An uncertainty-managing batch relevance-based approach to network anomaly detection

Gianni D'angelo[a], Francesco Palmieri[b,*], Massimo Ficco[c], Salvatore Rampone[a]

[a]*Department of Sciences and Technologies, Sannio University, Via dei Mulini 59A, I-82100 Benevento, Italy.*
[b]*Department of Computer Science, University of Salerno, Via Giovanni Paolo II, 132, I-84084 Fisciano (SA), Italy.*
[c]*Department of Industrial and Information Engineering, Second University of Naples, Via Roma 29, I-81031 Aversa (CE), Italy.*

Abstract

The main aim in network anomaly detection is effectively spotting hostile events within the traffic pattern associated to network operations, by distinguishing them from normal activities. This can be only accomplished by acquiring the a-priori knowledge about any kind of hostile behavior that can potentially affect the network (that is quite impossible for practical reasons) or, more easily, by building a model that is general enough to describe the normal network behavior and detect the violations from it. Earlier detection frameworks were only able to distinguish already known phenomena within traffic data by using pre-trained models based on matching specific events on pre-classified chains of traffic patterns. Alternatively, more recent statistics-based approaches were able to detect outliers respect to a statistic idealization of normal network behavior. Clearly, while the former approach is not able to detect previously unknown phenomena (zero-day attacks) the latter one has limited effectiveness since it cannot be aware of anomalous behaviors that do not generate significant changes in traffic volumes. Machine learning allows the development of adaptive, non-parametric detection strategies that are based on "understanding" the network dynamics by acquiring through a proper training phase a more precise knowledge about normal or anomalous phenomena in order to classify and handle in a more effective way any kind of behavior that can be observed on the network. Accordingly, we present a new anomaly detection strategy based on supervised machine learning, and more precisely on a batch relevance-based fuzzyfied learning algorithm, known as U-BRAIN, aiming at understanding through inductive inference the specific laws and rules governing normal or abnormal network traffic, in order to reliably model its operating dynamics. The inferred rules can be applied in real time on online network traffic. This proposal appears to be promising both in terms of identification accuracy and robustness/flexibility when coping with uncertainty in the detection/classification process, as verified through extensive evaluation experiments.

Keywords: Network Anomaly Detection, Machine Learning, Supervised Classification, Fuzzy-based techniques, Inductive Inference.

*Corresponding author.
Email addresses: dangelo@unisannio.it (Gianni D'angelo), fpalmieri@unisa.it (Francesco Palmieri), massimo.ficco@unina2.it (Massimo Ficco), rampone@unisannio.it (Salvatore Rampone)

1. Introduction

Together with the astonishing deployment of network technologies and the consequent increment in traffic volumes, the importance of network misuse detection and prevention frameworks is proportionally growing in almost all the modern organizations, in order to protect the most strategic resources from both external and internal threats. In this scenario, the task of identifying and categorizing network anomalies essentially consists in determining all the circumstances in which the network traffic pattern deviates from its normal behavior, that in turn depends on multiple elements and considerations associated to the activities taking place every day on the network.

However, the main difficulty related to a really effective detection is associated to the continuous evolution of anomalous phenomena, due to the emergence of new previously unknown attacks, so that achieving a precise, stable and exhaustive definition of anomalous behavior, encompassing all the possible hostile events that can occur on a real network, is practically impossible. Nevertheless, detection systems must not be limited by the a priori knowledge of a specific set of anomalous traffic templates or be conditioned by a large number of complex operating parameters (e.g., traffic statistic distributions and alarm thresholds), and hence have to be able to recognize and directly classify any previously unknown phenomenon that can be experienced on the network. As a consequence, the ultimate goal of modern anomaly detection systems is behaving in a adaptive way in order to flag in "real-time", all the deviations from a model that is built dynamically and in an incremental way by capturing the concept of normality in network operations according to a learning-by-example strategy. These new systems, overcoming the known limitations of the more traditional ones based on pattern detection and statistical analysis, are empowered by flexible machine learning techniques.

Accordingly, we propose a novel anomaly detection strategy, particularly suitable for IP networks, based on supervised machine learning, and more specifically on a batch relevance-based fuzzyfied learning algorithm known as U-BRAIN.

This strategy aims at understanding the processes that originate the traffic data, by deriving the specific laws and rules governing it, in order to reliably model its underlying dynamics. This is accomplished by performing inductive inference (or better, generalization) on traffic observations, based on some empirical pre-classified "experiential" (training) data, representing incomplete information about the occurrence of specific phenomena that describe normal or anomalous network activities. In addition, the adopted learning scheme allows a certain degree of uncertainty in the whole detection process making the resulting framework more solid and flexible in managing the large variety and complexity of real traffic phenomena. Then the inferred rules can be applied in real time on online network traffic.

We evaluated the effectiveness of the presented detection framework within a widely known test case scenario, in order to make the achieved results comparable with those of other proposal available in literature. These results demonstrated a quite satisfactory identification accuracy by placing our strategy among the most promising state-of-the-art proposals.

2. Background and Related Work

Network anomaly detection has gained a great attention in security research with about 40 years of experiences available in literature. The first approach to automatic detection has been proposed in [1], followed by a large number of contributions exploring many other solutions and proposals [2][3][4].

The earliest and more traditional detection approaches, mainly aiming at spotting intrusion activities, work by matching specific traffic patterns, gathered from the packets under observation, against a list of predefined signatures, each associated to a known attack or hostile/anomalous behavior. Some well-known examples are SNORT [5] and BRO [6]. While ensuring very good response times and a quite satisfactory degree of effectiveness in case of previously known menaces, these approaches are almost totally clueless in presence of new (zero-day) attacks, or when, due to minor modifications in its behavior, an already known attack does not closely match the associated signatures. In both the cases new up-to-date signatures must be generated and added to the list as soon as more detailed information about the hostile behavior become available. Unfortunately, this implies human intervention, and hence too much time to ensure real-time response.

Other very common detection systems are based on a statistical idealization of the network behavior and process the traffic observations through statistical-analysis techniques by flagging the outliers as anomalous events. The most significant examples are NIDES (Next-Generation Intrusion Detection Expert System) [7], an hybrid system providing a statistical analysis engine, and SPADE (Statistical Packet Anomaly Detection Engine) [8] a statistical detection system based on determining anomaly scores, available as a plug-in for SNORT. However, while straightforward and robust in their formulation (they do not require prior knowledge of the security menaces nor need packet inspection), statistic detection approaches may result too simplistic in their basic assumptions and hence scarcely reliable in their results. In fact, being based only on the statistical properties of the involved traffic flows, these approaches are too sensitive to the normality assumption, and really effective only against specific phenomena that imply significant variations in the statistical properties of the network traffic (Volume-based Attacks). More precisely, such detection techniques have to be based on extremely accurate statistical distributions that describe the traffic under observation. Unfortunately, modeling real network traffic, typically characterized by an inhomogeneous usage pattern, by using only pure statistical methods, may result in a poor choice in terms of real effectiveness. Furthermore, these solutions cannot be aware of hostile activities that only affect the packet contents (such as stack smashing or other kind of malicious code exploiting system/services vulnerabilities) or explicitly conceived to be undistinguishable from regular user activities (e.g., low-rate DoS attacks).

As an alternative that may reveal extremely effective in coping with the above challenges, machine learning provides fully automated detection capabilities, by allowing a system to learn by example what are the anomalous events occurring in the observed traffic. It also allows improving the detection performance over time with experience, as more and more examples (or training data), describing normal or anomalous behaviors, are provided in its knowledge base. In this way, the detection function, that is essentially a binary classifier working on the normal and anomalous traffic classes, is inferred from the aforementioned training data. Such data consist of a set of pre-classified traffic samples. In supervised learning, each sample is a pair consisting of an input object (typically a vector of traffic features) and a desired output value (the class value) also called the supervisory signal. The inferred classifier should assign the right output class value to any valid input sample. This implies that the learning paradigm should be reasonably capable to perform generalization from the knowledge contained in the training data to previously unseen situations.

The use of machine learning in anomaly detection, with the development of generalization capabilities from past experiences for classifying future data as normal or anomalous has been exploited in many proposals [9], based on neural networks [10] [11], SVMs [12] and data mining techniques [13][14]. These approaches can be further subdivided into generative or discriminative. A typical generative approach (e.g., [15]) constructs a model by starting only from normal training examples, and then evaluate several test instances in order to appreciate how well they fit such model. As an example, the ideas presented in [16] explore different machine learning techniques to construct detection models from past behavior. On the other hand, discriminative techniques (e.g., [12]), attempt to understand the difference between the normal and anomalous instance classes. A learning approach for reproducing packet level alerts for anomaly detection at the flow level has been presented in [17]. Several approaches rely on clustering techniques, such as ADWICE [18], performing unsupervised detection based on a fast incremental clustering technique. K-Means+ID3 [19], instead, is a supervised learning approach combining k-Means clustering and the ID3 decision trees in order to classify anomalous and normal network activities. Regarding the use of tree-based structures, the DGSOT+SVM [20] scheme is an iterative approach leveraging the dynamic generation of self-organizing hierarchical trees together with SVMs to be trained on the tree nodes, where support vectors are used as the basic knowledge to control the tree growth. Also non-linear analysis, combined with recurrence quantification techniques [51] has been used to construct a flexible detection approach based on understanding the most hidden traffic dynamics by simultaneously observing the traffic behavior on multiple time scales.

Other approaches introduced the concept of uncertainty within the learning strategy. Notably, a detection solution based on weighted fuzzy matching over frequent episode rules is presented in [21]. In addition, the approach presented in [22] applied fuzzy logic-based rules to audit data in order to classify it as normal or anomalous.

Starting from the above experience, our machine-learning based detection solution combines the strength of rule-based systems and the flexibility of fuzzy logic to reliably understand the fundamental properties of the network traffic in order to rapidly flag the occurrence of abnormal events.

3. A fuzzy rule-based detection strategy

The basic idea is building a formal model that expresses the relations between all the fundamental variables involved in the traffic dynamics, and hence "understands" the notions of normal and anomalous behavior from the available experience by learning the characteristics of the corresponding traffic classes and expressing them into laws and rules that are general enough to determine if any unseen instance belongs to the one or the other class. Obviously, the overall detection quality strongly depends on the accuracy and generality of the above model and hence on the completeness of the training data on which its "knowledge" about normal and anomalous phenomena is built.

Starting from the previous considerations, we modeled our anomaly detection strategy according to a supervised machine learning scheme, specifically conceived for learning disjunctive normal form (DNF) [23] boolean formulas from partial truth tables, possibly with uncertain values or missing information bits. Such formulas, determined by using the U-BRAIN (Uncertainty-managing Batch Relevance-based Artificial Intelligence) algorithm [24], describe the correlation rules between attribute conditions and class labels, modeling the normal and anomalous traffic profiles through boolean predicates on the observation attributes. Since the U-BRAIN algorithm works on boolean data, the above attribute values have to be quantized, in order to represent them in rules by using binary strings.

The aggregation of all the determined correlation rules defines the behavior of an inductively learned classifier that is able to analyze the deviation from normal traffic profiles and the proximity to the known anomalous ones, as determined from the historical observations available in the training set. Clearly, by relying on a supervised approach where both the phenomena of interest are known in the training set, such a classifier should be potentially able to achieve better detection performance respect to semi-supervised and unsupervised solutions, since more information is available in its training knowledge base. However, the unbalancing in the amount of training data representing the two classes, together with the presence of some uncertainty and noise in pre-classified samples, can significantly affect the final accuracy of the detection results. U-BRAIN faces uncertainty problems by improving the flexibility of rules through the introduction of fuzzy logic. This characteristic, allowing a certain degree of uncertainty at the binary classifier level, makes the resulting detection strategy significantly stronger both in terms of flexibility and ability to cope with the variety and complexity of network traffic phenomena. In this way the detection approach considers a wider range of implications when making its decisions, resulting in a more effective and powerful approach in presence of incomplete training data.

3.1. Initial knowledge construction

A fundamental preliminary task in a supervised network anomaly detection system is the initial "knowledge construction" where the most significant network utilization patterns, describing the fundamental traffic dynamics, should be described in terms of specific features gathered from traffic observations. This is a very slow and complex activity that implies collecting and pre-classifying network traffic observations over a sufficiently long period of time, in order to build a quite complete training set, reliably describing the historical knowledge of the network behavior, or "baseline". The training data will thus consist in a parametric network traffic model that can be viewed as an approximation of reality, where a limited set of parameters is available in form of the most discriminant traffic features that are able to describe the traffic behavior. Such traffic model can be realized by looking at different observation dimensions, such as inter-arrival times of packet transmission and reception events, together with information about packet sizes, flags, source and destination addresses, ports/services involved, by also attempting to use the memory of recent past to identify persistent events like end-to-end connections (traffic flows). These observations are represented as a time series, that is, a sequence of scalar samples measured at uniformly spaced time intervals. More formally, the training set $T = (t_1, t_2, \dots t_N)$ consists of N samples, each structured as an d-dimensional vector of traffic features $s_i = (f_1, f_2, \dots f_d)$. The training set is joined with a one-dimensional feature set $C = (c_1, c_2, \dots c_m)$ associated to the supervisory signal, where c_i represent the class (i.e., anomalous, not anomalous) to which each sample t_i belongs. In this way, the knowledge base of the classifier is described by a set or rules that will be "*inferred*" from the aforementioned collection of pre-classified training samples, where p of them are associated to anomalous traffic observation and q to normal ones, with $p \ll q$.

3.2. The U-BRAIN algorithm

U-BRAIN is a machine learning algorithm able to infer explicitly the laws that govern a process from examples. In its latest version, U-BRAIN can also act on incomplete data. Originally, the algorithm, named BRAIN [25], was conceived for recognizing the borders of coding regions in human DNA. The algorithm was based on the Michalski's STAR technique [26], on the candidate-elimination method introduced by Mitchell [27], and on the work of Haussler [28]. The BRAIN algorithm was later extended [24] to treat data with missing bits by using fuzzy sets. The resulting U-BRAIN algorithm keeps the computational complexity of the original one. The great versatility that characterizes it, makes U-BRAIN applicable in every industry and science field in which there are data to be analyzed, such as the financial world, the aviation industry as well as the biomedical scope. U-BRAIN models a process by a formula able to forecast the future process behavior. Specifically, the algorithm builds Boolean formulae F of n literals x_i ($i = \{1, \dots, n\}$) in DNF form, made up of disjunctions of conjunctive terms, starting from a set T of training data. The data (instances) in T are divided into two classes, named positive and negative, respectively modeled by the n-sized vectors $\vec{u_i}$ with $i = \{1, \dots, p\}$ and $\vec{v_j}$ with $j = \{1, \dots, q\}$, representing the issues to be classified. Each element $u_{i,k}$ or $v_{j,k}$ with $k = \{1, \dots, n\}$ can assume values belonging to the set $\{0.1, \frac{1}{2}\}$ respectively associated to positive, negative and uncertain values. The conjunctive terms of the formula are carried-out in an iterative way by two nested loops (see algorithm 1 schema). The inner cycle refers to the selection of the literals of each formula term, while the outer one is devoted to the terms themselves. In order to build a formula consistent with the given data, U-BRAIN compares each given positive instance with each negative one and builds a family of fuzzy sets of conditions that must be satisfied by at least one of the positive instances and violated by all the negative ones formally defined as:

$$S_{i,j} = \left\{ x_k | u_{i,k} > v_{j,k} \vee u_{i,k} = v_{j,k} = \tfrac{1}{2} \right\} \cup \left\{ \overline{x_k} | u_{i,k} < v_{j,k} \vee u_{i,k} = v_{j,k} = \tfrac{1}{2} \right\} \quad (1)$$

In other words, the k-th literal belongs to the $S_{i,j}$ set if the elements in the position k, belonging to the i-th positive instance $u_{i,k}$ and to the j-th negative instance $v_{j,k}$, are different or both equal to $\frac{1}{2}$.

Starting from these sets $S_{i,j}$, the algorithm determines for each literal x_k, belonging to them, a set of coefficients $R_{i,j}$, R_i and R, called relevances, forming a probability distribution:

$$\begin{aligned} R_{i,j}(x_k) &= \frac{\mu_{i,j}(x_k)}{\#(S_{i,j})} \\ R_i(x_k) &= \frac{1}{q} \sum_{j=1}^{q} R_{i,j}(x_k) \\ R(x_k) &= \frac{1}{p} \sum_{j=1}^{o} R_i(x_k) \end{aligned} \quad (2)$$

where $\mu_{i,j}$ is the membership function of the set $S_{i,j}$

$$\mu_{i,j}(x_k) = \begin{cases} 1 & \text{if } u_{i,k} = 1 \text{ and } v_{j,k} = 0 \\ \left(\frac{1}{2}\right)^{(p+q)} & \text{if } u_{i,k} > v_{j,k} \text{ and } \left(u_{i,k} = \frac{1}{2} \text{ or } v_{j,k} = \frac{1}{2}\right) \\ \left(\frac{1}{2}\right)^{(p+q+1)} & \text{if } u_{i,k} = \frac{1}{2} \text{ and } v_{j,k} = \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_{i,j}(\overline{x}_k) = \begin{cases} 1 & \text{if } u_{i,k} = 0 \text{ and } v_{j,k} = 1 \\ \left(\frac{1}{2}\right)^{(p+q)} & \text{if } u_{i,k} < v_{j,k} \text{ and } \left(u_{i,k} = \frac{1}{2} \text{ or } v_{j,k} = \frac{1}{2}\right) \\ \left(\frac{1}{2}\right)^{(p+q+1)} & \text{if } u_{i,k} = \frac{1}{2} \text{ and } v_{j,k} = \frac{1}{2} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

and where $\#(x)$ is the fuzzy cardinality of a subset s of a set S defined as $\#(x) = \sum_{x \in S} \mu_s(x)$. The membership function, defined in (3), is conceived in such a way that, in a comparison between two instances, the certain values have a much prominent role w.r.t. the missing ones while uncertain information may gain relevance only if the certain one is not sufficient (see [24]).

This allows the selection of the literals based on a maximum probability greedy criterion (the literal having maximum relevance value is selected).The goal of such greedy selection is simultaneously covering the maximum number of positive instances with the minimum possible number of literals. Each time a literal is chosen, the condition sets $S_{i,j}$, and the corresponding probability distribution, are updated by erasing the sets containing the literal itself. The inner cycle is then repeated and the term is completed when there are no more elements in the sets of conditions. Then the new term is added to the formula and, in the outer cycle, the positive instances satisfying the term are erased. Then, the inner cycle starts again on the remaining data. The algorithm ends when there are no more data to treat. The algorithm has two biases: the instance set must be self-consistent, that means that an instance cannot belong to both the classes, and no duplicated instances are allowed. In fact, it may happen that the initial set of training instances contains redundant information. This may be due to repeated instances present from the beginning of the process or resulting from a reduction step, whose task is limiting the presence of missing bits, by recovering them as possible. Such redundancy is automatically removed by keeping each instance just once and deleting all the repetitions, in order to avoid consistency violation that can halt the process.

Algorithm 1 The U-BRAIN schema

```
Require: $p > 0$. $q > 0$. $T = \{\vec{u_1}, \dots, \vec{u_p}, \vec{v_1}, \dots, \vec{v_q}\}$
1: $F \leftarrow \varnothing$ {Initialize $F$}
2: while there are positive instances $\vec{u_i} \in T$ do
3:     Uncertainty Reduction
4:     Repetition Deletion
5:     $m \leftarrow \varnothing$ {Initialize term $m$}
6:     Build $S_{i,j}$ sets from $T$
7:     while there are elements in $S_{i,j}$ do
8:         Compute $R_{i,j}$ for $\{x_k, \overline{x}_k\}, k = \{1, \dots, n\}$
9:         Compute $R_i$ for $\{x_k, \overline{x}_k\}, k = \{1, \dots, n\}$
10:        Compute $R$ for $\{x_k, \overline{x}_k\}, k = \{1, \dots, n\}$
11:        Choose literal $x$ with max relevance $R$
12:        $m \leftarrow m \cup \{x\}$ {Update term $m$}
13:        Update $S_{i,j}$ sets
14:    end while
15:    $F \leftarrow F \cup \{m\}$ {Add term $m$ to $F$}
16:    Update positive instances $\vec{u_i} \in T, i = \{1, \dots, p\}$
17:    Update negative instances $\vec{v_j} \in T, j = \{1, \dots, q\}$
18:    Check consistency
19: end while
20: return $F$
```

3.3. U-BRAIN Algorithm Complexity and its Parallel implementation

The overall algorithm's time complexity is $O(n^5)$ and its space complexity is $O(n^3)$ for large n (where n is the number of variables). So, processing large amounts of data with a single computer may be prohibitive in terms of both space and time needed. Of course, such a complexity is only referred to the training phase where the set of classification rules is initially built from the training data. Once these rules are available the detection activity is extremely simple and fast and hence can be performed in real-time by operating on-line on live network traffic, by potentially including the resulting classifier within a next-generation firewall or IDS system providing limited processing capabilities. The whole operating scenario is depicted in Figure 1 where all the off-line activities, associated to the U-BRAIN framework, are reported into a grey box, whereas the other ones can be managed on-line within the context of a real-time detection system.

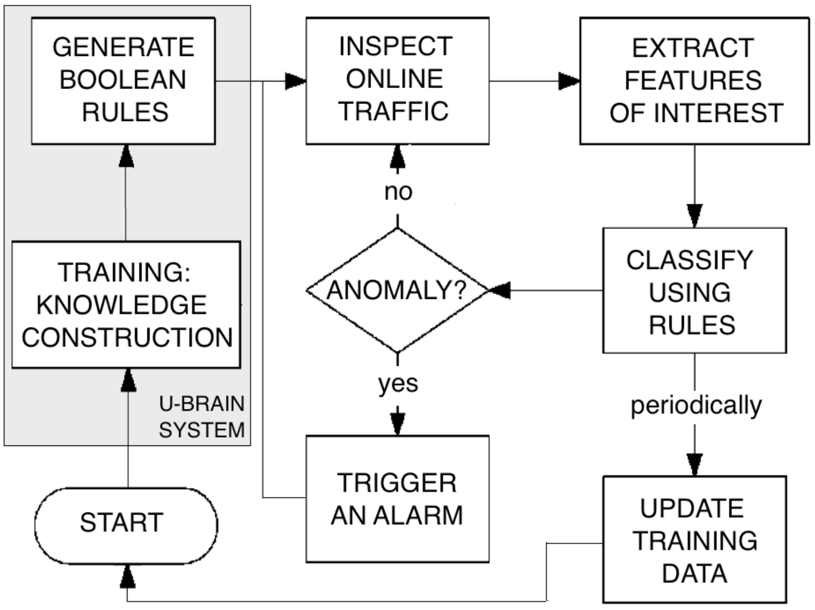

Figure 1: Detection system architecture

However, in order to keep the classifier up-to-date with the evolving network traffic dynamics, periodical re-training is necessary. Due to the above computing and space demands characterizing the training phase, such activity can be performed off-line and independently from the running classifier, by eventually distributing its load among multiple cooperating runtime machines that provide high performance High-Performance Computing (HPC) capabilities. Accordingly, the main goal of a recent work [29] has been the construction of a new version of the U-BRAIN algorithm relying on HPC capabilities, in order to overcome the limits related to its high computational complexity. The new version, using more clever mathematical and programming solutions, such as a Dynamic Programming and a new relevance representation, allows the algorithm implementation on parallel computing systems with reduced communication costs. To this aim an effective distributed computing and storage architecture has been specifically designed according to an high degree temporal and spatial locality principle [30]. The algorithm has been implemented by using a Single Program Multiple Data (SPMD) [31] technique together with a Message-Passing Programming paradigm. Overall, the results obtained on standard data sets show that the parallel version is up to 30 times faster than the serial one. Moreover, by increasing the problem size, with a constant number of processors, the average speed-up further increases. Recently, the U-BRAIN parallel version has been used in diagnosis of aerospace structures' defects. In such a context, the algorithm demonstrated to be effective as a defect classifier in non destructive testing [32].

4. Performance Evaluation

In evaluating the performance of the proposed detection strategy our main aim was making our results comparable with alternative approaches already available in literature. Unfortunately, this is not immediate, in lack of a generally recognized benchmark for assessing and validating anomaly detection solutions. In fact, most of the publicly available data sets and taxonomies that can be used for benchmarking anomaly detection systems are generally known to be error-prone and of limited significance in their results.

However, while strongly criticized [33] as being excessively "artificial" for the usage of synthetic simulated background data not containing the noise that characterize real traffic, the KDD'99 dataset [34], originally produced from DARPA Lincoln Labs, is widely recognized as the most used publicly available labeled dataset for comparing network anomaly detection systems, as it can be appreciated from a huge number of research works. It starts from several weeks of raw traffic data collected within a local-area network (LAN) simulating a typical U.S. Air Force LAN, in which 24 attack types have been identified and labeled, and all the data flow activity has been summarized into connections by using the Bro IDS [6], that extracted 41 distinct features for each connection. Observations are partitioned into two subsets representing respectively the training and the testing set, where the raw training data yielded about 5 million connection records and the test one around 2 million ones. The test set also includes specific attack types not present in the training data (that can be considered as zero-day attacks) in order to asses the classifier generalization capability. In order to avoid some of the known consistency problems of the KDD'99 data set [35] we used its preprocessed version known as NSL-KDD [36] that removed inconsistencies and redundancy in the training set, so that the performance of the classifiers will not be perturbed or biased by the methods which exhibit better detection rates on frequent records. Although the NSL-KDD data set is not a perfect representative of a real network scenario (it still suffers from some of the weaknesses reported in [37]), we believe that it can be considered the best available compromise for comparing/benchmarking different anomaly detection methods.

4.1. Feature Reduction

First of all, we are interested in investigating the relevance of the 41 features available in KDD'99 samples with respect to dataset labels. In order to ensure a satisfactory degree of performance, mainly in terms of responsiveness, the number of features constituting each sample should be kept as small as possible, by diversifying the features and making them discriminative as well as mutually independent. This is due to the fact that a group of highly independent features is more effective in discriminating between the two classes than any of them taken individually.

Several techniques can be used for determining the best, and most relevant features that can be used for building robust learning models. Their goal is significantly improving the detection performance by eliminating irrelevant and redundant features from the available data and hence reducing the overall problem dimension. This transforms our data representation into a shorter, more compact, and hopefully, more predictive one keeping only the really useful features from the original feature set.

The simplest approach for selecting the really necessary features is trying to extract the more specific properties of the traffic under observation, by properly mining the pre-classified feature vectors in the training set. This implies searching the whole feature space, a component at a time, for the subset of features that is most likely to best characterize the normal and anomalous traffic classes, by ranking each individual feature according to the aforementioned considerations and identifying and removing useless features that may adversely affect the detection performance.

4.1.1. Ranking

Ranking may be based on several criteria that are able to quantify the relevance of a given feature, and hence its role in determining the class label. The most used for this purpose is information gain, based on the feature's entropy, and defined as:

$$IG(f_k, c_i) = P(f_k, c_i) log \frac{P(f_k, c_i)}{P(f_k)P(c_i)} + P(\bar{f}_k, c_i) log \frac{P(\bar{f}_k, c_i)}{P(\bar{f}_k)P(c_i)} \tag{4}$$

where the $P(\cdot)$ are occurrence probabilities in S and $\sum_i IG(f_k, c_i) = H(C) - H(C|S)$. When the feature f_k is relevant, and hence necessary for an accurate detection of class c_i, the associated entropies will assume values close to 0 and thus the information gain will be consequently close to 1.

Also the well known χ^2 test can be used as a good quality metric to evaluate the predictive properties of a specific feature, by computing the χ^2 statistic for each combination of feature f_k and class c_i:

$$\chi^2(f_k, c_i) = \frac{N(P(f_k, c_i)P(\bar{f}_k, \bar{c}_i) - P(f_k, \bar{c}_i)P(\bar{f}_k, c_i))^2}{P(f_k)P(\bar{f}_k)P(c_i)P(\bar{c}_i)} \tag{5}$$

where N is the total number of observations in the training set.

We obtained the same (and hence extremely coherent) results from both the Information Gain and χ^2 tests on the whole training set, resulting in the ranking reported in Table 1.

4.1.2. Feature Selection

The top k features (resulting in a percentage of the number of original ones) can be chosen after ordering the whole feature set according to the ranking score. However, features' quality may significantly vary, so that performing feature selection based only on the ranking order can potentially introduce in the set some low-discriminating features. Analogously, the features can be selected by introducing a threshold on the ranking score, (e.g., 0.01 for Information Gain) and including the features with a value over it. However, there is a time consuming but more effective alternative, exploring the predictive power of groups of features by using tree-based classification algorithms such as C4.5 [38]. This may result in an automatic feature selection strategy based on an initial mining phase operating on a sufficiently large quantity of pre-classified data, that starting from a reasonably complete set of features removes one by one the least discriminants

Table 1: Features ranking results

Order	Ranking Score	Feature	Order	Ranking Score	Feature
1	0.81620015	src_bytes	21	0.08826684	dst_host_srv_rerror_rate
2	0.6715649	service	22	0.06263911	protocol_type
3	0.63304893	dst_bytes	23	0.05674069	rerror_rate
4	0.51938822	flag	24	0.05217739	dst_host_rerror_rate
5	0.51868903	diff_srv_rate	25	0.05156684	srv_rerror_rate
6	0.50984907	same_srv_rate	26	0.03672469	duration
7	0.4759185	dst_host_srv_count	27	0.01155446	hot
8	0.43821385	dst_host_same_srv_rate	28	0.00960996	wrong_fragment
9	0.41091066	dst_host_diff_srv_rate	29	0.00650408	num_compromised
10	0.40596111	dst_host_serror_rate	30	0.00371697	num_root
11	0.40475154	logged_in	31	0.00215482	num_access_files
12	0.39806695	dst_host_srv_serror_rate	32	0.00116837	is_guest_login
13	0.39273998	serror_rate	33	0.00091826	num_file_creations
14	0.38358863	count	34	0.00051404	su_attempted
15	0.37912493	srv_serror_rate	35	0.00032406	root_shell
16	0.27083688	dst_host_srv_diff_host_rate	36	0.00010426	num_shells
17	0.19803814	dst_host_count	37	0.00006037	num_failed_logins
18	0.18887561	dst_host_same_src_port_rate	38	0.00003811	land
19	0.14155352	srv_diff_host_rate	39	0.00000717	is_host_login
20	0.09424551	srv_count	40	0	num_outbound_cmds
			41	0	urgent

ones in terms of ranking, by verifying that the classification accuracy does not fall under a predetermined value.

Accordingly, we progressively removed a feature at a time in successive J48-based classifications, until the relative absolute error ε, remains under the 1% threshold. ε has been measured as as the total absolute error relative to what the error would have been if the prediction φ_i had been the average $\overline{\tau}$ of the target values τ_i, so that:

$$\varepsilon = \frac{\sum_{i=1}^{n} |t_i - o_i|}{\sum_{i=1}^{n} \left|t_i - \bar{t}\right|} \tag{6}$$

The whole selection process results in only 6 (starting from the first in table 1) necessary features as it can be seen from the chart in Figure 2, that implies a significant reduction in the overall space and runtime complexity in both the training and testing phases.

Quantization has been performed on all the six attributes (the first two are nominal/discrete, the other numeric/continuous) in order to transform them into binary data strings as required by the U-BRAIN algorithm. Since the *service* and *flag* attributes are respectively composed by 70 and 11 elements, we used 7 bits to represent the former one and 4 bits for the latter. Analogously, by considering their minimum and maximum values, we have chosen 12 bits for *src_bytes* and *dst_bytes*, while 8 bits were devoted to *same_srv_rate* and *diff_srv_rate*. So, 52 bits are necessary to represent each instance, where 51 bits are dedicated to the feature space and one bit is required for the label representing the target class (0=normal, 1=anomalous). Each connection has been labeled as either associated to normal activity or to a network anomaly, regardless of the specific attack type reported in the KDD dataset. The individual instance layout is presented in table 2.

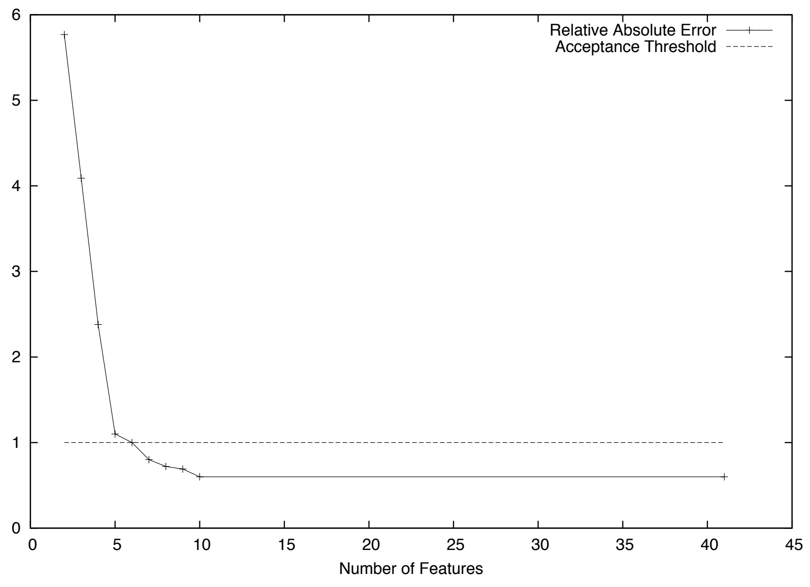

Figure 2: Feature selection results

Table 2: Generic instance layout

Description	**Attribute**	**Bits**
destination network service	service	7
connection attributes and error status	flags	4
bytes from source to destination	src_bytes	12
bytes from destination to source	dst_bytes	12
% of connections to the same service	same_srv_rate	8
% of connections to different services	diff_srv_rate	8
label	class	1

4.2. Training and Testing

In order to assess the potential of the proposed strategy for detecting network anomalies we provided a proof of concept using U-BRAIN to find rules and formulae for distinguishing between anomalous (intrusions, attacks) and normal connections/events. U-BRAIN has been properly trained on the NSL-KDD training set and its real effectiveness has been successively verified on the corresponding testing set. However, due to the U-BRAIN consistency bias, a data pre-processing/cleaning step has been necessary to remove the inconsistencies on both training and testing data. In particular, all the new duplicates introduced by feature reduction (the instances passed from 41 to 6 attributes) have been removed. The resulting dataset is composed by 21178 training instances (8757 positive and 12421 negative) and 6032 testing instances (2116 positive and 3916 negative), as reported in Table 3.

Table 3: The NSL-KDD data set after preprocessing

Training Set			**Testing Set**		
Positives	**Negatives**	**Total**	**Positives**	**Negatives**	**Total**
8757	12421	21178	2116	3916	6032

First of all, 10-fold cross validation has been performed on the entire training set, in order to ensure results consistency and guarantee a better reliability to the whole process. Accordingly, all the training data have been randomly divided into 10 disjoint subsets (folders), each containing approximately the same amount of instances (see Table 4).

Table 4: 10-fold cross-validation instance breakdown

Test	**Training Data**			**Test data**		
	Positives	**Negatives**	**Total**	**Positives**	**Negatives**	**Total**
1	7843	11218	19061	914	1203	2117
2	7870	11191	19061	887	1230	2117
3	7831	11230	19061	926	1191	2117
4	7865	11196	19061	892	1225	2117
5	7883	11178	19061	874	1243	2117
6	7840	11221	19061	917	1200	2117
7	7852	11209	19061	905	1212	2117
8	7886	11175	19061	871	1246	2117
9	7972	11089	19061	785	1332	2117
10	7971	11082	19053	786	1339	2125

In each experiment, nine folders have been used as training data, while the remaining folder is used for validation. This process has been repeated 10 times, for each different choice of the validation folder. The results are reported in Table 5, resuming how well the model assigns the correct class value to the test instances.

In detail, the first four columns, after the test number, represent the confusion matrix in terms of True Positives (TP: correct detections), False Negatives (FN: missed detections), True Neg-

Table 5: 10-fold cross-validation classification performance on NSL-KDD training data

Test	Confusion matrix				Accuracy	Sensitivity	Specificity	Precision	F1	Correlation	ROC	Error
	TP	FP	TN	FN					Measure	Coefficient	Area	Ratio
1	905	9	1194	9	0.991	0.990	0.993	0.990	0.990	0.983	0.991	0.009
2	880	14	1216	7	0.990	0.992	0.989	0.984	0.988	0.980	0.990	0.010
3	913	14	1177	13	0.987	0.986	0.988	0.985	0.985	0.974	0.987	0.013
4	880	19	1206	12	0.985	0.987	0.984	0.979	0.983	0.970	0.986	0.015
5	866	18	1225	8	0.988	0.991	0.986	0.980	0.985	0.975	0.988	0.012
6	905	11	1189	12	0.989	0.987	0.991	0.988	0.987	0.978	0.989	0.011
7	894	23	1189	11	0.984	0.988	0.981	0.975	0.981	0.967	0.984	0.016
8	856	21	1225	15	0.983	0.983	0.983	0.976	0.979	0.965	0.983	0.017
9	775	23	1309	10	0.984	0.987	0.983	0.971	0.979	0.967	0.985	0.016
10	778	19	1320	8	0.987	0.990	0.986	0.976	0.983	0.973	0.988	0.013
Mean	865	17	1225	11	0.987	0.988	0.986	0.980	0.984	0.973	0.987	0.013
StDev					0.003	0.003	0.004	0.0062	0.004	0.006	0.003	0.003
Cin @ 95%					(0.985-0.989)	(0.986-0.990)	(0.984-0.989)	(0.976-0.985)	(0.981-0.987)	(0.969-0.978)	(0.985-0.989)	(0.011-0.015)

atives (TN: correct silence), False Positives (FN: false alarms). The following seven columns report respectively:

- *Accuracy* $\left(\frac{(TP+TN)}{(TP+TN+FP+FN)}\right)$, that is the portion of correctly classified instances.
- *Sensitivity* $\left(\frac{TP}{(TP+FN)}\right)$, also called True Positive Rate, that is the portion of positive instances which are correctly identified as positives by the classifier.
- *Specificity* $\left(\frac{TN}{(TN+FP)}\right)$, also called True Negative Rate, that refers to the classifier's ability to identify negative results.
- *Precision* $\left(\frac{TP}{(TP+FP)}\right)$, that is a measure of retrieved instances that are relevant.
- *F1-Measure* $\left(\frac{2 \cdot TP}{(2 \cdot TP+FP+FN)}\right)$, that is the harmonic mean of precision and sensitivity, hence gathering into a single value both the metrics. In fact, evaluating precision and sensitivity also in a joint way may be very useful, since it is quite easy optimizing one of the metrics by declining the other.
- *Matthews Correlation Coefficient* [39] $\left(\frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}\right)$, correlating the observed and predicted binary classifications by simultaneously considering true and false positives and negatives. It can assume a value between -1 and +1, where +1 represents a perfect prediction, 0 no better than random prediction and -1 indicates total disagreement between prediction and observation.
- *ROC Area* or Area under the ROC curve (plotting the detection rate against the false positive one), summarizes the total accuracy of the detector in a way that accounts for both the gains in True Positive Rate and the losses in False Positive Rate. More precisely, it represents the probability for the classifier of ranking a randomly selected positive instance better than a randomly selected negative one, by reflecting the inherent difficulty of the detection task in presence of noise. Accordingly, with a ROC Area value of 0.9, for example, there is an 90% probability that given two randomly selected alternatives, the correct one will be identified, so that a ROC Area value of 1 is associated to perfect detection while an value of 0.5 implies completely random results.

Finally, the last column reports the error ratio referred to the per-folder classification process.

The mean value, the standard deviation and the confidence interval (at a 95% of confidence level) are reported for all performance parameters in order to emphasize the reliability and significance of the results.

Once the classification formulae have been built from the entire training set, the overall detection performance has been evaluated on the testing set. The results are reported in Table 6, where we can observe a significant classification accuracy associated to a very high precision in identifying traffic that is not affected by anomalies. This is also confirmed by the observation of the F1-Measure results, strongly reflecting the excellent combined performance in precision and sensitivity. In addition, both the Matthews Correlation Coefficient and the area under the ROC curve report respectively a very satisfactory score (0.87) and a value (0.93) quite near to a perfect prediction. In addition, the whole experiment resulted in a very limited error ratio. We can argue that the errors observable from the confusion matrix can be in most part due to specific kind of non-noisy attacks present in the testing set (e.g., buffer overflow), that due to their inherent structure in term if traffic pattern may be indistinguishable from normal activity and hence lead to contradictory results. However, this should not be considered a real problem in a network anomaly detection system, since the results show a good efficiency in identifying traffic patterns that can be considered "normal", and we are essentially interested in distinguishing the occurrence of suspicious events (and eventually flagging them for further analysis) deviating from the "normal" traffic behavior.

Table 6: Classification performance on the NSL-KDD testing set

Confusion matrix				**Accuracy**	**Sensitivity**	**Specificity**	**Precision**	**F1**	**Correlation**	**ROC**	**Error**
TP	**FP**	**TN**	**FN**					**Measure**	**Coefficient**	**Area**	**Ratio**
1890	128	3788	226	0.941	0.893	0.967	0.936	0.914	0.870	0.930	0.059

In addition, in order to evaluate our anomaly detection strategy also on real traffic data, we tested it against a packet trace collected at the Federico II University of Napoli, containing all the (anonymized) traffic incoming from an 1 Gbps link to the Internet of the engineering Campus, collected on March 24, 2009 from 00:00 to 13:00. Only incoming traffic has been collected for traffic cleaning and purity reasons (filtering undesired anomalous events is much easier since the expected traffic pattern is known). The trace contains several anomalous events simulated through distributed SYN floods and port-scanning attacks occurring at various times (see table 7), properly chosen as representative of most of the anomalous traffic patterns that can be observed on a border connection to the Internet (inbound distributed denial of service attacks, bandwidth floods, single and multiple scans). The features of interest, involving number of packets, as well as their rates and average sizes (resulting from feature selection, as previously described), have been extracted by using the CAIDA CoralReef toolset.

The classification results reported in table 8, related to the 10-fold cross validation, demonstrate the effectiveness of the proposed approach also on real data.

Start Time	Duration	Attack	Packet rate
01:15	60s	SYN flood	500/s
02:15	300s	SYN flood	500/s
03:15	600s	SYN flood	500/s
04:15	60s	SYN flood	250/s
05:15	300s	SYN flood	250/s
06:15	600s	SYN flood & portscan	250/s

Table 7: The simulated attacks in the real traffic trace.

Table 8: 10-fold cross-validation classification performance on real traffic data

Test	Confusion matrix				Accuracy	Sensitivity	Specificity	Precision	F1	Correlation	ROC	Error
	TP	FP	TN	FN					Measure	Coefficient	Area	Ratio
1	77	13	2007	11	0.989	0.875	0.994	0.856	0.865	0.859	0.934	0.011
2	81	32	1988	7	0.981	0.920	0.984	0.717	0.806	0.803	0.952	0.019
3	80	24	1996	8	0.985	0.909	0.988	0.769	0.833	0.829	0.949	0.015
4	78	212	1808	10	0.895	0.886	0.895	0.269	0.413	0.454	0.891	0.105
5	76	12	2008	12	0.989	0.864	0.994	0.864	0.864	0.858	0.929	0.011
6	67	8	2013	20	0.987	0.770	0.996	0.893	0.827	0.823	0.883	0.013
7	49	26	1994	39	0.969	0.557	0.987	0.653	0.601	0.587	0.772	0.031
8	65	8	2012	23	0.985	0.739	0.996	0.890	0.807	0.804	0.867	0.015
9	59	17	2003	29	0.978	0.670	0.992	0.776	0.720	0.710	0.831	0.022
10	44	15	2032	17	0.985	0.721	0.993	0.746	0.733	0.726	0.857	0.015
Mean	67.6	36.7	1986.1	17.6	0.974	0.791	0.982	0.743	0.747	0.745	0.887	0.026
StDev					0.029	0.120	0.031	0.185	0.142	0.132	0.058	0.029
Cin @ 95%					(0.954-0.995)	(0.705-0.877)	(0.960-1.004)	(0.611-0.875)	(0.645-0.849)	(0.651-0.840)	(0.845-0.928)	(0.005-0.046)

4.3. Results Comparison

4.3.1. Classification Stage

In order to obtain a clearer view about the quality of the above results on terms of classification performance we compared them with those obtained on the same data set by using several of the most common and effective supervised classification techniques whose implementation details are widely known. These techniques have been explicitly chosen to cover the fundamental categories of approaches available in literature, ranging from *rule-based* methods (in our case C4.5/J48 [40][41]), to *neural networks* (with a Multilayer Perceptron (MLP) [42]), *Support Vector Machines* (SVM) [43][44] or *Bayesian networks* (by using a Naive-Bayes classifier [45]).

As comparison metrics, in addition to the more traditional accuracy and precision, we used the Matthews correlation coefficient that is considered to be the most effective quality measure in binary classification, and in particular in anomaly detection where the amount of normal instances greatly surpasses the number of anomalous ones. In fact, such coefficient is significantly more stable, in presence of very different class sizes, than other widely used metrics such as the area under the ROC curve. It also achieves an optimum balance of the types of errors over the two classes. As it can be easily appreciated from Figure 3 as well as from the detailed performance results reported in Table 9, the U-BRAIN-based strategy outperforms all the other methods on the NSL-KDD data. Network anomalies are rare with respect to ordinary situations, and it is preferable to ensure high specificity in the anomaly detection, even at the cost of sacrificing the sensitivity of the test, so having more false negatives but very few false positives. This is exactly the kind of behavior shown by the U-BRAIN rules/formulas whose main aim is effective generalization through inductive inference, also supported by the introduction of a certain degree of tolerance/uncertainty in the detection in order to be more adaptive in presence of the continuously evolving network phenomena.

Table 9: Classification performance comparison results

	Accuracy	Sensitivity	Specificity	Precision	Correlation
U-Brain	0.941	0.893	0.967	0.9362	0.870
J48	0.858	0.962	0.779	0.767	0.736
SVM	0.795	0.966	0.664	0.685	0.639
MLP	0.768	0.905	0.664	0.671	0.571
Naive Bayes	0.698	0.977	0.487	0.591	0.508

4.3.2. Application Stage

In order to compare the efficiency and effectiveness of the proposed technique with other known anomaly detection approaches, not only in terms of classification performance, but also at the final application stage, we performed a tentative comparison on the final accuracy of the results achieved on the same dataset by our scheme and several distinguished (supervised and unsupervised) anomaly detection methods such as the k-Nearest Neighbor outlier mining algorithm (K-NN) [46], fixed-width clustering (Cluster) [47], ADWICE [18], fpMAFIA [48], SVM+DGSOT [20], SVM [48], K-Means+ID3 [19], HDG-clustering [49], RPCPAI [50] and Recurrence Quantification-based [51], whose ROC graphs and Area measures are available in literature. The area under the ROC curve has been chosen for comparison since it is one of the most standardized accuracy metrics for anomaly detection algorithms, that consequently has been widely used in literature, so that many performance results referring to the KDD dataset are publicly available.

Based on the comparison results reported in Figure 4, we can assert that the proposed detection method can compete with the other available techniques since its detection performance is located exactly on the average (0.93) of all the examined ROC results, represented by the dotted line in the aforementioned Figure. We can also appreciate a very low standard deviation (0.06) and a $[0.8831 - 0.9697]$ 95% confidence interval, both denoting a good reliability and statistical significance of the above results.

5. Conclusions

Identifying anomalous events is one of the best ways to discover a lot of existing malfunctions and handle most of the security and performance problems that may occur in modern networks. Hence, the availability of reliable detection devices and strategies becomes a fundamental prerequisite for next generation network-empowered infrastructures. We presented a new supervised machine learning approach to anomaly detection, whose goal is understanding the dynamics and behaviors characterizing network traffic in order to generate a set of rules and criteria that can be used to effectively discriminate anomalous events in the normal traffic flow. Such approach couples the capability of inferring rigid decisional structures, represented as boolean formulas, from incomplete sample observations, with the flexibility introduced by a fuzzy-based uncertainty management strategy. This allows the detection engine to easily adapt to the very different

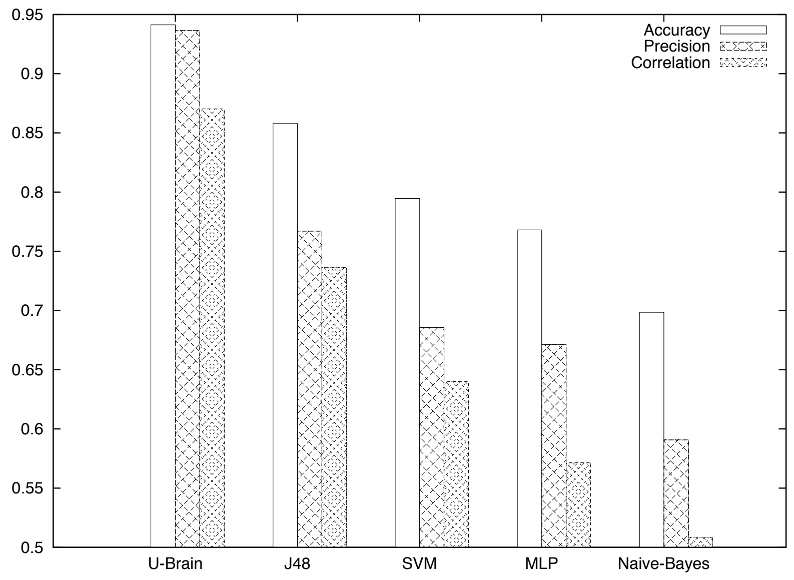

Figure 3: Classification performance comparison histogram

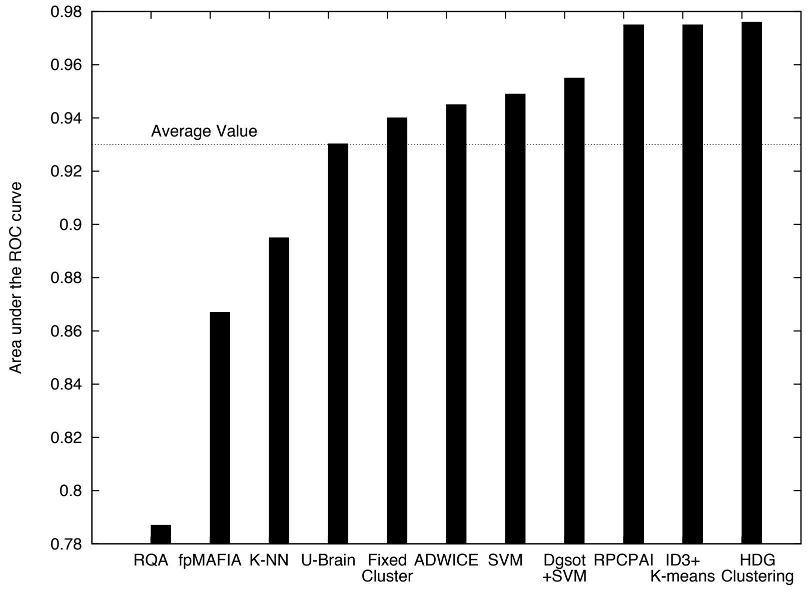

Figure 4: Application performance comparison histogram

kind of phenomena that can be experienced on a real network. The results of the experimental evaluation demonstrate the ability of successfully handling very different kind of events/phenomena within the context of a quite difficult selection of training and testing data, commonly used for assessing detection approaches.

However, it must be considered that the detection capability is directly depending on accuracy of the self-learnt rules describing the traffic model, so that, if the training data are not enough complete and realistic, i.e., they do not reflect all the aspects characterizing the real network traffic, the risk of false positives and/or negatives increases. This introduces the need of an incremental knowledge construction ad refinement process, implemented within the context of a continuous supervised re-training mechanism, managed trough human-driven results validation. Furthermore, while rule construction is a computationally heavy task that has to be managed off-line, on a periodic basis, by relying on properly crafted parallel computing environments, the on-line detection activity, based on the above rules, is extremely simple and effective in term of performance and can be easily implemented in most of the next-generation security equipments available on modern networks.

Finally, it should be considered that, by relying on a native rule-based detection strategy, where the inferred rules have the main goal of reliably describing the model (ideally dynamically kept up-to-date through periodic re-training) that represents network traffic, this approach is potentially more effective against previously unknown phenomena and robust against obfuscation mechanisms.

References

[1] J. P. Anderson, Computer security threat monitoring and surveillance, Tech. rep., Computer Security Division of the Information Technology Laboratory, National Institute of Standards and Technology (1980).

[2] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM Comput. Surv. 41 (3) (2009) 15:1–15:58.

[3] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Macia-Fernandez, E. Vazquez, Anomaly-based network intrusion detection: Techniques, systems and challenges, Computers & Security 28 (1-2) (2009) 18–28.

[4] A. Patcha, J.-M. Park, An overview of anomaly detection techniques: Existing solutions and latest technological trends, Computer Networks 51 (12) (2007) 3448–3470.

[5] M. Roesch, et al., Snort: Lightweight intrusion detection for networks, in: Proceedings of the 13th USENIX Conference on System Administration, Vol. 99, 1999, pp. 229–238.

[6] V. Paxson, Bro: a system for detecting network intruders in real-time, Computer networks 31 (23) (1999) 2435–2463.

[7] D. Anderson, T. F. Lunt, H. Javitz, A. Tamaru, A. Valdes, et al., Detecting unusual program behavior using the statistical component of the Next-generation Intrusion Detection Expert System (NIDES), SRI International, Computer Science Laboratory, USA SRI-CSL-95-06, 1995.

[8] S. Staniford, J. A. Hoagland, J. M. McAlerney, Practical automated detection of stealthy portscans, Journal of Computer Security 10 (1) (2002) 105–136.

[9] T. Lane, C. E. Brodley, An application of machine learning to anomaly detection, in: Proceedings of the 20th National Information Systems Security Conference, Vol. 377, Baltimore, USA, 1997, pp. 366–380.

[10] A. K. Ghosh, A. Schwartzbard, A study in using neural networks for anomaly and misuse detection., in: USENIX Security, 1999.

[11] V. N. Dao, V. Vemuri, A performance comparison of different back propagation neural networks methods in computer network intrusion detection, Differential Equations and Dynamical Systems 10 (1&2) (2002) 201–214.

[12] S. Mukkamala, G. Janoski, A. Sung, Intrusion detection using neural networks and support vector machines, in: Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on, Vol. 2, IEEE, 2002, pp. 1702–1707.

[13] W. Lee, S. J. Stolfo, K. W. Mok, Mining in a data-flow environment: Experience in network intrusion detection, in: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 1999, pp. 114–124.

[14] W. Lee, S. J. Stolfo, K. W. Mok, A data mining framework for building intrusion detection models, in: Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on, IEEE, 1999, pp. 120–132.

[15] C. Warrender, S. Forrest, B. Pearlmutter, Detecting intrusions using system calls: Alternative data models, in: Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on, IEEE, 1999, pp. 133–145.

[16] P. K. Chan, M. V. Mahoney, M. H. Arshad, Learning rules and clusters for anomaly detection in network traffic, in: Managing Cyber Threats, Springer, 2005, pp. 81–99.

[17] N. Duffield, P. Haffner, B. Krishnamurthy, H. Ringberg, Rule-based anomaly detection on ip flows, in: INFOCOM 2009, IEEE, IEEE, 2009, pp. 424–432.

[18] K. Burbeck, S. Nadjm-Tehrani, Adwice–anomaly detection with real-time incremental clustering, in: Information Security and Cryptology–ICISC 2004, Springer, 2005, pp. 407–424.

[19] S. R. Gaddam, V. V. Phoha, K. S. Balagani, K-means+ id3: A novel method for supervised anomaly detection by cascading k-means clustering and id3 decision tree learning methods, Knowledge and Data Engineering, IEEE Transactions on 19 (3) (2007) 345–354.

[20] L. Khan, M. Awad, B. Thuraisingham, A new intrusion detection system using support vector machines and hierarchical clustering, The VLDB Journal?The International Journal on Very Large Data Bases 16 (4) (2007) 507–521.

[21] D. peng Chen, X. song Zhang, Internet anomaly detection with weighted fuzzy matching over frequent episode rules, in: Apperceiving Computing and Intelligence Analysis, 2008. ICACIA 2008. International Conference on, 2008, pp. 299–302. doi:10.1109/ICACIA.2008.4770028.

[22] J. E. Dickerson, J. A. Dickerson, Fuzzy network profiling for intrusion detection, in: Fuzzy Information Processing Society, 2000. NAFIPS. 19th International Conference of the North American, IEEE, 2000, pp. 301–306.

[23] E. Mendelson, Introduction to mathematical logic, CRC press, 1997.

[24] S. Rampone, C. Russo, A fuzzified BRAIN algorithm for learning DNF from incomplete data, Electronic Journal of Applied Statistical Analysis (EJASA) 5 (2) (2012) 256–270.

[25] S. Rampone, Recognition of splice junctions on DNA sequences by BRAIN learning algorithm., Bioinformatics 14 (8) (1998) 676–684.

[26] R. S. Michalski, A theory and methodology of inductive learning, Springer, 1983.

[27] T. M. Mitchell, Generalization as search, Artificial intelligence 18 (2) (1982) 203–226.

[28] D. Haussler, Quantifying inductive bias: Ai learning algorithms and valiant's learning framework, Artificial intelligence 36 (2) (1988) 177–221.

[29] G. D'Angelo, S. Rampone, Towards a hpc-oriented parallel implementation of a learning algorithm for bioinformatics applications, BMC Bioinformatics 15 (5) (2014) 1–15.

[30] J. S. Vitter, External memory algorithms and data structures: Dealing with massive data, ACM Computing surveys (CsUR) 33 (2) (2001) 209–271.

[31] F. Darema, The spmd model: Past, present and future, in: Recent Advances in Parallel Virtual Machine and Message Passing Interface, Springer, 2001, pp. 1–1.

[32] G. D'Angelo, S. Rampone, Diagnosis of aerospace structure defects by a hpc implemented soft computing algorithm, in: Proceedings of the IEEE International Workshop on Metrology for Aerospace, Benevento, Italy, 2014, pp. 408–412.

[33] S. T. Brugger, J. Chow, An assessment of the DARPA IDS Evaluation Dataset using Snort, Tech. rep., University of California, Davis, Department of Computer Science (2007).

[34] DARPA, Kdd cup 1999 data set, available at the following website http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[35] M. Tavallaee, E. Bagheri, W. Lu, A.-A. Ghorbani, A detailed analysis of the KDD CUP 99 data set, in: Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications 2009, 2009.

[36] M. Tavallaee, Nsl-kdd dataset, http://www.iscx.ca/NSL-KDD.

[37] J. McHugh, Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory, ACM transactions on Information and system Security 3 (4) (2000) 262–294.

[38] R. O. Duda, P. E. Hart, D. G. Stork, Pattern classification, John Wiley & Sons,, 1999.

[39] D. M. Powers, Evaluation: from precision, recall and f-measure to roc, informedness, markedness & correlation, Journal of Machine Learning Technologies 2 (1) (2011) 37–63.

[40] G. H. John, Robust linear discriminant trees, in: Learning from Data, Springer, 1996, pp. 375–385.

[41] R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[42] M. Augusteijn, B. Folkert, Neural network classification and novelty detection, International Journal of Remote Sensing 23 (14) (2002) 2891–2902.

[43] I. Steinwart, D. R. Hush, C. Scovel, A classification framework for anomaly detection, in: Journal of Machine Learning Research, 2005, pp. 211–232.

[44] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, ACM Transactions on Intelligent Systems and Technology (TIST) 2 (3) (2011) 27.

[45] G. H. John, P. Langley, Estimating continuous distributions in bayesian classifiers, in: Eleventh Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Mateo, 1995, pp. 338–345.

[46] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, S. Stolfo, A geometric framework for unsupervised anomaly detection, in: Applications of data mining in computer security, Springer, 2002, pp. 77–101.

[47] J. Oldmeadow, S. Ravinutala, C. Leckie, Adaptive clustering for network intrusion detection, in: Advances in Knowledge Discovery and Data Mining, Springer, 2004, pp. 255–259.

[48] K. Leung, C. Leckie, Unsupervised anomaly detection in network intrusion detection using clusters, in: Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38, Australian Computer Society, Inc., 2005, pp. 333–342.

[49] C.-F. Tsai, C.-C. Yen, Unsupervised anomaly detection using hdg-clustering algorithm, in: Neural Information Processing, Springer, 2008, pp. 356–365.

[50] M. Mohammadi, A. Akbari, B. Raahemi, B. Nassersharif, H. Asgharian, A fast anomaly detection system using probabilistic artificial immune algorithm capable of learning new attacks, Evolutionary Intelligence 6 (3) (2014) 135–156.

[51] F. Palmieri, U. Fiore, Network anomaly detection through nonlinear analysis, Computers and Security 29 (7) (2010) 737–755.