Intrusion Detection Using Ensembles

Alexandre Balon-Perin[*†]
abalonpe@ulb.ac.be

Björn Gambäck[*‡]
gamback@idi.ntnu.no

Lillian Røstad[*]
lillianro@idi.ntnu.no

[†]Ecole Polytechnique
Université libre de Bruxelles
Brussels, Belgium

[*]Department of Computer and Information Science
Norwegian University of Science and Technology
Trondheim, Norway

[‡]SICS — Swedish Institute
of Computer Science AB
Kista, Sweden

***Abstract*—The paper discusses intrusion detection systems built using ensemble approaches, i.e., by combining several machine learning algorithms. The main idea is to exploit the strengths of each algorithm of the ensemble to obtain a robust classifier. Network attacks can be divided into four classes: probe, remote to local, denial of service, and user to root. Each module of the ensemble designed in this work is itself an ensemble using bagging of decision trees and is specialized in the detection of one class of attacks. Experiments highlighted the efficiency of the approach and showed that increased accuracy can be obtained when each class of attacks is treated as a separate problem and handled by specialized algorithms. In all experiments, the ensemble was able to decrease the number of false positives and false negatives.**

***Index Terms*—intrusion detection; ensemble approaches; bagging**

I. Introduction

Intrusion detection systems (IDSs) are monitoring devices that have been added to the wall of security in order to prevent malicious activity on a system. Here we will focus on network intrusion detection systems mainly because they can detect the widest range of attacks compared to other types of IDSs. Network IDSs analyse traffic to detect on-going and incoming attacks on a network. Additionally, they must provide concise but sound reports of attacks in order to facilitate the prevention of future intrusions and to inform the network administrators that the system has been compromised. Current commercial IDSs mainly use a database of rules (*signatures*), to try to detect attacks on a network or on a host computer. This detection method is presently the most accurate, but also the easiest to evade for experienced malicious users, because variants of known attacks (with slightly different signatures) are considered harmless by the IDS and can pass through without warning. New attacks and attacks exploiting zero-day vulnerabilities can also slip through the security net if their signatures are unknown to the IDS. (A *zero-day vulnerability* is a software weakness unknown by the system developers which potentially could allow an attacker to compromise the system. 'Zero-day' refers to the first day, day zero, that the vulnerability is observed.)

Hence there is a need for mechanisms allowing the IDS to learn by itself how to detect previously unseen attacks or variants of known attacks. However, the problem is further complicated by the extreme requirements of robustness of the IDS. It must be able to detect all previously seen and unseen attacks without failure, it must never let an attack pass through unnoticed, and it must never deliver unwanted warnings when the traffic is in fact legitimate. For a summary of the main challenges that machine learning has to overcome to be useful for intrusion detection, see [1].

Despite this, several attempts have been made to build automatically adaptable intrusion detection systems using various machine learning algorithms. So far though, the machine learning classifiers trigger too many false alarms to be useful in practice. Part of the problem is the lack of labelled datasets to train the classifiers on. The only labelled dataset available is the KDD99 dataset (www.sigkdd.org/kddcup) which is an adaptation of the DARPA98 dataset created in 1998 by the Defense Advanced Research Projects Agency (DARPA). To address these problems, new machine learning paradigms have been introduced in the field of intrusion detection, and in general the machine learning community has in recent years paid more attention to ensemble approaches, i.e., combinations of several machine learning algorithms.

Network attacks can be divided into four classes: probe, remote to local, denial of service, and user to root. Most previous machine learning-based solutions include a single algorithm in charge of detecting all classes of attacks. Instead, in this work, one module of an ensemble is specialised on the detection of attacks belonging to one particular class. The main idea is to exploit the strengths of each algorithm of the ensemble to obtain a robust classifier. Ensembles are particularly efficient in cases like this, when a problem can be segmented into parts, so that each module of the ensemble is assigned to one particular subproblem. The modules in turn include one or more algorithms cooperating together.

Furthermore, each class of attacks is characterized by very specific properties, observable through the values of certain features on instances in the dataset belonging to a specific class of attacks. However, even though feature selection is often applied in IDSs using machine learning techniques, often only one set of features is selected for *all* classes of attacks. In this work, one set of features is selected for *each* class of attacks according to their relevance to the corresponding class. The corresponding algorithm(s) is then fed with the appropriate set of features. The system can, in theory, reach a very high accuracy with a small cost, and the ensemble processing can potentially be parallelized.

The rest of the paper is laid out as follows: First Section II gives an overview of the state-of-the-art by introducing the resources and methods used in the experiments and describing related work, in particular focusing on previous efforts in applying ensemble-based methods to intrusion detection. The

core of the paper is Section III which details two rounds of experiments carried out, on feature selection for ensembles resp. on feeding an ensemble of machine learning algorithms with the most successful sets of features identified. Section IV then discusses the results of the experiments at length and points to ways in which the present work could be extended. Finally, Section V sums up the previous discussion.

II. Ensemble-based Intrusion Detection

The ensemble method is a way to build different types of approaches to solving the same problem: the outputs of several algorithms used as predictors for a particular problem are combined to improve the accuracy of the overall system. The difficulty of ensemble approaches lays in the choice of the algorithms constituting the ensemble and the decision function which combines the results of the different algorithms. Often, the more algorithms the better, but it is important to take into account the computational expense added by each new algorithm. The decision function is often a majority vote which is both simple and efficient, but alternatives should be analysed to obtain an optimal combination. Another advantage of ensemble approaches is their modular structure, unlike hybrid constructions which are engineered with algorithms having non-interchangeable positions. Consequently, the ensemble designer can easily replace one or more algorithms with a more accurate one.

Bagging and boosting are the two main techniques used to combine the algorithms in an ensemble. In an ensemble using the boosting technique, the algorithms are used sequentially. The advantage of this technique is that the most difficult examples can be classified correctly without adding too much computational overload. In an ensemble using the bagging technique all algorithm of the ensemble are used in parallel. In this case, each algorithm builds a different model of the data and the outputs of all predictors are combined to obtain the final output of the ensemble. In order to build different models, either each algorithm of the ensemble, or the data fed to each algorithm, or both, can be different. Since all algorithms perform in parallel, each of them can be executed on a different processor to speed up the computation.

A. The KDD99 Dataset

As observed in the introduction, part of the problem of automatically creating good intrusion detection systems is the lack of labelled datasets to train on. The only one available is the KDD cup 99 dataset, which was used for the first time in the 3rd International Knowledge Discovery and Data Mining Tools Competition in 1999. It is based on the DARPA98 dataset (built during the DARPA98 IDS evaluation program) which includes seven weeks of data from traffic passing through a network engineered for the purpose, i.e., the traffic was generated in a simulated and controlled environment.

Table I shows the distribution of instances of the KDD cup 1999 training and test sets over the different classes. All the examples are separated into the class `Normal` and four different classes of attacks: `Probe`, `R2L` (remote to local), `DoS` (denial of service), and `U2R` (user to root). Each entry in the sets is represented by 41 features such as `duration`, `src_bytes`, `dst_bytes`, etc., and a label. The training set contains 4,898,431 entries and is highly unbalanced. Whereas the `DoS` class contains 3,883,370 instances, the classes `U2R` and `R2L` are represented by only 52 and 1,126 instances, respectively. With such a small number of examples to train on, it can be expected that it will be difficult for the classifiers to predict the correct classes of unseen examples.

TABLE I
Types of attacks in the KDD cup 99 data sets

Set	`Normal`	`Probe`	`R2L`	`DoS`	`U2R`
Training	972,781	41,102	1,126	3,883,370	52
Test	60,593	4,166	16,347	229,853	70

The test set is composed of 311,029 entries with a distribution of the examples over the different classes similar to that in the training set. However, the number of examples belonging to the class `R2L` is more than ten times higher, so that in order to perform well on the test set, the predictor must acquire a very high power of generalisation. Most importantly, the number of unseen attacks added in the test set is huge: for the classes `U2R`, `R2L` and `Probe`, it is respectively 44.29%, 63.34% and 42.94%. Furthermore, the attacks "spy" and "warezclient" belonging to the class `R2L` are not represented in the test set. In particular, "warezclient" attacks count for more than 90% of the `R2L` training set. Finally, two entries in the test set erroneously have a `service` value of `ICMP`, as also previously reported [2]. Those were removed from the test set before the experiments reported in Section III.

The major criticisms of the KDD99 dataset include the unbalanced distribution of the data, the redundant records which can introduce a bias in the learning phase because of their frequency, that the dataset includes old attacks which have been mostly mitigated, and that the data were captured from a controlled environment somewhat different from what is observed in the wild. The first two issues can be addressed by sampling appropriate sets of examples in each class. However, the distribution of `R2L` attacks in the training set and the test set is a problem which is difficult to overcome. Nevertheless, the KDD99 dataset is far from useless. Firstly, if an IDS using machine learning does not perform well on old attack provided that the data are well sampled, why would it on newer ones? Furthermore, most of the research in the field of machine learning applied to intrusion detection uses the KDD99 dataset, making it a vector of comparison between different approaches. The controlled nature of the environment in which the data were captured is probably the most problematic. For example, the high number of attacks in comparison to normal traffic does not reflect the reality of a network in which almost all traffic is normal.

B. Related Work

Intrusion detection systems have been around since the 80s. In the late 90s researchers in artificial intelligence started to incorporate their algorithms to improve IDSs. An intrusion detection system should be able to autonomously recognize

malicious actions in order to defend itself against variants of previously seen attacks and against attacks exploiting zero-day vulnerabilities. Misuse-based IDSs can only detect attacks whose signatures are available in their signature database. Signatures of attacks are very specific, and a slight variation of the attack can make it unnoticeable for the IDS. That is why learning mechanisms must be implemented to detect and prevent these attacks without having to wait for an update of the signature database or a patch for the vulnerable system. Still, machine learning algorithms are designed to recognize examples similar to those available in the training set used to build the model of the data. Consequently, an IDS using machine learning would have a hard time detecting attacks which patterns are totally different from the data previously seen. In other words, even though machine learning is a suitable candidate to detect variants of known attacks, detecting completely new types of attacks might be out of reach for these kinds of algorithms.

For a summary of most research involving machine learning applied to IDSs until 2007, see [3] which covers a range of techniques, including fuzzy sets, soft computing, and bio-inspired methods such as artificial neural networks, evolutionary computing, artificial immune systems, and swarm intelligence; comparing the performance of the algorithms on the KDD99 test set and showing that all algorithms perform poorly on the `U2R` and `R2L` classes. The best results reported are by genetic programming with transformation functions for `R2L` and `Probe` and by linear genetic programming (LGP) for `DoS` and `U2R` (with 80.22%, 97.29%, 99.7% and 76.3% accuracy, respectively). However, since ensemble-based methods is a fairly new technique applied to intrusion detection, their description in the review is somewhat limited. The works on the topic date from 2003 and many papers were written in 2004 and 2005. Recently, there has been a renewed interest of ensembles in this field [4]–[6].

Abrahams *et al.* have performed three types ensemble-based experiments, all on a subset of the DARPA1998 dataset composed of 11,982 randomly selected examples: First, in [7], an ensemble composed of different types of artificial neural networks (ANN), support vector machines (SVM) with radial basis function kernel, and multivariate adaptive regression splines (MARS) combined using bagging techniques was compared to the results obtained by each algorithm executed separately. SVM used alone outperformed the other single algorithms, but was totally outperformed by the ensemble.

Second, in [8], the combination of classification and regression trees (CART) and bayesian networks (BN) in an ensemble using bagging techniques was explored. Feature selection was applied to speed up the processing: the performance on the set of 41 features was compared to a set of 12 selected by BN, 17 selected by CART and 19 features selected by another study. The final ensemble was composed of three CART to detect `Normal`, `Probe` and `U2R` examples, respectively; one ensemble of one CART and one BN to detect `R2L` examples; and one ensemble of one CART and one BN to detect `DoS` examples — with each classifier trained on its resp. reduced set of features; an approach quite similar to the one used in the present paper. This was then extended by adding a hybrid model composed of SVM and decision trees (DT) to the ensemble [9]. However, the hybrid model did not seem to help much.

Third, in [10], fuzzy rule-based classifiers, linear genetic programming (LGP), DT, SVM, and an ensemble were evaluated using feature selection to reduce the number of variables of the dataset to 12. The fuzzy rule-based classifier outperformed the other methods when trained on all 41 features, while LGP seemed more appropriate when using a smaller feature set. The ensemble was composed of one DT in charge of the `Normal` instances, one LGP each for `Probe`, `R2L` and `DoS`, and one fuzzy set of rules for `U2R`. The results obtained with the ensemble were very encouraging with accuracy $> 99\%$ for all classes (on the data subset).

Folino *et al.* [11] instead used the entire KDD99 dataset and examined the performance of a system composed of several genetic programming ensembles distributed on the network based on the island model. The system showed average performance for the `Normal`, `Probe` and `DoS` classes, but very low for the `U2R` and `R2L` classes.

The key conclusion from all these works is that ensemble approaches generally outperforms approaches in which only one algorithm is used. An ensemble is a very efficient way to compensate for the low accuracy of a set of weak learners. Moreover, feature selection should provide specific subsets to train algorithms specialised in the detection of one particular class of attacks. Mukkamala *et al.* [12]–[14] identified the five most important features for each class of attacks. The features were selected using SVM, LGP and MARS. Surprisingly, neither `protocol_type` nor `service` was selected by the three algorithms for the `DoS` class. In contrast, Kayacik *et al.* [15] concluded that those features were the most significant for that classs, even though their experiments were conducted on hierarchical self-organizing maps (SOM).

III. Experiments

The problem of intrusion detection can be divided into five distinct subproblems, one for each class of instances (`Normal` and the four types of attacks: `Probe`, `U2R`, `R2L`, and `DoS`). Here each problem will be handled by one or more algorithms of an ensemble, allowing each subproblem to be treated separately in the experiments and to join the solutions to the subproblems into a general solution for the problem of intrusion detection. A dataset for each attack subproblem was built by sampling a number of examples in one class of attacks and the same number in the class `Normal` in order to have a balanced dataset with 50% anomalous and 50% normal examples (no algorithm was explicitly designed to detect normal traffic). A balanced dataset is necessary to avoid the problem of skewed classes where the accuracy of the predictor can be made artificially high by increasing the number of instances from one of the classes.

For the classes of attacks with few examples, `R2L` and `U2R`, the entire set was selected. For the `Probe` class, 10,000 instances were selected randomly. This number was chosen to have a significant sample with as many different examples as possible without affecting the training time too much. The `DoS` training set contains 3,883,370 instances, with ‘neptune’ and ‘smurf’ attacks counting for the majority (with resp. 1,072,017

and 2,807,886 instances). The other types of attacks have much smaller number of examples, e.g., the type of DoS called 'land' is represented only 21 times. For this reason, samples of 5,000 examples each were selected randomly from the 'neptune' and 'smurf' sets. All examples of the other types of attacks were included for a total of 13,467 DoS instances. For all four classes, the same number of Normal instances was selected. The experiments performed are in direct continuity of the work done by Mukkamala *et al.* [12]–[14], which identified the key features relevant to each of the four classes of attacks. The first step of the experiments was to assess the sets of features selected in [12]. Then in a second round of experiments those sets were fed to an ensemble of machine learning algorithms. All models were evaluated by 10-fold cross-validation.

A. Experimental Setup

Figure 1 shows the model for the ensemble used in this work. First, the network packet being analysed is sent to four different detector modules, one each for Probe, R2L, U2R, and DoS. Each module executes a preprocessing step to extract a number of features from the packet; the set of features varies depending on the module (as further described in Section III-B). The extracted features are then dispatched to different decision trees which have been previously trained with the same features on the training set, as shown at the top of the figure for the Probe detector. Each decision tree is a binary classifier which outputs 0 if the packet is considered normal traffic and 1 if the packet is classified as anomalous. A vector of dimension n containing the output of n classifiers is then fed to the module decision function. In the figure $n = 4$, but it could be any number of algorithms.

Finally, a vector of dimension 4 containing the output of each module is fed to the ensemble decision function which combines the results and outputs a value describing if the packet is considered normal or anomalous, and if anomalous from which class of attacks. The easiest situations are when all modules, or all modules except one, output Normal. In the former case, the system classifies the packet as normal. In the latter, the system classifies the packet as anomalous and is able to unambiguously identify the class of attack concerned. If more than one module classify the packet as anomalous, it will be more difficult for the network administrator to understand which class of attack the anomalous packet belongs to.

The resulting model is an ensemble of ensembles with feature selection applied independently for each module. However, in this work, we will not be concerned with the decision functions for each module. Instead, we will evaluate the intersection of the sets of false positives and false negatives produced by the four algorithms in each module. This will give us the optimal performance that each module could achieve. The most important advantages of this model is the possibility to execute the algorithms in parallel and the modularity allowing the exchange of any algorithm of the ensemble without any modification of the rest.

B. Feature Selection Assessment

In the first experiment, several classifiers were trained with different number of features. The goal of the experiment was not to find the best algorithm possible and fine-tune it, but rather to conclude on how well an algorithm performs with a smaller set of features. In this case, it is only natural to use exactly the same setting for the algorithms and to compare the performance based only on the sets of features. Five decision trees were trained with different sets of features. The results obtained represent the performance of the algorithms on the cross-validation set which is extracted from the training set. The second experiment assessed performance on the test set.

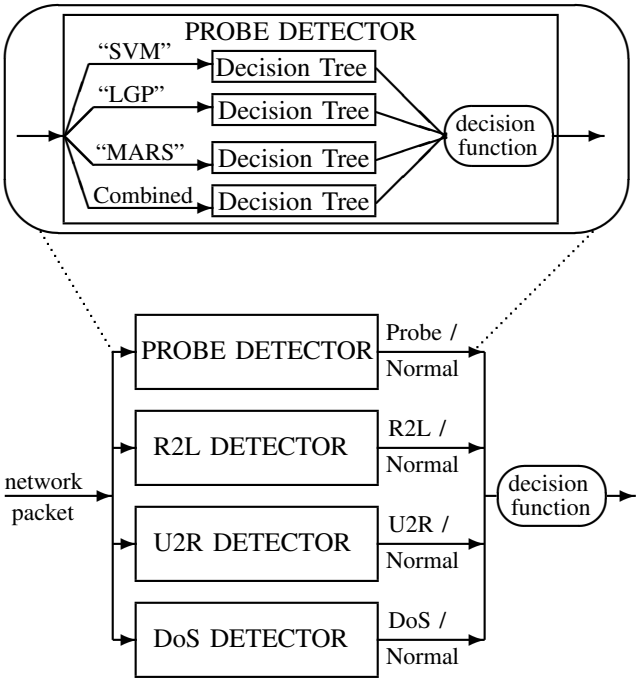

Fig. 1. Model of the ensemble

The first classifier was trained with all 41 features in the dataset. The next three were trained with 5 features selected in [12] for each class of attacks by the three algorithms support vector machines (SVM), linear genetic programming (LGP) and multivariate adaptive regression splines (MARS). The last classifier was trained on a "combined" set of features: the union of the feature sets selected by the three algorithms. The number of features in the "combined" set is 11 for Probe, 14 for U2R, 11 for R2L and 12 for DoS. These additional sets help bringing down the number of false positives and false negatives, as we will see in the results of the experiments.

As displayed in Figure 1, the algorithm used as a classifier was decision tree (DT). Attempts were also made to use an SVM with a Gaussian radial basis function kernel. However, it could have given an advantage to features picked by SVM when used for feature selection, and it seemed that the choice of SVM greatly affected the set of features selected by MARS. Furthermore, SVM was much slower than DT, roughly two orders of magnitude both for training and classification. In particular classification time is an important criterion to take into account when building a real-world application.

The results obtained in terms of accuracy are shown in Table II and can be compared to those obtained with 41 features by [9] using the same decision tree. For the class Probe, the

TABLE II
ACCURACY OF THE FEATURE SELECTION ASSESSMENT

Classifier	Probe	U2R	R2L	DoS
DT: 41 features	99.865	93.000	99.022	99.948
DT: 5 SVM features	99.815	96.000	98.578	93.346
DT: 5 LGP features	99.320	90.000	97.378	98.689
DT: 5 MARS features	99.750	97.000	98.044	99.863
DT: combined features	99.895	96.000	98.933	99.948

TABLE III
FEATURE SELECTION ASSESSMENT: FALSE POSITIVES AND NEGATIVES

	Probe		U2R		R2L		DoS	
Classifier	FP	FN	FP	FN	FP	FN	FP	FN
DT: 41 features	12	17	4	3	17	10	6	8
$\texttt{ensemble}_{\max}$	0.7	3	0.3	0.3	6.6	0.5	0	1.6

accuracy is exactly the same as in [9]: 99.86%. The classifiers trained with sets of 5 features are not far behind the one trained with all 41. The reduced feature sets seem to be a good choice when the algorithms are trained using decision trees. However, the classifier fed with the 5 features selected by LGP performs slightly worse than the others and could be replaced by a more accurate algorithm.

The results for U2R are worse than for Probe, but this was expected: each false positive and false negative has a larger impact on the general accuracy due to the small number of examples. The results are much better than the 68.00% accuracy obtained by [9] on U2R. However, the classifier trained on features selected by LGP again perfomed poorly. Interestingly, the algorithms trained on the features selected by SVM and MARS out-performed the one trained on all features. This is probably since 41 features are too many to generalize from given the small number of examples.

The results for R2L are similar to those obtained for Probe, even though the number of instances in the dataset is much smaller. The results are also much better than [9] who obtained 84.19% accuracy on this class. This experiment clarifies that classifying Probe attacks and R2L attacks are two very distinct problems, even if they are both intrusions, which is why they should be treated separately. Again, the selected features seem to be a good choice even if a little drop of accuracy can be observed compared to Probe. The classifier trained on the features selected by MARS has a high rate of false positives and the one trained on features selected by LGP has the lowest accuracy, but also a lower false positive rate which implies a higher false negative rate.

DoS also shows better results than [9] who obtained 96.83% accuracy. The classifier trained on features selected by SVM obtained the worse score, whereas features selected by MARS gave the best score after the set of all features and the combined feature set. This is important since there is a set of 5 features that can perform almost as well as the full feature set even on larger number of training examples.

The overall numbers of false positives (FP) and false negatives (FN) drop significantly when using more than one algorithm, as Table III shows. For the FP and FN analysis, we call $\texttt{ensemble}_{\max}$ the number of examples wrongly classified by all three algorithms trained on sets of 5 features and the one trained on the "combined" feature set. This is the maximum an ensemble composed of these four algorithms could achieve if the combination of their individual results was optimal; here calculated by taking the intersection of the set of examples misclassified for each algorithm. The experiment was run ten times for each class of attacks to ensure accuracy of the results and to find the types of attack in each class misclassified most of the time by $\texttt{ensemble}_{\max}$.

All types of Probe attacks appear at least once as an FN, however, 'satan' and 'portsweep' seem to be the most difficult attacks to detect. When comparing the problematic instances of 'satan', 'portsweep' and 'ipsweep' with true instances of the same attack types, it seems that src_bytes is the feature that gives the classifiers most trouble. In fact, for probe attacks, src_bytes should be very small if not equal to zero; when an example of these attacks has a high value for src_bytes, it goes undetected. This is a big problem since an attacker could easily fill the packets of the attack with random bytes to evade the IDS. It could seem like a good idea to get rid of this feature; however, src_bytes is very important to detect Probe attacks: the only classifier that performs poorly is the one trained on the features selected by LGP, a feature set that does not include src_bytes.

For the U2R class, in general either one FP or one FN appears in each test run. The FP can be explained by the small number of examples in the dataset, only 52 Normal examples are present. The FN is always a 'rootkit' attack which is wrongly classified as normal traffic, but it is not always the same instance, indicating that some information is missing for the decision tree to classify 'rootkit' attacks. These can be any kind of malware such as worm, Trojan or virus with the ability to hide its presence and actions to the users and processes of a computer; this is called a stealth attack. The diversity found in malware probably has a huge impact on the problem. Moreover, there are only 10 'rootkit' attacks in the dataset, increasing the difficulty. Examining the values of these examples for the 14 features of the combined algorithm revealed that almost all 10 instances have very different values for those features. The $\texttt{ensemble}_{\max}$ performs perfectly in most cases, but it is difficult to conclude anything with such a small dataset: One FP or FN out of 10 instances of the cross-validation set is quite a bad score.

The combination of all algorithms helps to bring down the number of false positives and false negatives also for R2L, but these numbers are again too high for a real-world application. There are eight different types of R2L attacks represented in the training set. After running the experiments ten times, only three types of these attacks trigger false negatives for the $\texttt{ensemble}_{\max}$: 'spy', 'imap' and 'phf'. There is not much documentation about 'spy' attacks which are not even represented in the test set. However, the signatures of 'imap' and 'phf' are described in [16]. Detection of these attacks

TABLE IV
ACCURACY OF THE MODEL ASSESSMENT

Classifier	Probe	U2R	R2L	DoS
DT: 41 features	93.087	90.000	50.000	79.345
DT: 5 SVM features	77.628	40.000	50.000	87.698
DT: 5 LGP features	87.482	83.571	61.033	76.105
DT: 5 MARS features	84.037	85.000	50.000	82.200
DT: combined features	79.969	94.286	50.000	85.361

requires very specific features. In the case of an 'phf' attack, the IDS "must monitor http requests watching for invocations of the phf command with arguments that specify commands to be run" [17]. None of the 41 features in the KDD99 dataset gives any information about a specific command being run on the system. It would be impractical to do so for each specific command vulnerable to an attack. However, this could be the reason why the machine learning algorithms are incapable of detecting these kind of attacks with certainty. There are two ways to solve this problem, either new features could be added to the dataset or an IDS using signatures of attacks should perform the detection for these particular types of attacks. In the former case, the new features should not be too specific to ensure that new attacks could also be identified. In the second case, the IDS loses its ability to detect similar attacks but its accuracy increases. To detect an 'imap' attack, an IDS should be "programmed to monitor network traffic for oversized Imap authentication strings" [17]. This seems more within reach of our IDS, since `service` and `src_bytes` are both represented in the feature set.

$\texttt{ensemble}_{\max}$ was highly successful on the `DoS` class, returning zero FP. Table III shows that the number of FN is reduced as well. Three types of attacks trigger FN: 'smurf', 'neptune' and 'back'. The first two rarely appear in the list; however, the third seems to be the most difficult type to handle. This is not a surprise, since to detect a 'back' the IDS must look for a big number of frontslashes ("/") in the request URL [16]. There are no features in the dataset taking this particularity into account. Consequently, the model has to rely on other features to make up for the lack of information, leading to an imperfect result. Nevertheless, as expected, $\texttt{ensemble}_{\max}$ brings robustness to the accuracy of the IDS.

C. Model Assessment

In the second round of experiments, several classifiers were trained with different number of features on examples from the training set. Again the algorithm used as classifier was decision tree. The goal of the experiment was to evaluate the model used in the previous experiment on the test set after training on the same number of examples as selected for the training set in each class in the first experiment. As discussed in Section II-A, the test set is composed of many examples with unseen attacks (attacks that are not represented in the training set). The experiment aimed to assess if the ensemble was capable of generalizing to new types of attacks belonging to the same classes as the ones previously seen.

TABLE V
MODEL ASSESSMENT: FALSE POSITIVES AND FALSE NEGATIVES

	Probe		U2R		R2L		DoS	
Classifier	FP	FN	FP	FN	FP	FN	FP	FN
DT: 41 features	86	490	3	11	0	16,347	69	7,268
$\text{ensemble}_{\max}$	11.4	524	1.6	1	1	7,779	16.6	688

In most cases, the accuracy of all algorithms degraded drastically in comparison to the first experiment as shown in Table IV, where the values represent one run of the program. In particular, the set of features selected by SVM obtains the worst results, and does not seem to generalize well to new types of attacks. The set selected by LGP managed to keep a respectable accuracy on the `Probe` class, while all classifiers except SVM showed results very similar to those in the feature selection experiments on `U2R`, with the "combined" set of features being the best one, outperforming even the algorithm trained with all 41 features in the same way that was observed in the feature selection experiment.

Particularly bad results could be expected for `R2L` because of the poor distribution of attacks in the training set, and Table IV confirms this: the accuracy of all algorithms is equal or close to the 50% guessing baseline. Most of the attacks are 'warezclient' (1020 out of 1126 in total for the `R2L` training set) leaving only 106 instances of all other attack types (seven different types) to train on — and 'warezclient' is not even represented in the test set. There is no chance that the models built would perform well on new attacks (or even on old) with this limited training set. Also the results for `DoS` were much worse than in the first experiment, with the set of features selected by LGP obtaining by far the worst results. Nevertheless, all other algorithms performed better than the one trained with all features. As Table V shows, the $\texttt{ensemble}_{\max}$ is able to handle part of the new attacks, but does not recognize them as easily as the old ones, and the number of false negatives is very high for most classes.

For `Probe`, the most surprising fact is that the attack 'ipsweep' seems to go undetected almost all the time. This result is unusual because 'ipsweep' was available in the training set and did not cause any trouble in the previous experiment. One reason for this would be in the examples of 'ipsweep' from the test set were very different from the ones in the training set. However, after examining the training set carefully, typical values for the features of an 'ipsweep' attack were observed, and it appears that the values of 'ipsweep' in the test set are in the same range as those in the training set. To conclude, the results are not as bad as they look. First, almost all old attacks are perfectly detected, especially 'portsweep' and 'satan' which triggered FN in the first experiment are now absent from the attacks triggering FN. The new attacks are detected most of the time, but the number of FN is still too high to be useful in a real-world application. Finally, solving the problem of 'ipsweep' would substantially bring down the number of FN.

For `U2R`, the $\texttt{ensemble}_{\max}$ brings down the number of FP to 1 and the number of FN to 0 with an average value of 1.6 and 1 respectively over five runs of the program. As expected,

sometimes a 'rootkit' attack goes undetected, just as in the first experiment. Besides, 'ps' also rarely appears as an FN. The most surprising result comes from undetected 'buffer overflow' even though it never happened in the previous experiment. However, 'xterm' and 'sqlattack' are detected all the time which is good because it means that the $\texttt{ensemble}_{\max}$ generalizes well for the U2R class.

The number of FN for R2L explodes. Old and new types of attacks are similarly misclassified. The only conclusion that can be drawn is that the R2L training set contains too few examples of each type of attack to be of any help.

For DoS, the major part of FN are due to new attacks. 'pod' is the only old attack that regularly triggers a few FN, while other old attacks such as 'smurf' and 'neptune' sometimes trigger FN, but the number of FN for those are very low. New attacks are more problematic, with 'mailbomb', 'apache2', 'processtable' and 'udpstorm' recurrently triggering FN, even if most of these attacks are detected in general. Even though its generalization power is limited, $\texttt{ensemble}_{\max}$ performed quite well overall on unseen DoS attacks and helped bring down both FP and FN. This is quite an improvement, but again not enough for a real-world application.

IV. Discussion and Future Work

The Feature Selection Assessment experiments showed that the ensemble approach is indeed a very powerful paradigm that can be used to bring down the number of FP and FN. The lower accuracy observed by individual algorithms is countered by the union of their results. Even with sets containing only five features, the results are very encouraging. Moreover, treating each class of attack as a different problem solved by a specialised algorithm seems to work well when compared to strategies using one algorithm to detect all classes of attacks. "Divide and conquer" and "Unity is strength" seem to be opposite views, but they are actually both applied in this work with impressive results. In general, algorithms using fewer features have slightly lower accuracy and prediction time but much lower training time. The results obtained by Mukkamala *et al.* [7] seem to be correct. However, the features selected by LGP give the worst result in most cases except for DoS where it is the feature set selected by SVM which performs poorly. Consequently, the sets of features selected by LGP should be reconsidered for all classes except DoS, while the set of features selected for DoS by SVM should be replaced. The number of different types of attacks that go undetected is very small and only few examples of these attacks are problematic. Most of the time, the problem lays in the lack of information contained in the dataset. Some attacks require very specific features and should probably be handled by specialized programs or signature-based IDSs. The class Probe is a bigger problem since most of the attacks belonging to this class exploit a legitimate feature used by network administrators; as a result, all types of Probe attacks trigger FN at some point, even though 'portsweep' and 'satan' are the most problematic.

A smaller feature set means that less information must be extracted from a network packet in the data preprocessing phase. Since the accuracy is not lowered too much in the best cases, this is a huge improvement that could be used in real IDSs. Moreover, the union of all algorithms using fewer features tremendously improves the accuracy: on average over ten runs of the program, only 0.7 FP and 3 FN were observed for Probe over 20,000 examples, 6.6 FP and 0.5 FN for R2L over 2,252 examples, 0.3 FP and 0.3 FN for U2R over 104 examples, and 0 FP and 1.6 FN for DoS over 20,000 examples. Even though these results are much better than what could be achieved with a single algorithm, they are still quite far from being useful in a real-world application where the false positives and negatives should be < 1 for some 15 millions examples in a 10Gb/s Ethernet network. Arguably, 90% of the 15 millions examples will be normal traffic containing no attack at all, but $\texttt{ensemble}_{\max}$ still has to be improved to stand a chance against clever hackers.

The results described above are the best that an ensemble composed of these algorithms and sets of features could achieve. In its current state, there is no point in building an experiment to assess a real combination of the results of the individual algorithms in the $\texttt{ensemble}_{\max}$. Further work will have to be carried out to find the best suitable algorithms and sets of features. Nevertheless, it is interesting to see how well this $\texttt{ensemble}_{\max}$ can perform when predicting previously unseen attack types. That was the topic of the second round of experiments, on Model Assessment. Even if $\texttt{ensemble}_{\max}$ in general helps tremendously to bring down the numbers of false positives and false negatives, it is still far from reaching the accuracy appropriate for a real-world application. In particular, datasets which are not carefully designed are proven to be useless in building accurate models of the attacks. This is the case with the R2L training set which contains mainly examples of the 'warez_client' attack which is not even represented in the test set and very few examples of all other types of attacks. The performance of $\texttt{ensemble}_{\max}$ was acceptable for the classes of attacks U2R and DoS. The performance on the Probe class was also standard, even though 'ipsweep' attacks went undetected for unknown reasons. Overall, we can say that the results of this second round of experiments were not very satisfying, but once again proved the usefulness of the ensemble approach.

In the future, particular attention has to be paid to the features relevant to each attack. New features carrying meaningful information about the attacks must be designed to help the machine learning algorithms to successfully classify all types of attack. The DoS and Probe classes are mostly characterized by time-related features, whereas R2L and U2R mostly are characterized by content-related features extracted from the payload of the network packets.

V. Conclusion

The aim of this work was to show that ensemble approaches fed with appropriate features sets can help tremendously in reducing both the number of false positives and false negatives. In particular, our work showed that the sets of relevant features are different for each class of attacks which is why it is important to treat those classes separately. We developed our own IDS to evaluate the relevance of the sets of features selected by Mukkamala *et al.* [12]. This system is an ensemble of four ensembles of decision trees. Each of the four ensembles

is in charge of detecting one class of attacks and composed of four decision trees trained on different sets of features. The first three decision trees were fed with sets of five features selected in [12]. The last decision tree was fed with the union of these three sets of five features from which the redundant features were removed.

The experiments showed that these sets were appropriate in most cases. In the first experiment, the set of features selected by linear genetic programming gave the worst results, except for the class `DoS` for which the set of features selected by SVM performed poorly. The second experiment gave less interesting results because of the inappropriate distribution of examples between the training and test sets of the KDD99 data. In particular, the ensemble could not generalize properly on the `R2L` class because the training set mainly contains a type of attack that is not represented in the test set. In both experiments, we looked at the number of instances that were misclassified by all four algorithms in order to obtain a result from the best combination of these algorithms. Further work would be required to develop a real decision function combining the results of the different algorithms. However, since the accuracy obtained here was not good enough for a real-world application, designing decision functions was unnecessary. Nevertheless, we are convinced that this work is heading in the right direction in order to overcome the limitations of current intrusion detection systems.

Finally, a thorough analysis of the examples that were misclassified by the ensemble was performed, in particular highlighting the types of attacks that were systematically misclassified by the ensemble. By looking at the signature of these attacks, we were able to find out the reasons for the classification errors. In most cases, the attacks displayed very specific features not captured by the set of variables in the dataset. These attacks should probably be handled by a specialized system or new variables should be developed to train the classifiers.

Acknowledgements

The authors would like to express their thanks to Esteban Zymanyi, Olivier Markowitch and Liran Lerman (all at Université Libre de Bruxelles) for valuable comments.

References

[1] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, Washington, DC, Jun. 2010, pp. 305–316.

[2] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proceedings of the 2nd International Conference on Computational Intelligence for Security and Defense Applications*, Ottawa, Ontario, Canada, Jun. 2009, pp. 53–58.

[3] S. X. Wu and W. Banzhaf, "The use of computational intelligence in intrusion detection systems: A review," *Applied Soft Computing*, vol. 10, no. 1, pp. 1–35, 2010.

[4] E. Bahri, N. Harbi, and H. N. Huu, "Approach based ensemble methods for better and faster intrusion detection," in *Proceedings of the 4th International Conference on Computational Intelligence in Security for Information Systems*, Torremolinos-Málaga, Spain, Jun. 2011, pp. 17–24.

[5] S. González, J. Sedano, A. Herrero, B. Baruque, and E. Corchado, "Testing ensembles for intrusion detection: On the identification of mutated network scans," in *Proceedings of the 4th International Conference on Computational Intelligence in Security for Information Systems*, Torremolinos-Málaga, Spain, Jun. 2011, pp. 109–117.

[6] P. Zhang, X. Zhu, Y. Shi, L. Guo, and X. Wu, "Robust ensemble learning for mining noisy data streams," *Decision Support Systems*, vol. 50, no. 2, pp. 469–479, Jan. 2011.

[7] S. Mukkamala, A. H. Sung, and A. Abraham, "Intrusion detection using an ensemble of intelligent paradigms," *Journal of Network and Computer Applications*, vol. 28, no. 2, pp. 167–182, Apr. 2005, special issue on computational intelligence on the internet.

[8] S. Chebrolu, A. Abraham, and J. P. Thomas, "Feature deduction and ensemble design of intrusion detection systems," *Computers & Security*, vol. 24, no. 4, pp. 295–307, 2005.

[9] S. Peddabachigari, A. Abraham, C. Grosan, and J. P. Thomas, "Modeling intrusion detection system using hybrid intelligent systems," *Journal of Network and Computer Applications*, vol. 30, 2005.

[10] A. Abraham, R. Jain, J. P. Thomas, and S.-Y. Han, "D-SCIDS: Distributed soft computing intrusion detection system," *Journal of Network and Computer Applications*, vol. 30, no. 1, pp. 81–98, 2007.

[11] G. Folino, C. Pizzuti, and G. Spezzano, "An ensemble-based evolutionary framework for coping with distributed intrusion detection," *Genetic Programming and Evolvable Machines*, vol. 11, pp. 131–146, 2010.

[12] S. Mukkamala, A. Sung, and A. Abraham, "Cyber security challenges: Designing efficient intrusion detection systems and antivirus tools," in *Enhancing Computer Security with Smart Technology*. USA: CRC Press, 2005, pp. 125–161.

[13] S. Mukkamala and A. H. Sung, "Identifying significant features for network forensic analysis using artificial intelligent techniques," *International Journal of Digital Evidence*, vol. 1, no. 4, pp. 1–17, 2003.

[14] A. H. Sung and S. Mukkamala, "The feature selection and intrusion detection problems," in *Proceedings of the 9th Asian Conference on Advances in Computer Science*. Chiang Mai, Thailand: Springer-Verlag, 2004, pp. 468–482.

[15] H. Gunes Kayacik, A. Nur Zincir-Heywood, and M. I. Heywood, "A hierarchical SOM-based intrusion detection system," *Engineering of Applied Artificial Intelligence*, vol. 20, no. 4, pp. 439–451, Jun. 2007.

[16] K. Kendall, "A database of computer attacks for the evaluation of intrusion detection systems," in *DARPA off-line intrusion detection evaluation, proceedings DARPA information survivability conference and exposition*, 1999, pp. 12–26.

[17] ——, "Intrusion detection attacks database," Webpage (last accessed: June 24, 2012), 2007, http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/docs/attackDB.html.