Tool and Artifacts (TAR) Paper

Evaluation of Anomaly Detection algorithms made easy with RELOAD

Tommaso Zoppi, Andrea Ceccarelli, Andrea Bondavalli
University of Florence, Department of Mathematics and Informatics, Viale Morgagni 65 – Florence (IT)
{tommaso.zoppi, andrea.ceccarelli, andrea.bondavalli}@unifi.it

Abstract — **Anomaly detection aims at identifying patterns in data that do not conform to the expected behavior. Despite anomaly detection has been arising as one of the most powerful techniques to suspect attacks or failures, dedicated support for the experimental evaluation is actually scarce. In fact, existing frameworks are mostly intended for the broad purposes of data mining and machine learning. Intuitive tools tailored for evaluating anomaly detection algorithms for failure and attack detection with an intuitive support to sliding windows are currently missing. This paper presents RELOAD, a flexible and intuitive tool for the Rapid EvaLuation Of Anomaly Detection algorithms. RELOAD is able to automatically i) fetch data from an existing data set, ii) identify the most informative features of the data set, iii) run anomaly detection algorithms, including those based on sliding windows, iv) apply multiple strategies to features and decide on anomalies, and v) provide conclusive results following an extensive set of metrics, along with plots of algorithms scores. Finally, RELOAD includes a simple GUI to set up the experiments and examine results. After describing the structure of the tool and detailing inputs and outputs of RELOAD, we exercise RELOAD to analyze an intrusion detection dataset available on a public platform, showing its setup, metric scores and plots.**

Keywords — anomaly detection, intrusion detection, tool, RELOAD, algorithm, sliding windows, machine learning.

I. Anomaly Detectors

Cyber-physical infrastructures or Systems of Systems are composed of many different software layers and a multitude of services. Due to the complexity, dynamicity and governance of these systems, instrumenting each individual service for monitoring purposes and characterizing all the possible errors or attack that may manifest is often not feasible [23], despite being widely acknowledged as crucial. To such extent, *anomaly detection* [1] was proposed; it deals with the problem of finding patterns in data that do not conform to the expected behavior.

Patterns refer to alterations of the behavior of services or systems that are caused by specific and non-random factors. Pattern changes may be caused by ongoing attacks [2], services misbehavior [23], or failures [3]. Anomaly detectors characterize an expected behavior of a service or a system and compare it with observed data to infer the health of such system. Anomaly detection has been proven useful to timely detect attacks and failures in a multitude of works; for example, to support intrusion detection systems [2], [36], detect side-channel attacks [34], identify occurring failures in a critical application [11], detect faults in high-dimensional data streams [21], achieve dependability assurance in utility clouds [22] or Systems-of-Systems [49], analyze production logs [47], or identify suspicious behaviors of applications [23].

Four actions are crucial for the proper application of anomaly detection solutions: *i*) selection of the relevant features of the target system that should be monitored i.e., the features that are most appropriate to identify occurring anomalies; *ii*) identification of the most suitable anomaly detection algorithm, for a given system; *iii*) appropriate tuning of algorithm parameters; and *iv*) proper identification of voting strategies, to decide on anomalies based on the information offered by the above-mentioned features.

To perform these activities, this paper presents the RELOAD tool, a software solution that is specifically crafted to support and automate the evaluation of unsupervised anomaly detection algorithms intended for the purpose of attack and failure detection. The tool offers an automated methodology to: *i*) import data from data sets; *ii*) select the most relevant features of the data set; *iii*) import anomaly detection algorithms; *iv*) run algorithms on data sets; *v*) evaluate multiple strategies to decide on anomalies based on results from single features; *vi*) evaluate the algorithms for different configurations i.e., suggest the most suitable configuration of algorithms parameters; *vii*) present results of the evaluation activity. RELOAD can also operate with algorithms that are based on sliding windows e.g., [21], [11]. As conclusive remark, RELOAD is easy to use: this has been also confirmed by non-experts, which used the tool for educational purposes.

This paper is structured as follows: Section II reports on the available solutions for data mining and data analysis, illustrating the contribution of the tool. Section III describes the structure and methodology, while Section IV details the GUI of the tool. Section V presents a case study in which RELOAD is successfully applied. Section VI describes the usability assessment that we performed with the aid of MSc students. Finally, Section VII concludes the paper.

II. Related Works and Contribution

A. Existing Solutions and Limitations

Understanding the role of tools to evaluate and compare anomaly detection algorithms is intuitive. In fact, it is generally difficult to perform an extensive experimental campaign without supporting frameworks that automate execution of experiments and data analysis.

For data mining purposes, frameworks such as *ELKI* [4], *WEKA* [7], *RapidMiner* [5], or libraries such as *Scikit* [6] and *Pandas* were created that allow comparing the performance of essentially *any* data mining and machine learning algorithm following specific methodologies. ELKI and WEKA provide Java-based executables with built-in algorithms including anomaly detection ones, while Scikit and Pandas are a *Phyton* libraries for data mining that include anomaly-based techniques such as *Random Forests* [35] or *Gradient Boosting* [10]. Despite these solutions allow extending their functionalities, they require extensive customization to execute algorithms for determined purposes and settings, and in general their full set of functionalities is not intuitive to grasp. Instead, *RapidMiner* [5] is an enterprise suite that offers graphical support with a complete user interface, but adding algorithms is rather complicated and the tool is not free to use.

Summarizing, these powerful solutions already include some anomaly detection algorithms, and new ones can be added. As a drawback, we observe that features, settings and usage are not always easy to understand and do not offer a complete support to the *experimental evaluation of* unsupervised *anomaly detection algorithms* in the domain of dependable and secure systems. In particular, the following requirements can be achieved – when possible – with the above solutions only through extensive customization and experience:

- Focus on unsupervised algorithms, which are often acknowledged as the most suitable way to identify unknown vulnerabilities as zero-day attacks [1], [36], [49]: implementations of unsupervised algorithms are scattered in the above-mentioned frameworks (e.g., Isolation Forests [30] can be found only in [7]).
- Provide support for sliding windows algorithms, that are often relevant in this domain [21], [11].
- Integrate different feature selection techniques, allowing also to sequentially executing a pool of the above techniques. This way, it is possible to define a subset of relevant features to be monitored, and evaluate voting strategies to decide on anomalies on the basis of the scores from such features. This is important to maximize detection efficacy and minimize resource consumption.

B. RELOAD Characteristics

The peculiarities and characteristics of RELOAD can be described along the following lines.

Domain. RELOAD provides functionalities specific for the evaluation of anomaly detection algorithms intended for detecting unknown attacks or errors. In fact, extensive support is offered for unsupervised and sliding window algorithms, automated tailoring of features, profiling of algorithms parameters, and voting strategies.

Input/output data. Our tool can load data from different kinds of data sources, that may either be i) text files, or ii) MySQL databases, while a support for data streams is currently under development. In addition, it provides outputs results in CSV files that can be easily manipulated. Currently, it contains loaders to the attack data sets KDDCup99 [15], NSL-KDD [12], ADFA-LD [16], ISCX2012 [13], and UNSW-NB15 [14], and to the failure data set available in [37].

Feature Selection. Once datasets are loaded, the user may and should apply feature selection techniques [19], to filter out features that will not provide actionable data for the purpose of identifying errors or attacks. While some built-in strategies were implemented, e.g., *Variance*, *Pearson correlation* [19], a big pool of feature selection strategies as the one based on *Information Gain* are derived from the implementations in the WEKA [7] framework.

Algorithms. RELOAD is able to integrate existing algorithms as well as novel algorithms written by RELOAD users. Further, it offers the opportunity to run different algorithms on data sets and collect metric scores. To facilitate comparison, RELOAD includes 10 algorithms, selected among those already used in research works for failure and intrusion detection. The algorithms are selected from the six main families [1], [18]: *clustering* (K-Means [4]), *statistical* (HBOS [25]), *classification* (SVM [28], Isolation Forest [30]), *neighbour-based* (kNN [27], ODIN [26]), *density-based* (LOF [24], COF [31]) and *angle-based* (ABOD [29], FastABOD [29]). Some of the algorithms were implemented by RELOAD users, while most of them were imported from ELKI and WEKA. In addition, the tool embeds 6 sliding window algorithms such as SPS [11].

Metrics. The tool computes the most relevant metrics for evaluating anomaly detectors [9], selected by surveying research papers. Built-in metrics are reported and discussed in Section III. This offers a comprehensive evaluation with the default configuration of RELOAD. New metrics can be defined if needed.

Usability. The tool has an intuitive GUI to select the target algorithms, define configuration parameters, and choose data sets. It does not need relevant expertise; for example, it was successfully used by MSc students with limited experience on experimental evaluation (see Section VI).

Extensibility. RELOAD can be modified to add new algorithms, metrics or voting strategies. It is open source, developed in *Java* to increase portability, available at [20].

C. Contribution and Relevance of RELOAD

The tool itself does not provide novel technologies, algorithms or techniques, but instead adopts state-of-the art findings in an orchestrated and intuitive fashion. As opposed to existing tools and web portals as *ELKI*, *WEKA*, *Scikit (*and *Pandas)*, *RapidMiner* and others, RELOAD embeds the following characteristics (see also Table I).

- *Easy to use*, since it embeds known techniques by hiding many implementation details and variants to the final user, which is requested to select just a few inputs. Indeed, it is a powerful artifact to be used as a teaching support for bachelor or masters' degrees.
- *Open source*, since code is available on online public repositories and free to use.
- *Lightweight and portable*, given that Java 8+ is installed in the target machine.

TABLE I. COMPARING RELOAD WITH EXISTING FRAMEWORKS. ✔INDICATES OPTIMAL MATCHING, * IDENTIFIES A PARTIAL OR SUB-OPTIMAL MATCHING.

Framework	GUI	Open Source	Extensible	Unsupervised Algorithms	Sliding Windows
WEKA		✔	✔	*	✔
ELKI	✔	✔	✔	*	*
RapidMiner	✔		*	✔	✔
Scikit-Pandas		✔	✔	✔	*
RELOAD	✔	✔	✔	✔	✔

- *Easily extensible* by adding new algorithms, with built-in interfaces to embed algorithms and other techniques e.g., feature selection strategies, depending on the needs of the user.
- *Shaped for anomaly detection,* selecting relevant unsupervised algorithms from different frameworks and including a full – but almost transparent to the user – support to sliding windows (see "Unsupervised Algorithms" and "Sliding Windows" in Table I).

III. The RELOAD Tool

We describe the inputs, the most relevant components of the tool, and the main relationships between software modules, while the methodology is presented at the end of the section.

A. *Inputs*

Data set, data streams and features. In this paper we use the term *features* to identify the monitored values that are collected observing the target system. Examples of features are the used memory [32], number of cache accesses [34], or number of network packets received in a time interval [33]. We use the term *data point* to refer to the set of values observed, for all the features and for a given data set, at a given instant of time. For example, in a log file, usually a data point corresponds to a row. We also distinguish between *Training Data* (TD) and the rest of the data set, that we call *Evaluation Data* (ED). Further, we consider only labelled data sets i.e., anomalous data points are explicitly marked. Such labels are not fed to algorithms during training, but are instead needed to compare the effectiveness of different algorithms once trained, or the effectiveness of different parameters' setup for a given algorithm.

Data sets are read using *loaders*, configuration files that contain information about the data sets. Through the loaders, RELOAD gathers raw data and metadata from the data set. At the current state of the implementation, default loaders allow connecting to i) CSV files, ii) ARFF text files, and ii) MySQL databases, as they are commonly used in most applications. A loader should specify the data sets type, structures, and the way the dataset is partitioned into TD and ED. RELOAD could also operate with data streams through the setup of dedicated loaders, which are currently under development. For simplicity, we will only consider data sets in the rest of the discussion.

Feature Selection Strategy(ies). The user should choose one or more strategies for feature selection that he/she wants to apply when gathering data. At the current state of the implementation, the tool allows selecting features through *Variance*, *Pearson Correlation*, *Information Gain* strategies, while learner-based alternatives as the ones based on *Random Forests* are currently under development. Multiple feature selection strategies can be applied sequentially.

Algorithms. Currently, RELOAD includes 10 unsupervised anomaly detection algorithms and 6 sliding window algorithms. The hierarchical organization of super-classes shown in Figure 1 helps adding a new algorithm. Interfaces are built in the tool, allowing the user to extend existing abstract classes mainly implementing two methods responsible of i) how the algorithm performs its initial training (if empty, no training is performed), and ii) calculating the anomaly score for a given data point. Since several algorithms were taken from existing frameworks as ELKI and WEKA, wrappers to call and execute such algorithm e.g., *DataSeriesElkiAlgorithm* class in Figure 1 are already deployed into the tool. When the user chooses which algorithms to execute, he/she can also propose various i) configuration parameters (e.g., number k of relevant neighbours for neighbour-based algorithms [26], [29]) and ii) decision functions [48], to convert numeric scores into boolean, through a configuration file. RELOAD will test all parameters combinations and report on individual scores, ultimately pointing out the more convenient parameters combination.

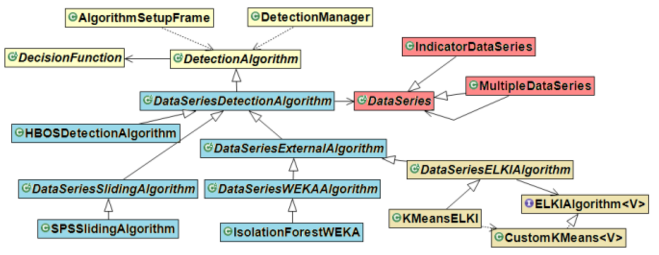


Figure 1. Class Diagram portion: Management of Algorithms in RELOAD.

Except SPS, sliding windows algorithms are simulated sliding versions of existing algorithms LOF, KNN, ABOD, IsolationForests, KMeans, since no public implementations were made available in [42], [43], [44], [45], [46]. Every time a new data point is added to the window, we run the corresponding non-sliding algorithm by using the content of the window as training set. Despite not being optimal in terms of execution time this simulation allows estimating the detection capabilities of non-sliding algorithms when applied to sliding windows.

Metrics. RELOAD includes a broad range of metrics to measure the effectiveness of anomaly detectors [9]. Metrics are correct detections (true positives, TP, and true negatives, TN), missed detections (false negatives, FN), and wrong detections (false positives, FP), as well as the aggregated metrics, precision (P), recall (R), false positive rate (FPR), accuracy (A), F-measure (F1), Matthews coefficient (MCC) and area under ROC curve (AUC). As it can be observed in Figure 2, to introduce a metric, a user must create a new class overriding either *BetterMaxMetric* (the higher, the better), or *BetterMinMetric* interfaces, and adding a field with the metric name to the *enum MetricType*. All available metrics are computed whenever RELOAD is executed with its default configuration, while a target metric should be always selected by the user to rank algorithms and their effectiveness in detecting anomalies.

B. *Main Software Components*

This section reports on the main modules that execute and evaluate anomaly detection algorithms. *GUI*, *Anomaly Checkers Manager*, *Policies Selector* and *Voter* modules are highlighted in Figure 3 by red bold-font labels.

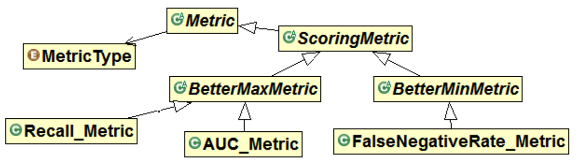


Figure 2. Class Diagram portion: some Metrics used by RELOAD.

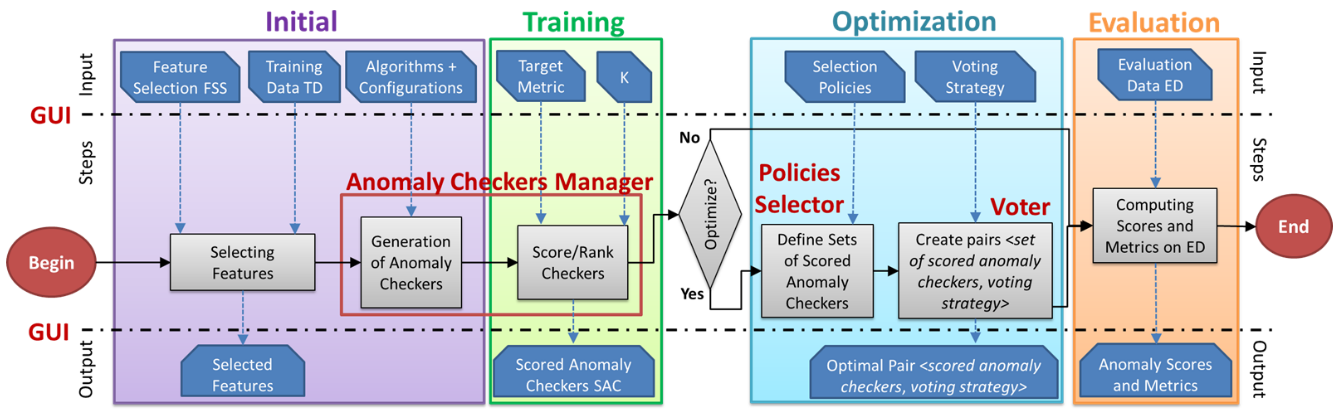

Initial
Training
Optimization
Evaluation
Input
Steps
Output
GUI
GUI
Feature Selection FSS
Training Data TD
Algorithms + Configurations
Target Metric
K
Selection Policies
Voting Strategy
Evaluation Data ED
Input
Steps
Output
Anomaly Checkers Manager
Begin
Selecting Features
Generation of Anomaly Checkers
Score/Rank Checkers
Optimize?
No
Yes
Policies Selector
Voter
Define Sets of Scored Anomaly Checkers
Create pairs <set of scored anomaly checkers, voting strategy>
Computing Scores and Metrics on ED
End
Selected Features
Scored Anomaly Checkers SAC
Optimal Pair <scored anomaly checkers, voting strategy>
Anomaly Scores and Metrics

Figure 3. Workflow of the RELOAD tool.

Anomaly Checkers Manager. It generates and operates a set of *anomaly checkers*: we define an anomaly checker as a unique couple <*feature, algorithm*>. For each data point, an anomaly checker is able to decide if an anomaly is raised i.e., the anomaly checker produces an anomaly score.

The features used by anomaly checkers can also be *composed features* [37]. In fact, the Anomaly Checkers Manager operates to aggregate features using given relations. For example, two features as *bytes sent per second* and *bytes received per second* may be aggregated creating a composed feature as the *byte sent/byte received rate per second*. Filtered features may be aggregated depending on specific rules e.g., always aggregate all the filtered features into a unique composed feature, or if specific properties are met e.g., two filtered features are highly or loosely coupled [19]. Strategies to create composed features in RELOAD are described in Section IV.A.

Policies Selector. This component evaluates anomaly checkers according to a metric. More in detail, each anomaly checker is first applied on a data set (using the Anomaly Checker Manager), and then used to compute a score for a given metric to decide on the most effective ones. As *selection policies*, the Policies Selector includes and applies by default the metrics in [37]:

- BEST *x*: the *x* anomaly checkers that have the best ranking according to a specific metric;
- FILTERED *y*: the *y* anomaly checkers with the best ranking, filtered to avoid more than one anomaly checker built using the same feature;
- THRESHOLD *z*: all the anomaly checkers that reach a threshold *z*, for a given metric e.g., *recall* > 0.6.

Additional selection policies can be included by updating a configuration file.

Voter. The Voter combines the outputs of a given set of *m* anomaly checkers. The Voter applies a *voting strategy* [17], and identifies a data point as anomalous if at least *n* out of *m* anomaly checkers raise an anomaly. The value *n* can be defined through a configuration file; alternatively, the voter contains the following default voting strategies [37]:

- ALL: the data point is considered anomalous only if all the anomaly checkers identifies an anomaly;
- QUARTER/THIRD/HALF: the data point is considered anomalous only if a quarter/third/half of the *m* anomaly checkers identifies an anomaly;
- ONE/1: the data point is evaluated as anomalous if at least one of the *m* anomaly checker raises an anomaly.

GUI. A simple and intuitive Graphical User Interface (GUI) allows importing inputs and showing outputs; details can be found in Section IV.

C. *Methodology and Workflow of the Tool*

The workflow of RELOAD is depicted in Figure 3. From left to right, we can observe four phases, namely *initial*, *training*, *optimization* and *evaluation*. A background process automates the execution of such steps, invoking the components described in Section III.B when needed.

Initial Phase. The operations in the *initial phase* are performed once, when setting up the tool. Training data TD is an input of this phase, as well as the feature selection strategies FSS the user wants to apply. The features are automatically filtered according to FSS, removing those not useful for anomaly detection. Filtered features may then be combined creating composed features; filtered features and composed features build the selected features in Figure 3.

Only the resulting set of *filtered features* will be used in the following steps of the methodology. Algorithms are also an input of this phase: anomaly checkers are built using the selected algorithms and the filtered features. This phase is realized by the Anomaly Checkers Manager and GUI.

Training Phase. The *training phase* produces the list of Scored Anomaly Checkers (SAC) using the TD. First, the input metrics are imported. A specific metric, that we call *target metric*, is selected by the user (through the GUI). It is up to the user to understand the target metric that is more relevant for the system under analysis. Then, multiple instances of each anomaly checker are generated, one for each possible value of algorithm's parameter(s).

At this stage, RELOAD partitions the TD in two sets: *checkers_train* and *checkers_test* e.g., 70%-30% split. Then, it uses the *checkers_train* to train the anomaly checkers, and it uses the *checkers_test* as evaluation of the performance of the anomaly checkers, according to the target metric. This allows associating quantitative values to each anomaly checker: they are now *scored anomaly checkers* (SAC) and can be ordered by score. For example, consider an experiment were two selected features F1 and F2 are considered and used by *kNN* algorithm with possible sizes of the neighbourhood k ϵ {3, 5, 10}. RELOAD generates two anomaly checkers, AC1 = <F1, kNN> and AC2 = <F2,

kNN>, and each checker is instantiated three times: AC1 = {<F1, kNN(3)>, <F1, kNN(5)>, <F1, kNN(10)>} and AC2 = {<F2, kNN(3)>, <F2, kNN(5)>, <F2, kNN(10)>}. Each instance of kNN is trained using *checkers_train*; then, instances of each anomaly checker AC1 and AC2 are scored using *checkers_test* to extract the instance of AC1 and AC2 that guaranteed higher scores according to the target metric.

As a final remark, the user may tune the *k* parameter in Figure 3 to manage the above process through *k-fold cross validation* [38], which is largely adopted in the machine learning domain to reduce biases derived by overfitting the model to the training set.

Optimization Phase. RELOAD users may want to apply selection policies and voting strategies to understand i) the optimal set of the scored anomaly checkers, and ii) the best *voting strategy* to decide if a data point is anomalous. This is achieved from the *optimization phase*, which is optional. It is skipped if RELOAD users are satisfied with executing all the scored anomaly checkers on the ED with no aggregations i.e., having individual results for all the scored anomaly checkers. During optimization, selection policies are applied on the scored anomaly checkers. This allows identifying sets of scored anomaly checkers that satisfy the selection policies. Each of these sets is then matched to the voting strategies. The optimization phase terminates with the definition of pairs <*set of scored anomaly checkers*, *voting strategy*>. This phase is mostly realized thanks to the Policies Selector and the Voter components.

Evaluation Phase. Lastly, all the pairs <*set of scored anomaly checkers*, *voting strategy*> are exercised on the ED, to investigate which data points are anomalous. Resulting anomaly scores are compared with the true labels in the ED, to compute metric scores. This phase is performed mostly by background processes, and the GUI, that shows results.

D. Methodology and Workflow with Sliding Windows

When using sliding windows, the *initial phase* is the same as in Section III.C, with the dimension of the window as an additional input.

After the initial phase, the training phase is started. Typically, sliding window algorithms use a window of *w* data points, and continuously learn whenever a novel data point is progressively acquired. This means that many sliding windows algorithms may not have an initial training phase, and instead they continuously train using the novel data points progressively acquired.

More in detail, sliding windows algorithm are exercised in RELOAD as follows. If the optimization phase is *not* requested, the training and the evaluation phases are iterated in sequence, to add a novel data point to the sliding window, perform the learning phase, decide on the presence of anomalies and compute metrics scores. If the optimization phase is requested, the above procedure is first exercised on the TD to acquire the information for the optimization phase. Then, the optimization phase is executed. After, the algorithm performs as above: the learning phase is performed for all the novel data points on the ED, and the *evaluation phase* is executed to decide on anomalies and compute metrics scores. This iteration continues for the entirety of the data set or until the data stream is closed.

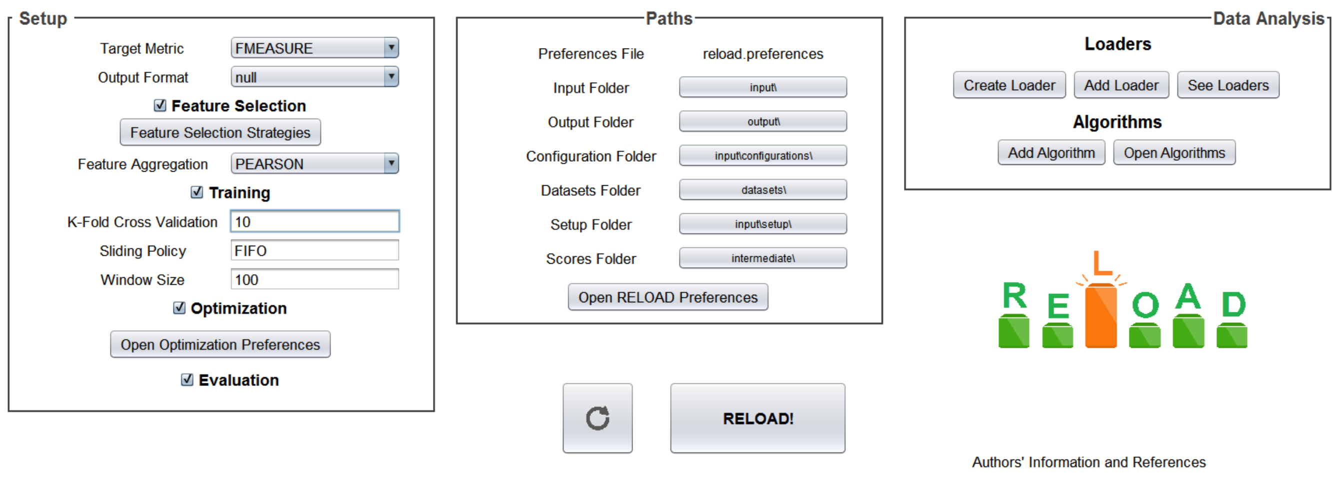
IV. Details about the GUI

The RELOAD GUI helps setting parameters, selecting algorithms and data sets, and checking results. Here we report details about *i*) *setup* see Figure 4, and *ii*) *results* GUIs. The rest of the RELOAD interaction with the user is limited to configuration files e.g., the loaders, and to output files with results.

A. Setup GUI

Setup Box: From top to bottom, the user performs the following actions. First, the user selects the target metric. Second, the output format is selected: the default option *null* provides textual information, while the option *GRAPH* creates summarizing graphs. We do not detail on this feature for brevity, but it just creates bar charts with the x-axis containing the data points, and the y-axis reporting the number of anomaly checkers raising an anomaly (a chart is created for each combination of algorithm, data set, voting strategy and selection policy). The *Feature Selection Strategies* button opens the window in Figure 5. It describes the available feature selection strategies, allowing the user to choose the ones that suit the problem the most. Selecting more strategies leads applying each strategy sequentially according to the order specified by the user in the table.

Strategies to aggregate the features that were selected at



Target Metric: FMEASURE
Output Format: null
☑ Feature Selection
Feature Aggregation: PEARSON
☑ Training
K-Fold Cross Validation: 10
Sliding Policy: FIFO
Window Size: 100
☑ Optimization
☑ Evaluation

Preferences File: reload.preferences
Input Folder: input\
Output Folder: output\
Configuration Folder: input\configurations\
Datasets Folder: datasets\
Setup Folder: input\setup\
Scores Folder: intermediate\

Figure 4. GUI to setup RELOAD.

the previous step include: i) NONE, that does not combine selected features, ii) UNION, the aggregation of all the selected features as a composed feature, iii) SIMPLE, that considers selected features individually (as in NONE) and also the UNION feature, and iv) PEARSON, that considers selected features individually and creates a composed feature if the *Pearson correlation index* between two or more features exceeds a given threshold set by the user.

Training phase heavily relies on the setup of the k parameter for k-fold validation [38]. Furthermore, if the analysis targets sliding window algorithms, the size of the sliding window should be set (field *Window Size*). The *Sliding Policy* field explains how to manage the sliding window, or rather how to decide if we want to add a novel data point and which data point in the window should be discarded. At the moment, two options are implemented: i) *FIFO*, the window always slides, replacing the oldest data point with the current one data set is read from top to bottom, and ii) *FIFO_Normal*, that blocks the sliding mechanism if the current data point is evaluated as anomalous by the algorithm. This last strategy avoids polluting the sliding window with anomalous data points.

Lastly, the selection policies and voting strategies are configured by opening a configuration file with the button *Open Optimization Preferences*: this file enlists the policies to apply e.g., BEST 3, FILTERED 10, along with the voting strategies e.g., ALL, HALF.

Paths Box: First, the user defines the path of the *input* folder, that is the root of all the configuration files and folders described below. Then, the user selects the folders: *i*) *output* , which will contain all the results in CSV format; *ii*) *configuration*, which contains the loaders and the metrics configuration files; iii) *datasets*, that specifies where textual files (if any) of datasets are placed, and *iv*) *setup*, that contains the data sets. Finally, the *score* folder is identified: it is used only for temporary storage of ranked anomaly checkers during computations.

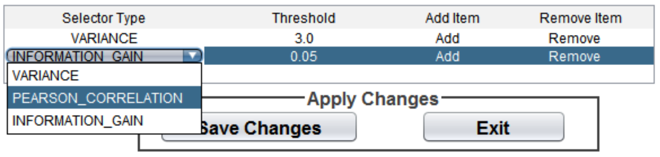
Data Analysis Box: The user selects the algorithms, among the implemented ones, and the data sets, among those in the *setup* folder and that have a loader. When a new data loader has to be defined, as it is the case pressing the "Create Loader" button in the "Data Analysis" box allows to choose a file name and opens a GUI to specify key items of loaders. Depending on the type of the loader e.g., file, database, RELOAD allows the user to choose relevant items to extract data from the chosen data source. Once the data loader is defined, the user can choose the algorithms he wants to apply on such dataset(s). By clicking the "Add Algorithm" button, RELOAD opens a window to allow choosing amongst all available algorithms. Multiple algorithms, and combinations of two or more algorithms, may be selected through the GUI showed in Figure 6. When one or more sliding window algorithms are selected, RELOAD will examine such sliding window algorithm(s) considering the *sliding policy* and *window size* values that the user can set through the GUI showed in Figure 4, box "Setup". On the bottom of the GUI, the *Update* button is used to refresh the interface whenever a configuration is modified, while the *Run* button starts the experiments.

B. Summary and Detailed GUI

RELOAD provides a *Summary GUI*, while producing many files to expand on specific aspects. More in detail, RELOAD creates, for each <dataset, algorithm> couple, files that report on i) the selected features, and the scores they reached on each feature selection strategy, ii) the combined features created, iii) the ranked anomaly checkers, the optimal voting strategy and anomaly threshold, if optimization is executed, and iv) a detailed list of the anomaly scores provided by each anomaly checker used for evaluation, along with the anomaly evaluation generated by the ensemble of anomaly checkers by applying voting strategy and anomaly threshold. In addition, RELOAD creates detailed views of each combination of data loader and algorithm the user selected for the experiment. This *Detailed GUI* shows metric scores of the algorithm on a given dataset by varying selection policy and voting strategy, both for training (and, when selected, optimization) and evaluation phases. Screenshots can be found in Section V.F, along with plots of algorithms' scores.

V. EXERCISING RELOAD

RELOAD was applied to different case studies, especially regarding data logs of service-oriented systems [37], [39]. The tool turned out to be helpful in identifying the more fitting algorithms, either for error or intrusion detection. To show the steps that a generic user has to follow for using the tool, in this section we refer to an entirely new dataset obtained by querying "intrusion detection" and sorting results by "relevance" in the *Kaggle* [40] datasets portal.



Selector Type	Threshold	Add Item	Remove Item
VARIANCE	3.0	Add	Remove
INFORMATION_GAIN	0.05	Add	Remove

Figure 5. GUI for setup of Feature Selection Strategies.

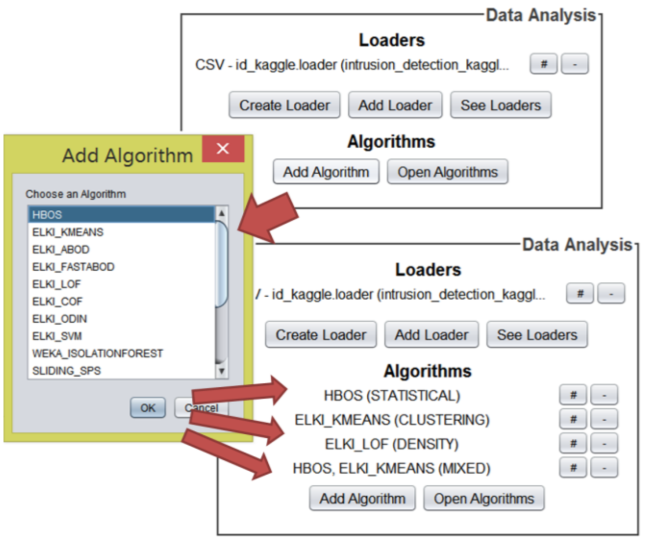


Figure 6. GUI for choosing Algorithms.

A. Checking prerequisites

Java 1.8.0_152 is already installed on our machine. In this case, since we want to download our dataset and the tool from their online repositories [40], [20], we also need internet connection. We remark here that, once the executable and the dataset have been downloaded, the tool is fully standalone and does not require external connections.

B. Analyzing and Refining Dataset

The dataset is partitioned in two CSV files, one for training (TD) and the other one for evaluation (ED). The structure of the two files is the same, except for an unlabelled column in the training set – the first one – that we remove due to the lack of information. The resulting data is structured in 42 columns, with respectively 125.973 and 10.000 data points for train and test set. The last column shows the label for each data point, that can be either *normal* (43.3% of the test set), or representing an attack *dos* (33.3%), *probe* (10.5%), *r2l* (12.0%), or *u2r* (0.9%). Information on the attacks are not reported in the portal, we assume that they cover the same categories as their KDDCup [15] and NSL-KDD [12] datasets. All but 2 columns are numeric, meaning that RELOAD can process them without needing further categorizations.

C. Downloading and Running RELOAD

RELOAD can be downloaded as a ZIP archive from [20]. The ZIP archive includes three items: the JAR file of RELOAD, a preferences file and a folder that contains configuration files. Once files are extracted from the archive, RELOAD can be launched from command line as *java -jar RELOAD.jar*.

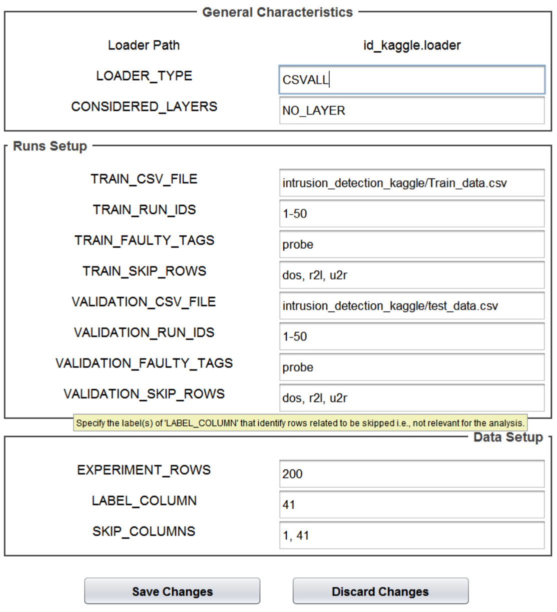
D. Configuring RELOAD

Once started, RELOAD shows the GUI in Figure 2.

Setup Box. Starting from the "Setup" box, we first want to define the reference metric e.g., F-Measure in Figure 4, and strategies to select features and aggregate them to create composed features. For this case study we selected VARIANCE(3) and INFORMATION_GAIN(0.05) feature selection strategies. The other options of the "Setup" box allow choosing how to create composed features, which we execute whenever the *Pearson correlation index* between two or more selected features is more than 0.8 – PEARSON(0.8) -, and to choose which of the phases in Figure 3 the user wants to execute. To the sake of this case study, we will run all the phases of RELOAD, checking all the *Feature Selection*, *Training*, *Optimization* and *Evaluation* checkboxes. We also proceed with a 10-fold validation of the training set as widely suggested [38] in the literature.

Path Box. The "Path" box does not require further adaptations. Only note that the default folder for datasets is specified as a "datasets" subfolder of the current directory. If the datasets the user wants to analyse is located in another folder, the user should either i) change the default path of RELOAD through GUI, or ii) move the files.

Data Analysis Box. Here the user i) defines the data loader(s), and ii) selects algorithms.


General Characteristics

Loader Path	id_kaggle.loader
LOADER_TYPE	CSVALL
CONSIDERED_LAYERS	NO_LAYER

Runs Setup

TRAIN_CSV_FILE	intrusion_detection_kaggle/Train_data.csv
TRAIN_RUN_IDS	1-50
TRAIN_FAULTY_TAGS	probe
TRAIN_SKIP_ROWS	dos, r2l, u2r
VALIDATION_CSV_FILE	intrusion_detection_kaggle/test_data.csv
VALIDATION_RUN_IDS	1-50
VALIDATION_FAULTY_TAGS	probe
VALIDATION_SKIP_ROWS	dos, r2l, u2r

Specify the label(s) of 'LABEL_COLUMN' that identify rows related to be skipped i.e., not relevant for the analysis.

Data Setup

EXPERIMENT_ROWS	200
LABEL_COLUMN	41
SKIP_COLUMNS	1, 41

Figure 7. GUI for setting up Loaders.

A new data loader can be defined by pressing the "Create Loader" button in the "Data Analysis" box. This allows to choose a file name and opens the GUI in Figure 7. We filled the fields of the *id_kaggle.loader* loader in the figure as follows. The CSV files were put in *datasets/intrusion_detection*_kaggle folder: we specified train and test file in the TRAIN_CSV_FILE and VALIDATION_CSV_FILE items. Then, we chose to analyse the performances of RELOAD in identifying probe attacks in this dataset. Therefore, we set the FAULTY_TAGS fields to "probe" and the SKIP_ROWS fields to "dos", "u2r", "r2l", or rather the remaining attacks we are not interested in. We select 50 batches both for training and for validation i.e., RUN_IDS fields, considering batches of 200 data points, as specified by EXPERIMENT_ROWS. Lastly, we specified the true label in the LABEL_COLUMN field, and the columns to be skipped (SKIP_COLUMNS in Figure 7).

To show the versatility of the tool, for this example we selected different algorithms as *HBOS*, *KMeans* and *ODIN*, a sliding window algorithm (*SPS*), and a combination of two algorithms such as HBOS and KMeans.

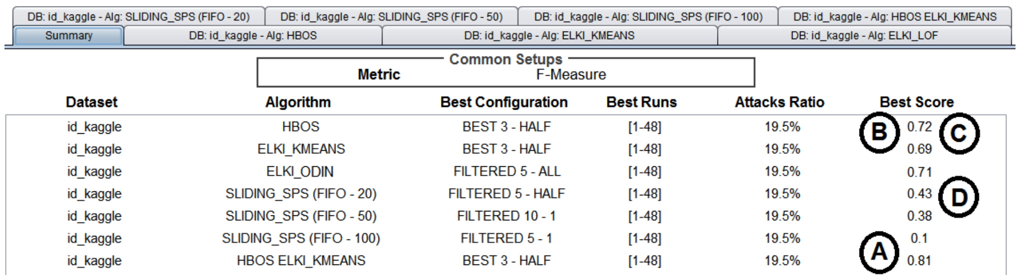
E. Running RELOAD

When everything is set, the user presses the RELOAD! button on the bottom of the GUI. This will start the process of selecting algorithms and executing anomaly detection. When the process completes, RELOAD will open a window that summarizes results, as shown in Figure 8.

F. RELOAD Summary and Detailed GUI

Summary and Detailed GUIs are reported in Figure 8 and Figure 9.

Summary GUI. The first tab to be seen is the "Summary" tab: RELOAD shows a row for each couple <*algorithm*, *data set*> analysed. Each row reports on *i*) the data set, *ii*) the algorithm, *iii*) the most performing pair of selection policy and voting strategy, according to the target metric, *iv*) the percentage of labelled anomalies over all data points in the ED, and *v*) the score of the target metric. This view

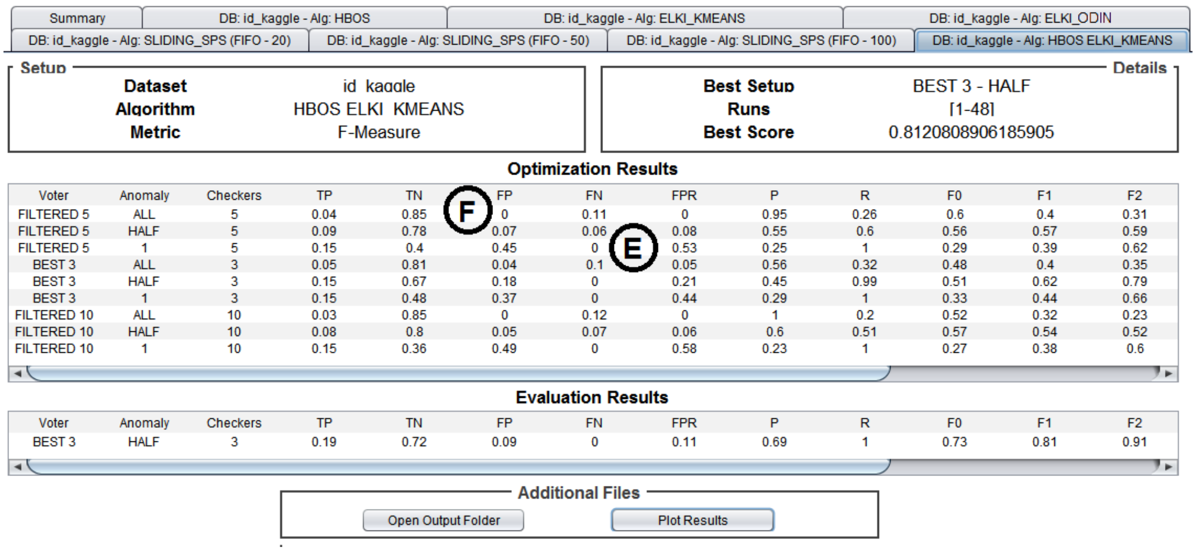

DB: id_kaggle - Alg: SLIDING_SPS (FIFO - 20)
DB: id_kaggle - Alg: SLIDING_SPS (FIFO - 50)
DB: id_kaggle - Alg: SLIDING_SPS (FIFO - 100)
DB: id_kaggle - Alg: HBOS ELKI_KMEANS
Summary
DB: id_kaggle - Alg: HBOS
DB: id_kaggle - Alg: ELKI_KMEANS
DB: id_kaggle - Alg: ELKI_LOF
B
C
D
A

Common Setups

Metric F-Measure

Dataset	Algorithm	Best Configuration	Best Runs	Attacks Ratio	Best Score
id_kaggle	HBOS	BEST 3 - HALF	[1-48]	19.5%	0.72
id_kaggle	ELKI_KMEANS	BEST 3 - HALF	[1-48]	19.5%	0.69
id_kaggle	ELKI_ODIN	FILTERED 5 - ALL	[1-48]	19.5%	0.71
id_kaggle	SLIDING_SPS (FIFO - 20)	FILTERED 5 - HALF	[1-48]	19.5%	0.43
id_kaggle	SLIDING_SPS (FIFO - 50)	FILTERED 10 - 1	[1-48]	19.5%	0.38
id_kaggle	SLIDING_SPS (FIFO - 100)	FILTERED 5 - 1	[1-48]	19.5%	0.1
id_kaggle	HBOS ELKI_KMEANS	BEST 3 - HALF	[1-48]	19.5%	0.81

Figure 8. GUI to summarize results of RELOAD.


Summary
DB: id_kaggle - Alg: HBOS
DB: id_kaggle - Alg: ELKI_KMEANS
DB: id_kaggle - Alg: ELKI_ODIN
DB: id_kaggle - Alg: SLIDING_SPS (FIFO - 20)
DB: id_kaggle - Alg: SLIDING_SPS (FIFO - 50)
DB: id_kaggle - Alg: SLIDING_SPS (FIFO - 100)
DB: id_kaggle - Alg: HBOS ELKI_KMEANS
F
E
Open Output Folder
Plot Results

Setup

Dataset id_kaggle
Algorithm HBOS ELKI_KMEANS
Metric F-Measure

Details

Best Setup BEST 3 - HALF
Runs [1-48]
Best Score 0.8120808906185905

Optimization Results

Voter	Anomaly	Checkers	TP	TN	FP	FN	FPR	P	R	F0	F1	F2
FILTERED 5	ALL	5	0.04	0.85	0	0.11	0	0.95	0.26	0.6	0.4	0.31
FILTERED 5	HALF	5	0.09	0.78	0.07	0.06	0.08	0.55	0.6	0.56	0.57	0.59
FILTERED 5	1	5	0.15	0.4	0.45	0	0.53	0.25	1	0.29	0.39	0.62
BEST 3	ALL	3	0.05	0.81	0.04	0.1	0.05	0.56	0.32	0.48	0.4	0.35
BEST 3	HALF	3	0.15	0.67	0.18	0	0.21	0.45	0.99	0.51	0.62	0.79
BEST 3	1	3	0.15	0.48	0.37	0	0.44	0.29	1	0.33	0.44	0.66
FILTERED 10	ALL	10	0.03	0.85	0	0.12	0	1	0.2	0.52	0.32	0.23
FILTERED 10	HALF	10	0.08	0.8	0.05	0.07	0.06	0.6	0.51	0.57	0.54	0.52
FILTERED 10	1	10	0.15	0.36	0.49	0	0.58	0.23	1	0.27	0.38	0.6

Evaluation Results

Voter	Anomaly	Checkers	TP	TN	FP	FN	FPR	P	R	F0	F1	F2
BEST 3	HALF	3	0.19	0.72	0.09	0	0.11	0.69	1	0.73	0.81	0.91

Additional Files

Figure 9. GUI to expand on the results of the application of HBOS and KMEANS algorithms to the case study.

allows to see which algorithm reached higher metric scores, or rather the optimal algorithm for the dataset or the system under investigation. In addition, for each sliding window algorithm the user selected (SPS in the example), RELOAD will show metric scores for each combination of *selection policy* and *window size* the user set through GUI. As a side note, we can observe how in this case combining anomaly checkers using either HBOS or KMEANS allowed improving metric scores (see mark "A" in Figure 8) with respect to executing either HBOS or KMEANS independently (marks "B" and "C" in the figure). In addition, by looking at SPS results, it is easy to observe that the usage of wider sliding windows does not help SPS in identifying attacks i.e., the wider the window, the lower the F-Measure as pointed out by mark "D" in Figure 8.

Detaied GUI. The GUI shows detail of combinations of data sets and algorithms for the different selection policies and voting strategies. An example is in Figure 9: given the HBOS and KMEANS algorithm on our case study, it shows the metrics computed for the selection policies BEST 3, FILTERED 5, FILTERED 10 and the voting strategies ALL, HALF, 1. It is worth noticing that the voting strategy severely impacts the performance of the algorithm under evaluation. As it is marked as "E" in Figure 9, the 1 strategy reduces the amount of false negatives (FN); instead, achieving consensus among several anomaly checkers composing the ensemble, e.g., the ALL strategy, reduces the number of false alarms (FP), see "F" mark in the figure. This can also be verified from the FN and FP columns in Figure 9 – see FILTERED 5 – ALL that does not have FPs, while FILTERED 5 – 1 does not have FNs.

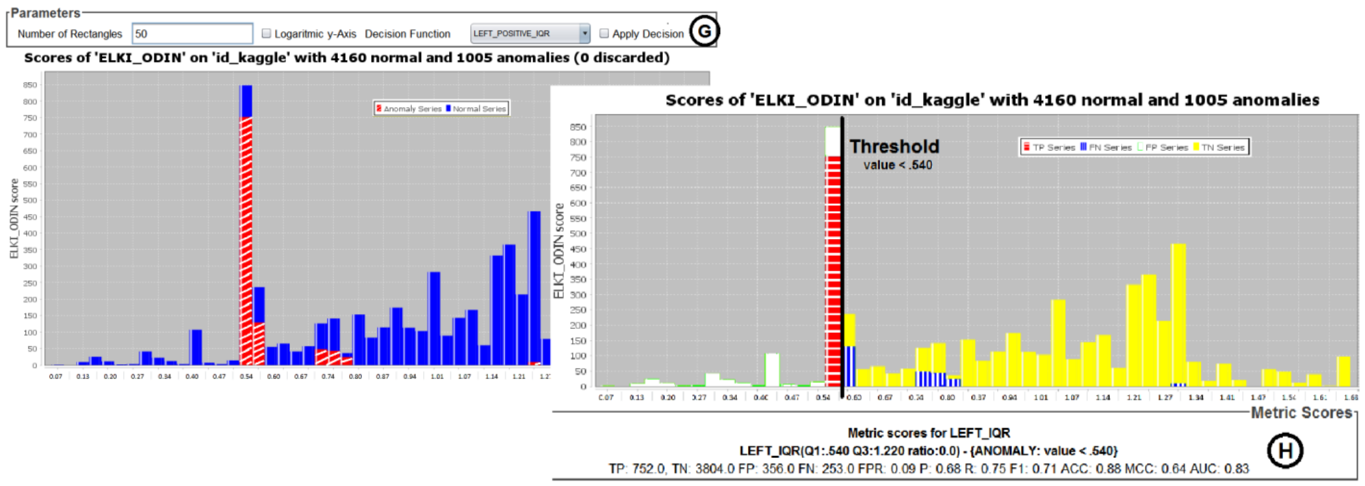
Plots. Ultimately, Figure 10 shows two bar charts that RELOAD outputs when the user presses the "Plot Results" button in Figure 9. More in detail, the figure shows the scores that ODIN algorithm assigns to each data point in the evaluation set. Such scores are grouped in 50 bars, which partition the interval defined by maximum (1.68 in Figure 10) and the minimum ODIN value (0). The height of the bars represents the amount of data points that were scored by ODIN with a value that falls in a given interval.

The bar chart on the left of Figure 10 depicts algorithm scores by painting columns with different colours depending on their true label in the dataset. Then, by using the dropdownlist marked as "G" in the figure is it possible to select a decision function out of the commonly used ones [48] to convert numeric scores into boolean and to calculate TP, TN, FP, FN and related metrics (see mark "H" in Figure 10). This view on the right of Figure 10, painting TP, TN, FP, FN with different patterns, is useful to understand the impact that the choice of the correct decision function has on the final evaluation of algorithm scores.

VI. User Assessment

We estimated the usability of the tool by understanding the perceived difficulties of non-expert users. We involved 16 students of our MsC in Computer Science in a 3-hours experiment. Students have only fundamental knowledge on quantitative assessment of dependable and secure systems.

We organized the 16 students in 8 groups. After that, we provided subsets of the attack data sets KDDCup99, NSL-KDD, ADFA-LD, ISCX2012, and UNSW-NB15. We asked each group to use the first 45 minutes of the experiment to



Parameters

Number of Rectangles: 50 | Logarithmic y-Axis | Decision Function: LEFT_POSITIVE_IQR | Apply Decision

Scores of 'ELKI_ODIN' on 'id_kaggle' with 4160 normal and 1005 anomalies (0 discarded)

Scores of 'ELKI_ODIN' on 'id_kaggle' with 4160 normal and 1005 anomalies

Threshold value < .540

Metric Scores

```
Metric scores for LEFT_IQR
LEFT_IQR(Q1:.540 Q3:1.220 ratio:0.0) - {ANOMALY: value < .540}
TP: 752.0, TN: 3804.0 FP: 356.0 FN: 253.0 FPR: 0.09 P: 0.68 R: 0.75 F1: 0.71 ACC: 0.88 MCC: 0.64 AUC: 0.83
```

Figure 10. Plots of algorithm scores (ODIN) for the evaluation set from the case study of the Kaggle dataset. Red-striped bars on the left figure represent ODIN scores corresponding to attacks in the dataset, while blue solid bars represent ODIN scores occurring with normal data points. On the right, we apply a decision function (data point is anomalous if ODIN score < 0.54, mark G), that allows deriving TP (red horizontal striped bars), TN (yellow solid bars), FP (white bars with green border), FN (blue bars with vertical stripes) and calculating other metrics, see mark H.

examine one data set. Then, we gave a 20-minutes tutorial on the tool. At this point, each group was tasked to apply RELOAD on the data set previously examined. Sample loaders were provided to the students, who had to adapt them for the target data set. Finally, we asked participants to fill the questionnaire in Table II, to assess the cognitive load required to use our tool. All questions were rated on a 5-point semantic Likert [8] scale i.e., 'Very Hard', score 1, to 'Very Easy', score 5, with the possibility to add comments.

All groups were able to terminate the assignment i.e., set appropriate loaders, configure and run RELOAD, and discuss results. Concerning the questionnaire, all answers except Q4 are in the top quartile i.e., the score is above 3.75. This let us conclude that the tool, despite having rooms for improvement, was perceived easy to use. The time needed to perform the experiments was perceived by the students as the main weaknesses of the tool (Q4). However, students focused on the duration of the experiment (which depends on the dimension of the data set and on the computational complexity of algorithms), rather than on the time required to set-up and start experiments. Finally, suggestions regarding the possible improvements of the tool (Q7 and, to a lesser extent, Q2b) were mainly directed to *i*) include additional documentation, ii) improve GUI, and *ii*) create a more intuitive way to define loaders. As the reader could notice, GUI was improved accordingly as presented in Section IV and V, including an easier way to set parameters of the loaders. Regarding documentation, we are currently adding tutorials to our repository to reduce the time needed to understand how to user RELOAD efficiently.

TABLE II. QUESTIONNAIRE AND AGGREGATED ANSWERS.

#	Question	Answers (avg and st.dev)
Q1	Past experience with data mining tools (Yes/No)	15 No, 1 Yes
Q2a	Ease of use of the tool as a whole	3.81 ± 0.54
Q2b	How to improve ease of use	Improving the GUI, options to promptly access help pages.
Q3	Easiness of selecting algorithms and data sets	3.94 ± 0.77
Q4	Time needed to calculate results	2.47 ± 0.92
Q5	Understandability of the results	4.06 ± 0.77
Q6	Completeness of the results	3.87 ± 0.64
Q7	Suggestions to improve RELOAD (open question)	Students commented on documentation, non-intuitive loaders, and using GPU cores.

VII. Conclusions

This paper presented RELOAD, an intuitive and open source tool [20] specifically tailored for the evaluation of unsupervised anomaly detection algorithms in the domain of dependable and secure systems.

RELOAD automates the selection of the most relevant features out of a data set to reduce the amount of data to be analyzed. Further, it includes built-in metrics for the evaluation and contains several unsupervised algorithms, which are often deemed the most useful [1], [36] for detecting unexpected attacks or failures. Additionally, it facilitates the execution of sliding window algorithms, that are particularly relevant for dynamic systems whose expected behavior changes through time and do not allow relying on training data [21]. The tool is shaped to be used also by non-expert people that want to learn the basics of the domain, as well as from practitioners that want to estimate detection capabilities of different anomaly detection algorithms on a pool of existing or custom datasets.

In addition, the tool provides sample parameters configurations of selected algorithms to be used. Support for run-time evaluations is currently under investigation and requires setting adequate data-stream loaders for the tool. Finally, although it is widely extensible, RELOAD includes a large set of built-in configurations, which are expected to satisfy most of the necessity of possible RELOAD users with reduced learning time.

References

[1] Chandola, V., Banerjee, A., & Kumar, V. (2009). "Anomaly detection: A survey". ACM computing surveys (CSUR), 41(3), 15.

[2] Modi, Chirag, et al. "A survey of intrusion detection techniques in cloud." Journal of Network and Computer Appl. 36.1 (2013): 42-57.

[3] Salfner, F., Maren L., Malek M. "A survey of online failure prediction methods." ACM Computing Surveys (CSUR) 42.3, 2010.

[4] Schubert, E., Koos, A., Emrich, T., Züfle, A., Schmid, K. A., & Zimek, A. (2015). A framework for clustering uncertain data. Proceedings of the VLDB Endowment, 8(12), 1976-1979.2010): 10.

[5] Rapid Miner tutorial, https://rapidminer.com/get-started/ [online, last accessed 5th May 2019]

[6] Scikit tutorial, http://scikit-learn.org/stable/documentation.html [online, last accessed 5th May 2019]

[7] Weka, https://www.cs.waikato.ac.nz/ml/index.html [online, last accessed 5th May 2019]

[8] Friborg, Oddgeir, Monica Martinussen, and Jan H. Rosenvinge. "Likert-based vs. semantic differential-based scorings of positive psychological constructs: A psychometric comparison of two versions of a scale measuring resilience." Personality and Individual Differences 40.5 (2006): 873-884

[9] Sokolova M., Japkowicz, S. "Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation" AI 2006 Springer Berlin Heidelberg, 1015-21.

[10] Friedman, Jerome H. "Stochastic gradient boosting." Computational Statistics & Data Analysis 38.4 (2002): 367-378.

[11] Bovenzi, A., Brancati, F., Russo, S., & Bondavalli, A. (2015). An os-level framework for anomaly detection in complex software systems. IEEE Transactions on Dependable and Secure Computing, 12(3), 366-372.

[12] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in Computational Intelligence for Securit and Defense Applications, 2009. CISDA 2009. IEEESymposium on. IEEE, 2009, pp. 1–6

[13] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark data sets for intrusion detection,"Computers & Security, vol. 31, no. 3, pp. 357–374, 2012.

[14] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in Military Communications and Information Systems Conference (Mil-CIS), 2015. IEEE, 2015, pp. 1–6.

[15] S. Rosset and A. Inger, "Kdd-cup 99: knowledge discovery in a charitable organization's donor database." SIGKDD Explorations, 1(2), 85-90.

[16] G. Creech and J. Hu, "Generation of a new ids test data set: Time to retire the kdd collection," in Wireless Communications and Networking Conf (WCNC), 2013 IEEE. IEEE, 2013, pp. 4487–4492.

[17] Di Giandomenico, F., and Strigini, L. "Adjudicators for diverse-redundant components". In Reliable Distributed Systems, 1990. Proceedings., Ninth Symposium on (pp. 114-123). IEEE.

[18] M. Goldstein and S. Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. PloS one, 11(4):e0152173, 2016.

[19] Saeys, Y., Abeel, T., & Van de Peer, Y. (2008, September). Robust feature selection using ensemble feature selection techniques. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 313-325). Springer, Berlin, Heidelberg.

[20] RELOAD Wiki – Github. github.com/tommyippoz/RELOAD/wiki [online, last accessed 25th July 2019]

[21] L. Zhang, J. Lin, and R. Karim, "Sliding window-based fault detection from high-dimensional data streams", IEEE Transactions on Systems, Man, and Cybernetics, vol. 47, no. 2, pp. 289–303, 2017.

[22] Pannu, H. S., Jianguo Liu, and Song Fu. "A self-evolving anomaly detection framework for developing highly dependable utility clouds." Global Communications Conf (GLOBECOM), 2012 IEEE.

[23] Cherkasova, L., et al.: Anomaly application change or workload change? towards automated detection of application performance anomaly and change. DSN 2008, 452–461 (2008)

[24] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: identifying density-based local outliers. In ACM sigmod record, volume 29, pages 93–104. ACM, 2000.

[25] M. Goldstein and A. Dengel. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. 2012. KI-2012: Poster and Demo Track, 59-63.

[26] J. Tang, Z. Chen, A. W.-C. Fu, and D. W. Cheung. Enhancing effectiveness of outlier detections for low density patterns. In Pacific-Asia Conference on Knowledge Discovery and Data Mining, pages 535–548. Springer, 2002.

[27] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In ACM Sigmod Record, volume 29, pages 427–438. ACM, 2000.

[28] M. Amer, M. Goldstein, and S. Abdennadher. Enhancing one-class support vector machines for unsupervised anomaly detection. In Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description, pages 8–15. ACM, 2013.

[29] H.-P. Kriegel, A. Zimek, et al. Angle-based outlier detection in high-dimensional data. In Proceedings of the 14th ACM SIGKDD Int. Conference on Knowledge discovery and data mining, pages 444–452. ACM, 2008.

[30] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In Data Mining, 2008. ICDM'08. Eighth IEEE Int. Conference on, pages 413–422. IEEE, 2008.

[31] Tang, J., Chen, Z., Fu, A. W. C., & Cheung, D. (2001). A robust outlier detection scheme for large data sets. In In 6th Pacific-Asia Conf. on Knowledge Discovery and Data Mining.

[32] Dupont, Laurent, et al. "Continuous anomaly detection based on behavior modeling and heterogeneous information analysis." U.S. Patent Application No. 12/941,849.

[33] Giotis, Kostas, et al. "Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments." Computer Networks 62 (2014): 122-136.

[34] Chiappetta, M., Erkay S., and Cemal Y.. "Real time detection of cache-based side-channel attacks using hardware performance counters." Applied Soft Computing 49 (2016): 1162-1174.

[35] Breiman, Leo. "Random forests". Machine learning 45.1 (2001): 5-32

[36] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava. A comparative study of anomaly detection schemes in network intrusion detection. In Proceedings of the 2003 SIAM Int. Conference on Data Mining, pages 25{36. SIAM, 2003.

[37] Zoppi, Tommaso, Andrea Ceccarelli, and Andrea Bondavalli. "MADneSs: a Multi-layer Anomaly Detection Framework for Complex Dynamic Systems." IEEE Transactions on Dependable and Secure Computing (2019). DOI 10.1109/TDSC.2019.2908366

[38] Rodriguez, Juan D., Aritz Perez, Jose A. Lozano. "Sensitivity analysis of k-fold cross validation in prediction error estimation." IEEE Trans. on pattern analysis and machine intelligence 32.3 (2010): 569-575.

[39] Falcão, F., Zoppi, T., Silva, C. B. V., Santos, A., Fonseca, B., Ceccarelli, A., & Bondavalli, A. (2019, April). Quantitative comparison of unsupervised anomaly detection algorithms for intrusion detection. In Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing (pp. 318-327). ACM.

[40] Kaggle – "Intrusion Detection" dataset uploaded by Jinner, https://www.kaggle.com/what0919/intrusion-detection [online, last accessed 5th May 2019]

[41] OpenML portal, https://www.openml.org/ [online, last accessed 5th May 2019]

[42] Zhou, A., Cao, F., Qian, W., & Jin, C. (2008). Tracking clusters in evolving data streams over sliding windows. Knowledge and Information Systems, 15(2), 181-214.

[43] Karimian, S. H., Kelarestaghi, M., & Hashemi, S. (2012, May). I-inclof: improved incremental local outlier detection for data streams. In Artificial Intelligence and Signal Processing (AISP), 2012 16th CSI International Symposium on (pp. 023-028). IEEE.

[44] Zhang, Liangwei, Jing Lin, and Ramin Karim. "Sliding window-based fault detection from high-dimensional data streams." IEEE Transactions on Systems, Man, and Cybernetics: Systems 47.2 (2017): 289-303.

[45] Mouratidis, K., & Papadias, D. (2007). Continuous nearest neighbor queries over sliding windows. IEEE transactions on knowledge and data engineering, 19(6), 789-803.

[46] Ding, Z., & Fei, M. (2013). An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window. IFAC Proceedings Volumes, 46(20), 12-17.

[47] He, S., Zhu, J., He, P., & Lyu, M. R. (2016, October). Experience report: System log analysis for anomaly detection. In 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE) (pp. 207-218). IEEE.

[48] Ali, Shawkat, and Kate A. Smith. "On learning algorithm selection for classification." Applied Soft Computing 6.2 (2006): 119-138.

[49] Zoppi, T., Ceccarelli, A., & Bondavalli, A. (2017, April). Exploring anomaly detection in systems of systems. In Proceedings of the Symposium on Applied Computing (pp. 1139-1146). ACM.