Application Examples of our Modules

We will consider here the following samples:

- the benign files `benign1`, `benign2` and `benign3`;
- the following malicious files `malicious1`, `malicious2` and `malicious3`;
- the benign folders `Dir-Benign1`, `Dir-Benign2`, `Dir-Benign3`;
- the malicious folders `Dir-Malicious1`, `Dir-Malicious2` and `Dir-Malicious3`;
- the unknown files `unknown1`, `unknown2`;
- and the unknown folders `Dir-Unknown1` and `Dir-Unknown2`.

Each name representing an hypothetical path to the files/folders you wish to analyse.

JavaScript Detection Tool

Detection of JavaScript samples respecting the grammar defined by `ECMA-International`, detection of broken JavaScript, and files not written in JavaScript.

Help
> python3 <path-of-JsDetection/JsDetection.py> --help
Mandatory attributes are --f <list-of-files> or --d <list-of-repositories>. The two options can be combined.

To analyse files
> python3 <path-of-JsDetection/JsDetection.py> --f benign1 malicious1

To analyse directories
> python3 <path-of-JsDetection/JsDetection.py> --d Dir-Malicious1 Dir-Unknown1 Dir-Unknown2

To analyse files and directories
> python3 <path-of-JsDetection/JsDetection.py> --f benign1 malicious1 --d Dir-Malicious1 Dir-Unknown1 Dir-Unknown2

Static Analysis of JavaScript Executables

Both lexical and syntactical analysis of JavaScript samples can be performed. This study is based on a frequency analysis of the 4-grams present in the considered files.

Help
> python3 <path-of-src/MainStaticAnalysisJs.py> --help
Mandatory attributes are --f <list-of-files> or --d <list-of-repositories>. The two options can be combined.

To create a CSV file containing the probability of the 4-grams encountered in the considered files
> python3 <path-of-src/MainStaticAnalysisJs.py> --f benign1 malicious1 --d Dir-Malicious1 Dir-Unknown1 Dir-Unknown2

To add a label in the CSV file (default value being '?' for unknown)
The first label stands for the first file and the first directory (if any), the second for the second ones etc.
> python3 <path-of-src/MainStaticAnalysisJs.py> --f benign1 malicious1 --l 'benign' 'malicious'
Here benign1 is labeled as 'benign' and malicious1 as 'malicious'.

> python3 <path-of-src/MainStaticAnalysisJs.py> --d Dir-Malicious1 Dir-Unknown1 Dir-Unknown2 --l 'malicious'
Here Dir-Malicious1 is labeled as 'malicious'. As for Dir-Unknown1 and Dir-Unknown2, they are labeled as '?' because no label has been given for them.

To store the CSV file in the directory Test/ (--ep option)
> python3 <path-of-src/MainStaticAnalysisJs.py> --f benign1 malicious1 --l 'benign' 'malicious' --ep 'Test/'

To produce histograms (--h option, the path can be chosen with the --hp option) **and/or PCA** (--g option , the path can be chosen with the --gp option)
> python3 <path-of-src/MainStaticAnalysisJs.py> --f benign1 malicious1 --l 'benign' 'malicious' --h True --g True

Clustering of JavaScript Executables

Clustering of JavaScript samples into k (configurable) families.

Help
> python3 <path-of-MachineLearning/Clustering.py> --help
Mandatory attributes are: --f <list-of-files> (or --d <list-of-repositories>, the two options can be combined);

and --c <number-of-clusters>.

Clustering of the given files in 5 clusters with k-means++ algorithm
> python3 <path-of-MachineLearning/Clustering.py> --f benign1 malicious1 --d Dir-Malicious1 Dir-Unknown1 Dir-Unknown2 --c 5

Clustering of the given files in 5 clusters with k-means++ algorithm **and graphical representation of the clusters** (--g option)
> python3 <path-of-MachineLearning/Clustering.py> --f benign1 malicious1 --d Dir-Malicious1 Dir-Unknown1 Dir-Unknown2 --c 5 --g True

Classification of JavaScript Executables

Detection of malicious JavaScript documents.

- **Creating a Model**

Help
> python3 <path-of-MachineLearning/LearnModel.py> --help
Mandatory attributes are: --f <list-of-files> (or --d <list-of-repositories>, the two options can be combined);

and --l <list-of-labels>.

As previously, the first label stands for the first file and directory (if any), the second for the second ones etc.

To create a model using a directory containing benign files, and another one with malicious files
> python3 <path-of-MachineLearning/LearnModel.py> --d Dir-Malicious1 Dir-Benign1 --l 'malicious' 'benign'

To specify a model path (--md option) **and/or a model name** (--mn option)
> python3 <path-of-MachineLearning/LearnModel.py> --d Dir-Malicious1 Dir-Benign1 --l 'malicious' 'benign' --md 'Test/' --mn 'model1'

To test the files used to build the model on the model and print the score (--ps option)
> python3 <path-of-MachineLearning/LearnModel.py> --d Dir-Malicious1 Dir-Benign1 --l 'malicious' 'benign' --ps True

To test the files used to build the model on the model and print the predictions (--pr option)
> python3 <path-of-MachineLearning/LearnModel.py> --d Dir-Malicious1 Dir-Benign1 --l 'malicious' 'benign' --pr True

- **Updating a Model**

Help
> python3 <path-of-MachineLearning/UpdateModel.py> --help
Mandatory attributes are: --f <list-of-files> (or --d <list-of-repositories>, the two options can be combined);

--l <list-of-labels>;

and --m <old-model-path> (see the previous point to create a model).

To update a model with benign and malicious files
> python3 <path-of-MachineLearning/UpdateModel.py> --f malicious1 malicious2 benign1 --l 'malicious' 'malicious' 'benign' --m <path-of-the-model-to-be-updated>

To specify a path for the new model (--md option) **and/or a model name** (--mn option)
> python3 <path-of-MachineLearning/UpdateModel.py> --f malicious1 malicious2 benign1 --l 'malicious' 'malicious' 'benign' --m <path-of-the-model-to-be-updated> --md 'Test/' --mn 'model2'

- **Testing a Model / Classifying new Files**

Help
> python3 <path-of-MachineLearning/ClassifyWithModel.py> --help
Mandatory attributes are: --f <list-of-files> (or --d <list-of-repositories>, the two options can be combined);

and --m <path-of-the-model-to-be-used> (see how to create a model, previously).

To detect malicious JS executables
> python3 <path-of-MachineLearning/ClassifyWithModel.py> --d Dir-Unknown1 Dir-Unknown2 --m <path-of-the-model-to-be-used>
A list of the files tested (only those respecting the grammar defined by `ECMA-International`) will be returned. For each file, you will see whether it was classified as 'benign' or as 'malicious'.