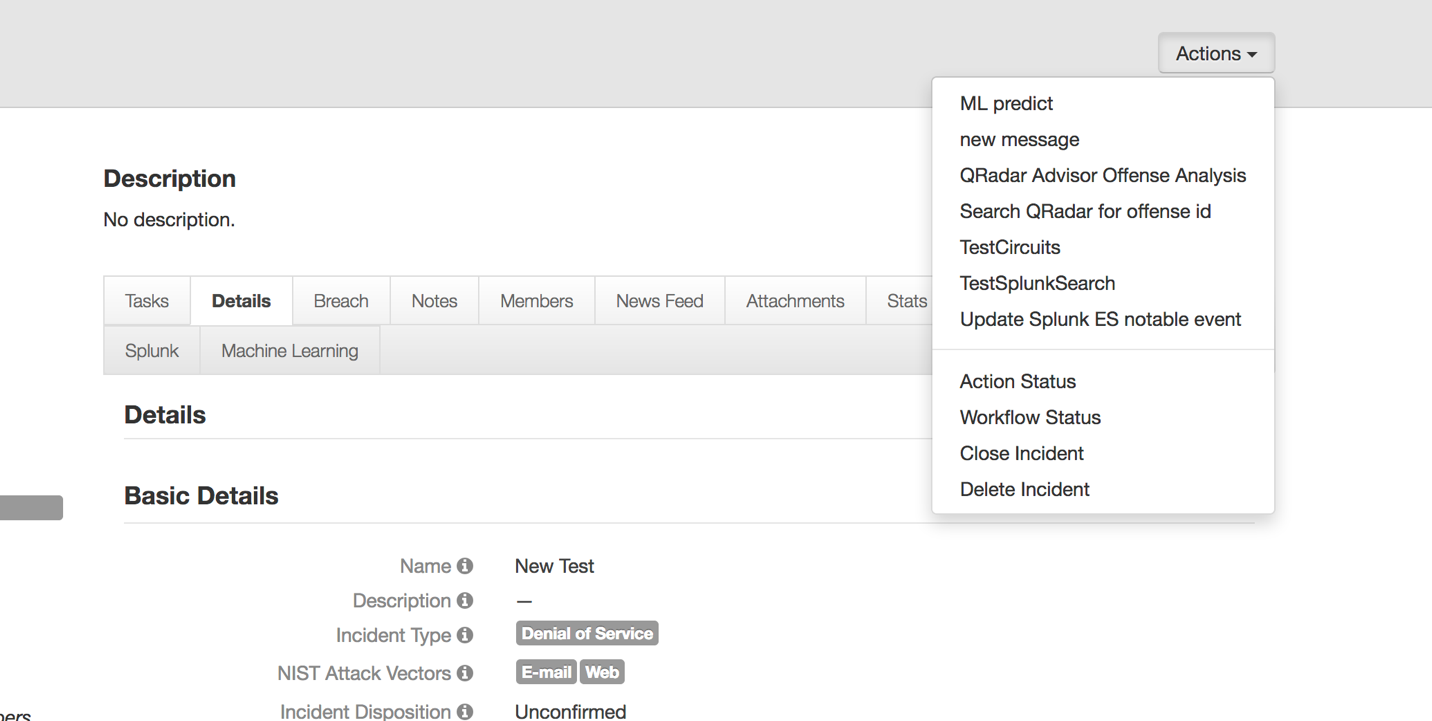


Then this tab and custom field will be shown in the incident page

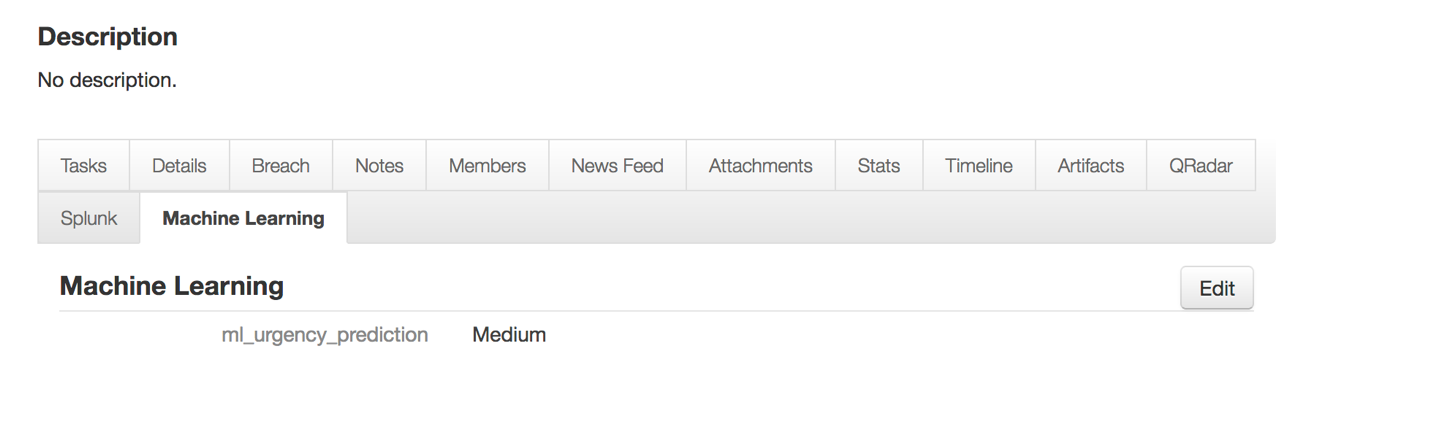


Now we are ready to predict. Please note that you need to ensure that all the features are non-empty. Go to the “Details” tab to check. In this particular example, features are “NIST Attack Vectors”, “Incident Type”, “Incident Disposition”, and “Negative PR”. From the following screenshot, we can see none of these fields is blank here. So this an incident we can predict.



Now from the incident Actions, click “ML predict”.

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



Note

Note in the sample workflow, we chose to store the prediction in a custom field ml\_urgency\_prediction. So user needs to manually copy it over to severity\_id (which is the predict field). So in some sense, the prediction is a suggestion, and the user still has to make the final decision in our sample.

If customer wants to write the prediction value into severity\_id directly, he just need to do this in the post-process script of the sample workflow.