Adaptive Intrusion Detection System via Online Learning

Conference Paper · December 2012

DOI: 10.1109/HIS.2012.6421346

2 authors:

Hai Thanh Nguyen
National Institute for Research in Computer Science and Control
35 PUBLICATIONS **400** CITATIONS
SEE PROFILE

Katrin Franke
Norwegian University of Science and Technology
108 PUBLICATIONS **1,607** CITATIONS
SEE PROFILE

Adaptive Intrusion Detection System via Online Learning

Hai Thanh Nguyen and Katrin Franke
Norwegian Information Security Laboratory
Gjøvik University College, Norway
hai.nguyen@hig.no, katrin.franke@hig.no

ABSTRACT

Adaptation of Intrusion Detection Systems (IDSs) in the heterogeneous and adversarial network environments is crucial. We design an adaptive IDS that has 10% higher accuracy than the best of four different baseline IDSs. Rather than creating a new 'super' IDS, we combine the outputs of the IDSs by using the online learning framework proposed by Bousquet and Warmuth [2]. The combination framework allows to dynamically determine the best IDSs performed in different segments of a dataset. Moreover, to increase the accuracy and reliability of the intrusion detection results, the fusion between outputs of the four IDSs is taken into account by a new expanded framework. We conduct the experiments on two different datasets for benchmarking Web Application Firewalls: the ECML-PKDD 2007 HTTP dataset and the CISIC HTTP 2010. Experimental results show the high adaptability of the proposed IDS.

General Terms

Information security, Machine Learning

Keywords

Intrusion detection, Web attack detection, Online machine learning, Adversarial Learning, Adaptability

1. INTRODUCTION

Network environments become more and more diverse with the present of many different network protocols, services, applications and so on. With this diversity, many different types of attacks appear and target to a computer or a network everyday. A single type of intrusion detection systems (IDSs), which has its own advantages and disadvantages, seems to be insufficient to detect all the attacks. For example, misuse IDSs, such as SNORT [15], can only detect attacks, which are described in the databases. This approach is known as a signature-based IDS. The anomaly IDSs [11] can detect novel attacks based on significant deviations of the observed activities from the established profiles of the legitimate users. However, at the same time these IDSs generate more false positives than the misuse IDSs. In addtion, it is difficult to produce effective models of legitimate patterns by expert knowledge. Several data-driven IDSs by using machine learning [22] are developed to cope with these problems. But this approach strongly depends on the utilized classifiers and the IDSs might perform well in a particular network environment with particular attack types. For example, the Decision-Tree-based IDS detects well the denial-of-service attacks, but it fails to identify most of the user-to-root and remote-to-local attacks [16]. Of course this happens with the particular KDD CUP 1999 dataset [8], yet it might likely happen in many other datasets. This phenomenon is according to No-Free-Lunch Theorem [20, 21] that says there is no universal classifiers for every kinds of data.

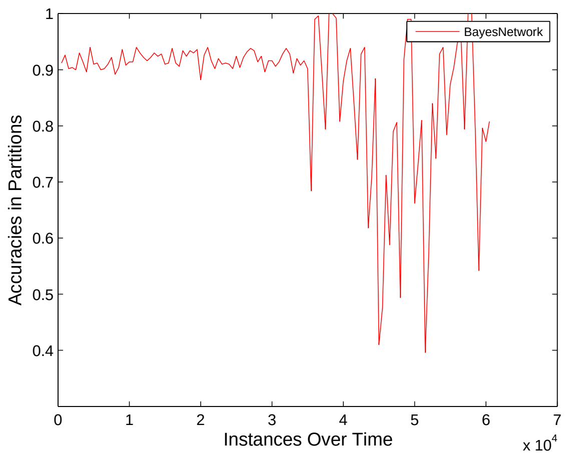

Figure 1: Accuracies of the BayesNetwork-based Intrusion Detection System (calculated w.r.t a partition of 500 instances). The IDS is unable to detect all attacks well.

Figure 1 shows a sample on the performance of a single Bayesian-Network-based IDS deployed to detect several Web attacks, such as SQL Injection, LDAP Injection and so on, in the ECML-PKDD 2007 dataset [14]. It is obvious that this machine-learning-based IDS performs well in some segments of the dataset, but cannot detect all the Web attack.

In the adversarial network environments, the situation is even worse as attackers might launch and change the types of attacks every time, thus the streams of the data are unpredictable. Since we don't know which types of attacks are coming next, the primary difficulty lays on selecting of the best IDS at a certain time. In our scenario, we assume that each IDS has its own favorite types of attacks to detect.

One might think to run several IDSs at the same time and then either select the IDS that works well on average, or select the one that provides the best performance in the previous data that is believed to be representative. However, this manual selection costs a lot of efforts and usually are incorrect since the stream of the data in the heterogeneous and adversarial network environments are changed all the time.

One might also think to automatically combine the outputs of the IDSs. A simplest way to combine the IDSs' outputs is the majority voting. At any time, the final decision will be the prediction of the most frequent output. This approach is very efficient, but incorrect in many cases since the majority can be wrong. Several efficient fusion techniques for ensemble IDSs (see, for example, [4]) were proposed. However, there is no theoretical guarantees about the ability of the ensemble IDSs to follow the best IDS or to follow the sequence of the best IDSs in different segments of the data. This ability is understood as the adaptability of the IDSs. Another combination method named Hedge/Boosting [6], which is a very efficient online learning framework. It has also the theoretical guarantee that says the combination of IDSs will have the performance closed to the performance of the best IDS for the whole period of time T. However, this approach is sensitive in the adversarial environments, in which attackers intend to destroy the superiority of the best IDS by changing the attacks all the time. After T times, all the baseline IDSs will perform badly the same, thus the combination by means of the Hedge/Boosting might not be useful. This adversarial activity targeted the ensemble IDSs can be included into the Exploratory category [1] for online machine learning.

In this work, we propose to apply another efficient online learning framework, which is the mixing algorithm introduced by Bousquet&Warmuth (2002) [2], to combine different outputs of the baseline IDSs. We call the new proposed method an Adaptive Intrusion Detection System (A-IDS). The mixing algorithm consists of two steps: the *Loss Update* and the *Mixing Update*. The *Loss Update* is essentially the Hedge algorithm [6]. The beauty of the *Mixing Update* is that it efficiently remembers the IDSs that perform well in the past through the weight values and quickly recover them to deal with their favorite data at the current moment. Therefore, the A-IDS quickly adapts to the changing environments, such as heterogeneous and adversarial network environments. More details on how and why does the algorithm work are given in the next section.

We apply the new proposed approach A-IDS to the Web attack detection problem. First, we select four different classifiers, which are the NaiveBayes, BayesNetwork, Decision Stump and RBFNetwork [5], to build four different baseline IDSs. The reason of our selection is their simplicity and efficiency that are very important for high-speed IDSs. The built IDSs then perform on two different benchmark datasets: ECML-PKDD HTTP 2007 [14] and CSIC HTTP 2010 [12]. The outputs of the IDSs are fused by an expanded framework and then combined by using the mixing algorithm. We also compare the proposed A-IDS with several existing online approaches, such as majority voting, Hedge/Boosting algorithm and each individual IDS. Some off-line ensemble learning methods, such as Adaboost [6], are not considered in the comparison. The obtained results show that the studied A-IDS adapts quickly to the changing of data as it outperforms all other algorithms. In particular, the A-IDS has 10% higher accuracy than the best of 4 different base machine-learning-based IDSs.

The paper is organized as follows. Section 2 describe the proposed adaptive intrusion detection system in more detail and explain its adaptability. We present some experimental results in Section 3 and some discussions in Section 4. The last section summarizes our findings and discuss about future works.

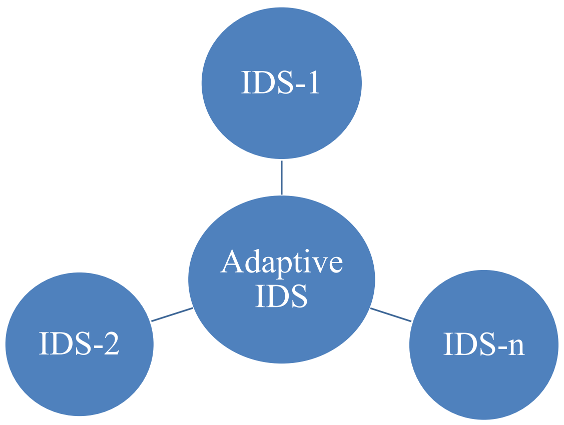

Figure 2: Adaptive IDS

2. ADAPTIVE INTRUSION DETECTION SYSTEM

In this section, we describe the new adaptive intrusion detection system (A-IDS) and explain how and why does it work.

The new adaptive IDS is essentially the central of the framework as depicted in Figure 2. Every time it receives the outputs from different base IDSs, fuse them and combine them to make its own decision. Afterwards, the true label of the current pattern arrives and the losses measuring the discrepancy between IDSs' outputs and the true label are incurred. The adaptive IDS then maintains a weight value for each of the base IDSs, which is the estimation of the quality of the IDS's output. The maintenance is based on the losses of the IDSs and is called the weight update strategy.

This proposed model is, in fact, the expert framework to combine predictions of expert advices (see, for example, [3, 10, 17]). Here the experts are the intrusion detection systems and their advices are the IDSs' outputs. In this paper, we update the weights of the IDSs by using the mixing algorithm that contains the *Loss Update* and the *Mixing Update* proposed by Herbster and Warmuth [7] and more recently developed by Bousquet and Warmuth [2]. The detail of the weight update strategy applied to intrusion detection problems is given below.

2.1 The Mixing Algorithm

Let T be the number of instances and n be the number of the base intrusion detection systems (IDSs). At a certain time or trial $t = 1, ..., T$, the adaptive intrusion detection system (A-IDS) receives a vector x_t of n IDS's outputs: $x_t = (x_{t,1}, ..., x_{t,n})$, where $x_{t,i} \in \{0, 1\}$ is the output of the i-the IDS. If $x_{t,i} = 1$ (or $x_{t,i} = 0$), then the i-the IDS predicts the current instance an attack (or a normal instance). The A-IDS then produces its own prediction $pred(t)$ and, following that, receives the true label y_t ($y_t \in \{0, 1\}$) of the t-th instance at the time t.

A loss function L is used to measure the discrepancy between the true label and the predictions of the base IDSs. At a trial t, the loss value $L_{t,i}$ of i-th IDS is defined as follows: $L_{t,i} = (y_t - x_{t,i})^2$.

The weight vector maintained by the A-IDS at trial t is denoted by $v_t = (v_{t,1}, v_{t,2}, .., v_{t,n})$, where $v_{t,i} \geq 0$ is the weight value of the i-th IDS and the sum of these values equal to 1, i.e. $\sum_{i=1}^{n} v_{t,i} = 1$. In the mixing algorithm, the maintenance of the weight vector v_t consists of two steps: *Loss Update* and *Mixing Update*. In the first step, the weight of the i-th IDS that provides incorrect prediction is multiplied by a factor $e^{-\eta L_{t,i}}$ and then renormalized. Here η is the learning rate parameter and is selected in advance. The second step mixes the current weight vector v_t^m with the past average weight vector $av_t = \sum_{q=0}^{t-1} v_q^m/t$ by following linear combination: $v_{t+1} = (1-\alpha)v_t^m + (\alpha)av_t$, where α is a parameter in $(0, 1)$.

Based on the inner product $\hat{y}_t = (v_t, x_t)$ between the weight vector v_t and the output vector x_t of the base IDSs, the prediction $pred(t)$ of the A-IDS is defined as 0 if $0 \leq \hat{y}_t \leq 0.5$ and as 1 otherwise. Obviously, at any trial t the prediction of the A-DIS is always governed by the IDS that has the maximal weight value.

The mixing algorithm applied to intrusion detection problems is summarized as follows:

1. **Parameters:** $\eta > 0$; $0 \leq \alpha \leq 1$

2. **Initialization:** $v_1 = v_0^m = \underbrace{(\frac{1}{n}, ..., \frac{1}{n}, ..., \frac{1}{n})}_{n\ times}$

3. **FOR** $t = 1$ **TO** T **DO**

   (a) **Prediction:** Let $\hat{y}_t$ be an inner product: $\hat{y}_t = (v_t, x_t)$. We define the prediction of the A-IDS:

   $$pred(t) := \begin{cases} 0,\ if\ 0 \leq \hat{y}_t \leq 0.5, \\ 1,\ if\ \hat{y}_t > 0.5. \end{cases}$$

   (b) **Loss Update:** with $L_{t,i} = (y_t - x_{t,i})^2$

   $$v_{t,i}^m = \frac{v_{t,i}e^{-\eta L_{t,i}}}{\sum_{j=1}^{n} v_{t,j}e^{-\eta L_{t,j}}}$$

   (c) **Mixing Update:**

   $$v_{t+1} = \sum_{q=0}^{t-1} \alpha v_q^m/t + (1-\alpha)v_t^m$$

It has been shown in previous works (see, for example, [3, 10]) that the *Loss Update* helps to find the best IDS from a pool of n base ones. However, this update is a double-edge knife: driving the weights of IDSs which have currently wrong predictions ($L_{t,i} = 1$) to zero so quickly as it uses the exponential value $e^{-\eta L_{t,i}}$. Therefore, it becomes very difficult for these IDSs to recover if they start doing well. To explain this, it can be seen from the *Loss Update* that the current weight $v_{t,i}^m$ of the i-th IDS depends on the previous weight $v_{t,i}$. We assume that the i-th IDS performed poorly in the previous time, thus the weight $v_{t,i}$ is small. If the IDS starts predicting well, which means the loss value $L_{t,i}$ equals to zero ($L_{t,i} = 0$) and the factor $e^{-\eta L_{t,i}}$ equals to 1 ($e^{-\eta L_{t,i}} = 1$), the small value $v_{t,i}$ still governs the value $v_{t,i}^m$. As a consequence, the algorithm with only the *Loss Update* slowly adapts to the changes in the performance of the IDSs or in the changes of the data assuming each IDS has its own favorite data.

The situation is even worse in the adversarial network environments where attackers intend to destroy the superiority of the best IDS and challenge its adaptability by changing the attack patterns all the time. As different IDSs classify well different types of attacks, the best IDS changes overtime. In this case, the quick recovery property of the IDSs to deal with their favorite data is very important. This property is understood as the adaptability of the IDSs in the heterogeneous and adversarial network environments. Fortunately, Herbster and Warmuth [7], and more recently Bousquet and Warmuth [2] have studied this property and proposed a *Mixing Update* in addition to the *Loss Update*. By remembering the past average weight vector av_t, the *Mixing Update* allows to keep the good IDSs for quickly recovery when necessary. It also allows currently poor IDSs, which were good in the past, maintain a slightly higher weight than those which have proved less useful overtime, thus a proper IDS is called. Formal proofs of the algorithms are provided in [2, 7]. In the next section, we show the application of the A-IDS to the Web attack detection problem.

3. EXPERIMENTS

In this section, we show the adaptation of the new proposed intrusion detection system via experiments conducted on two different datasets for benchmarking Web Application Firewalls: ECML-PKDD 2007 [14] and CSIC 2010 HTTP datasets [12]. In order to do that, we first introduce these two datasets in more details. The features extracted by using expert knowledge about Web attacks are described. We then present the experimental settings that aim to show the limitations of existing approaches and motivate the application of the new proposed adaptive intrusion detection system. Finally, we discuss about experimental results.

3.1 Datasets

The ECML-PKDD 2007 data set was generated for the ECML-PKDD 2007 Discovery Challenge [14]. In this experiment, we utilized the training set, which is composed of 50,000 samples including 20% of attacks (i.e. 10,000 attacks and 40,000 normal requests). The requests are labeled with specifications of attack classes or normal traffic. The classes of attacks in this data set are: Cross-Site Scripting, SQL Injection, LDAP Injection, XPATH Injection, Path traversal, Command Execution and SSI attacks.

Table 1: Number of instances in the datasets

	CSIC 2010	ECML-PKDD 2007
Number	61.000	50.000

Moreover, we additionally used our own generated CSIC 2010 data set [12] for experiments. The CSIC 2010 data set contains traffic targeted to a realistic Web application developed for this purpose. It consists of an e-commerce Web application running on an Apache server and it is composed of several Web pages that allow users to do actions such as buying items with a shopping cart or registering by providing their personal data. Some of the Web pages require certain parameters, for example the user's name and address for the registration process or the name of the product that the user wants to buy. The dataset is automatically generated and it contains normal and anomalous requests to all the Web pages composing the e-commerce Web application, with different values for those Web pages including parameters. In total, 36,000 normal requests and more than 25,000 anomalous requests are included in the dataset. All these requests are labeled (either as normal or as anomalous). The CSIC 2010 HTTP dataset includes modern Web attacks such as SQL injection, buffer overflow, information gathering, CRLF injection, XSS, server side include and parameter tampering.

By utilizing our expert knowledge about Web attacks, we listed 30 features that we considered relevant for the detection process (see Table 3 in Appendix). Some features refer to the length of the request, the length of the path or the headers, as length is important for detecting buffer-overflow attacks. We also observed that the non-alphanumeric characters were present in many injection attacks. Therefore, we took four types of characters into account: letters, digits, non-alphanumeric characters and other characters. As the non-alphanumeric characters have a special meaning in a set of programming languages, thus in Table 3 we refer to them as 'special' char. We analyzed the appearance of the characters in the path and in the argument's values. We also studied the entropy of the bytes in the request. Additionally, we collected the keywords of several programming languages that were often utilized in the injection attacks and counted the number of their appearances in different parts of the request as a feature. All the names of 30 features from expert knowledge for Web attack detection are given in Table 3 in Appendix.

3.2 Experimental Design

The experimental design consists of three parts:

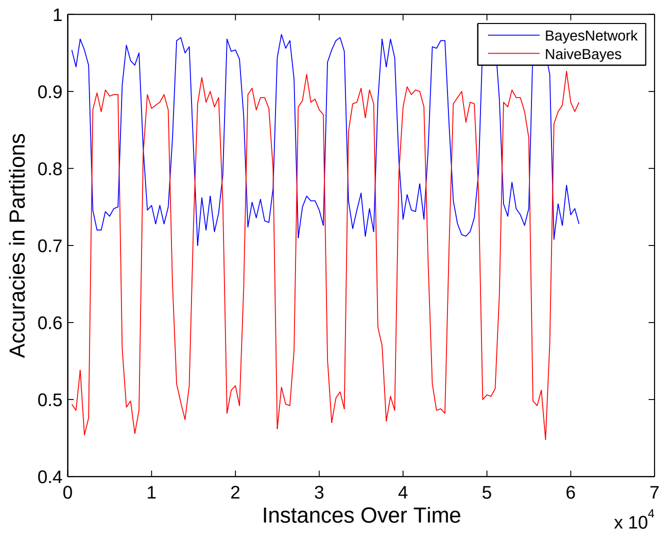

Figure 3: Accuracies of the BayesNetwork and NaiveBayes-based IDSs (calculated w.r.t a partition of 500 instances). The two IDSs perform alternatively in different segments of the dataset.

1. *Expert setting:* This part aims to show the drawback of using a single intrusion detection system to detect many different types of Web attacks. We also demonstrate that the *Loss Update*, which was described in section 2, is unsuitable for an adaptive IDS in the adversarial network environments.

2. *Expert combining:* This part is the setting to test the adaptive IDS proposed in section 2.

3. *Expert expanding:* In this part, we introduce a new expanded framework to increase the accuracies and reliability of intrusion detection results.

Ad. Expert setting To achieve the aims, we select four different machine learning classifiers, which are NaiveBayes, BayesNetwork, Decision Stump and RBFNetwork [5] to build four different IDSs. The reason of our selection is their simplicity and efficiency which are very important for intrusion detection. By using the Weka tool [19], we apply the 10-folds cross-validation to run these four classifiers on the datasets with extracted features. The IDSs' predictions of each instance in the datasets are generated to combine. But before this experiment, we make sure that all the attack and normal instances are mixed to simulate the adversarial network environments where different types of attacks change overtime. The obtained results, for instance, on the CSIC 2010 HTTP dataset show that non of the baseline machine-learning-based IDS has a higher accuracy than 83%, which is low in some senses.

Figure 3 visualizes the performances of the NaiveBayes and BayesNetwork-based IDSs. Obviously, the single intrusion detection (NaiveBayes-based or BayesNetwork-based) cannot detect all types of Web attacks very well. However, each of them provides good accuracies in different segments

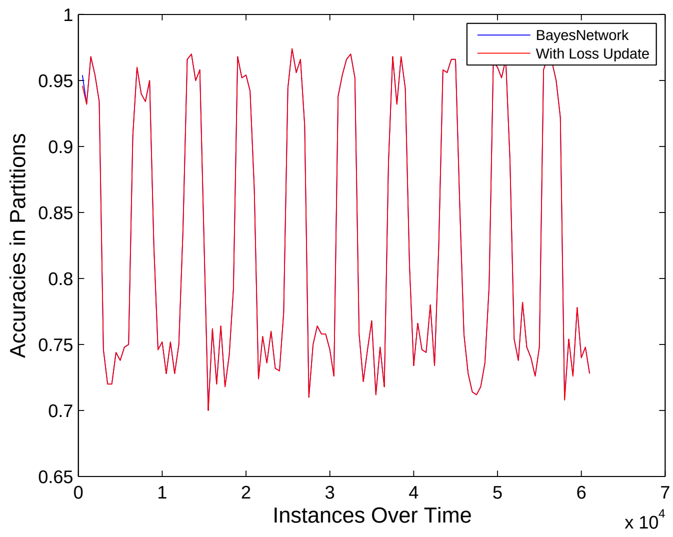

Figure 4: Accuracies of the BayesNetwork-based IDS and the IDS with *Loss Update* (calculated w.r.t a partition of 500 instances). Both IDSs perform almost the same.

of the dataset. An appropriate online combination of their outputs is needed.

We now apply the *Loss Update* strategy to combine the outputs of the four base IDSs in each trial. In order to do that, we select $\eta = 0.1$ and run the algorithm described in section 2 with a replacement of the *Mixing Update* by the following $v_{t+1,i} = v_{t,i}^m$ for keeping the weights after the *Loss Update*. The obtained results show that this combination doesn't help to improve the accuracy of intrusion detection as it provides the performance which is almost the same as the best IDS. Figure 4 visualizes the performance of the BayesNetwork-based IDS, which is the best IDS on the CSIC 2010 HTTP dataset in this experiment, and the performance of the combination with only *Loss Update*. It can be seen that these two accuracies are almost the same, except at the beginning the combination with the *Loss Update* provides worse accuracies than the BayesNetwork-based IDS' accuracies. Clearly, this combination approach doesn't take advantages of each base IDS in different segments of the data. Therefore, the *Loss Update* is inappropriate for an adaptive IDS in the changing and adversarial environments.

Ad. Expert combining In this part, we describe the application of the Bousquet and Warmuth's algorithm to build an adaptive intrusion detection system. But before the experiment, we randomly permute the instances of the datasets for ten times to mimic the adversarial environment. With the 10-folds cross-validation, the four selected classifiers perform on ten obtained different datasets by using Weka tool [19] and their accuracies are recorded. We then test the adaptive IDS proposed in section 2 for each of ten cases and summarize the results.

To run the mixing algorithm for A-IDS as described in section 2, we select the parameters $\eta = 0.1$ and $\alpha = 0.001$. The running time is fast since the computational complexity of the algorithm is linear in the number of instances T as the number of IDSs is small ($n = 4$). There are a number of *Mixing Updates* [2] with various recovery properties. However, in this experiment we chose the Uniform Past update, which is describe in section 2, because of its simplicity and efficiency.

Ad. Expert expanding In the new expanded framework, which we call A-ExIDS, the fusion between IDSs' outputs is taken into account before running the mixing algorithm. In fact, we propose to expand n outputs to all $2^n - 1$ possible output combinations as follows: suppose that we have a set S that contains m $(0 < m < n)$ outputs $O_S(t), t = \overline{1, m}$ to combine. The output O_S of the new IDS_S is defined:

$$O_S := \begin{cases} 0, \ if \ 0 \leq \sum_{t=1}^{m} O_S(t)/m \leq 0.5, \\ 1, \ if \ \sum_{t=1}^{m} O_S(t)/m > 0.5. \end{cases}$$

In our experiment, the number $n = 4$ is small and we have 15 IDSs' outputs in total, thus the algorithm is still efficient. After expanding, we run the mixing algorithm for the A-ExIDS with selected parameters $\eta = 0.1$ and $\alpha = 0.001$.

All the obtained results are given and discussed in the next section.

3.3 Experimental Results

In this section, we describe and discuss about the experimental results. Table 2 shows the performances of four single machine-learning-based IDSs, of the combination with only *Loss Update* and of the new proposed combination with *Mixing Update*. We also compare the new approach with the majority voting and Hedge/Boosting methods.

Table 2: Accuracies

	CSIC 2010	ECML-PKDD 2007
NaiveBayes	72.78± 0.01	85.12± 0.03
BayesNetwork	82.79± 0.03	86.95± 0.025
Decision Stump	74.73± 0.05	84.27± 0.07
RBFNetwork	72.46± 0.01	87.69± 0.04
Majority Voting	81	83
Hedge/Boosting	82.1± 0.04	86.3± 0.05
A-IDS	90.52± 0.06	91.27± 0.01
A-ExIDS	90.98± 0.05	92.56± 0.02

From the Table 2, it can be seen that the new adaptive IDS (A-IDS) outperforms each individual base IDS, the Majority Voting and the Hedge/Boosting algorithm. For example, in the case of the CSIC 2010 dataset it provides almost 10% and 8% higher accuracies than the best IDS, which is the BayesNetwork-based IDS, and the Hedge/Boosting algorithm, respectively. The A-ExIDS has the best accuracies in both datasets. Even though, this is slightly higher performances than the A-IDS, it illustrates the potentials to increase the accuracy and reliability of the intrusion detection results by considering the interaction between IDSs. However, when the number n of the base IDSs becomes large, the computational complexity of the A-ExIDS exponentially

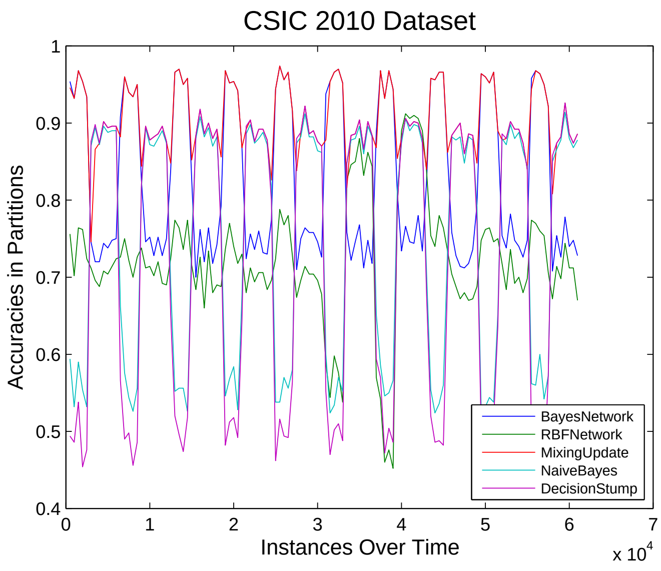

Figure 5: Accuracies of the IDSs (calculated w.r.t a partition of 500 instances) on the CSIC 2010 dataset). The A-IDS (Mixing Update) follows the best IDSs in different segments.

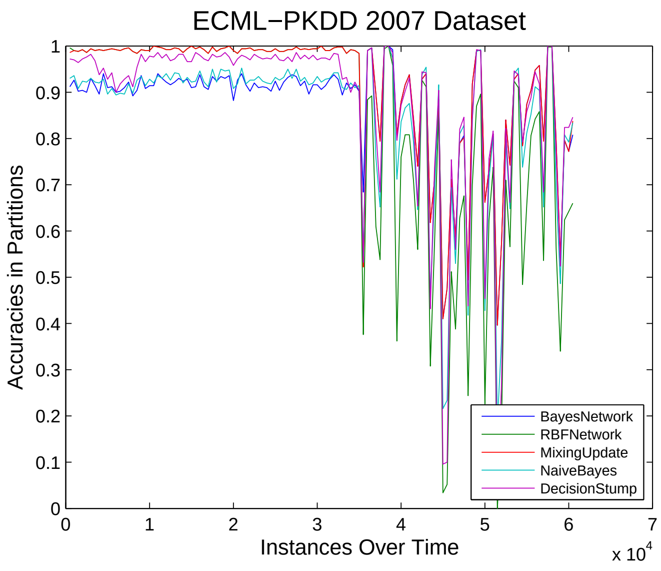

Figure 6: Accuracies of the IDSs (calculated w.r.t a partition of 500 instances) on the ECML-PKDD 2007 dataset). The A-IDS (Mixing Update) follows the best IDSs in different segments.

increases. In our experiment, the number n is small ($n = 4$), which likely happens in the real world because of the expense to run a large number of IDSs.

The significant improvements of the A-IDS and the A-ExIDS can be explained through the their adaptability, which is visualized, for example, in Figure 5 and Figure 6.

From Figures 5 and 6, it is clear that the new A-IDS follows the best IDS in each segment of the datasets. That means it adapts to the changing of the data instances.

4. RELATED WORKS AND DISCUSSIONS

In this section, we discuss about related works to our algorithm in the literature.

One of the first works on adaptive intrusion detection systems has been studied by Lee et. al. in [9]. In this work, the authors proposed to combine different types of features that are considered as models to detect different types of attacks. A machine learning classifier is trained on the dataset with feature vectors. Experimental results showed the adaptability of the machine-learning-based IDS. However, there wasn't any theoretical analysis about this property in the paper.

The idea of combination to increase the adaptability and robustness of the IDSs have been developed and confirmed further in many later works. For example, multiple classifier systems for intrusion detection (see, for example, [4]) were studied well and empirically demonstrated to be a good choice for adaptive IDSs. However, it is difficult to understand the adaptation and robustness properties of these systems.

A more recent work by Wang et. al. [18] proposes an online-boosting-based intrusion detection system in changing environments. The idea of their method is to apply the online boosting algorithm introduced by Oza [13], which is an online version of the Adaboost algorithm [6].

5. CONCLUSIONS AND FUTURE WORKS

We have designed an adaptive intrusion detection system (A-IDS) that can detect many different types of attacks in the heterogeneous and adversarial network environments. Rather than creating a new universal IDS, the outputs of the different base IDSs are combined by using the online learning framework proposed by Bousquet and Warmuth.

We tested the new proposed A-IDS in solving the Web attack detection problem by conducting an experiment on two HTTP datasets: the ECML-PKDD 2007 and the CSIC 2010 datasets. The obtained results showed the high adaptability of the A-IDS.

There are two possible avenues for future works. First, it would be interesting to answer the question on whether the parameters α and η can be tuned online or automatically selected. Second, we would like to apply the proposed framework to other information security problems, such as botnet malware detection, network attack detection and so on.

6. ACKNOWLEDGMENTS

This research was done while the author was visiting Machine Learning Laboratory, UC at Santa Cruz. The author is grateful to Manfred K. Warmuth for many useful discussions. This research is supported by The Research Council of Norway, Leiv Eiriksson Mobility Programme 2012 with project number 216801/F11.

7. REFERENCES

[1] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar. The security of machine learning. *Machine Learning*, 81(2):121–148, 2010.

[2] O. Bousquet and M. K. Warmuth. Tracking a small set of experts by mixing past posteriors. *Journal of Machine Learning Research*, 3:363–396, Mar. 2003.

[3] N. Cesa-Bianchi, Y. Freund, D. Haussler, D. P. Helmbold, R. E. Schapire, and M. K. Warmuth. How to use expert advice. *Journal of the ACM*, 44(3):427–485, May 1997.

[4] I. Corona, G. Giacinto, and F. Roli. Intrusion detection in computer systems using multiple classifier systems. In O. Okun and G. Valentini, editors, *Supervised and Unsupervised Ensemble Methods and their Applications*, volume 126 of *Studies in Computational Intelligence*, pages 91–113. Springer Berlin / Heidelberg, 2008. 10.1007/978-3-540-78981-9_5.

[5] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2 edition, Nov. 2001.

[6] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, Aug. 1997.

[7] M. Herbster and M. K. Warmuth. Tracking the best expert. *Machine Learning*, 32(2):151–178, 1998.

[8] W. Lee, S. J. Stolfo, and K. W. Mok. A data mining framework for building intrusion detection models. *Security and Privacy, IEEE Symposium on*, 0:0120, 1999.

[9] W. Lee, S. J. Stolfo, and K. W. Mok. Adaptive intrusion detection: a data mining approach. *Artificial Intelligence Review*, 14:533–567, 2000.

[10] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Journal of Information and Computation*, 108(2):212–261, Feb. 1994.

[11] T. F. Lunt and R. Jagannathan. A prototype real-time intrusion-detection expert system. In *Proceedings of the 1988 IEEE conference on Security and privacy*, SP'88, pages 59–66, Washington, DC, USA, 1988. IEEE Computer Society.

[12] H. T. Nguyen, C. Torrano-Gimenez, G. Álvarez, S. Petrovic, and K. Franke. Application of the generic feature selection measure in detection of web attacks. In *The 4th International Conference, Computational Intelligence in Security for Information Systems, CISIS*, pages 25–32, 2011.

[13] N. C. Oza. Online ensemble learning. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence, July 30 - August 3, 2000, Austin, Texas, USA*, page 1109, 2000.

[14] C. Raissi, J. Brissaud, G. Dray, P. Poncelet, M. Roche, and M. Teisseire. Web analyzing traffic challenge: description and results. In *Proceedings of the ECML/PKDD 2007 Discovery Challenge*, pages 47–52, 2007.

[15] M. Roesch. Snort: Lightweight intrusion detection for networks. In *LISA*, pages 229–238. USENIX, 1999.

[16] M. Sabhnani and G. Serpen. Why machine learning algorithms fail in misuse detection on kdd intrusion detection data set. *Intell. Data Anal.*, 8(4):403–415, Sept. 2004.

[17] V. G. Vovk. Aggregating strategies. In *Proceedings of the third annual workshop on Computational learning theory*, COLT '90, pages 371–386, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc.

[18] Y. Wang, W. Hu, and X. Zhang. Online boosting based intrusion detection in changing environments. In *Proceedings of the 2008 International Conference on Artificial Intelligence, ICAI 2008, July 14-17, 2008, Las Vegas, Nevada, USA*, pages 735–741, 2008.

[19] Weka Machine Learning Project. Weka. URL http://www.cs.waikato.ac.nz/˜ml/weka.

[20] D. H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Comput.*, 8(7):1341–1390, Oct. 1996.

[21] D. H. Wolpert. The supervised learning no-free-lunch theorems. In *In Proc. 6th Online World Conference on Soft Computing in Industrial Applications*, pages 25–42, 2001.

[22] S. X. Wu and W. Banzhaf. Review: The use of computational intelligence in intrusion detection systems: A review. *Appl. Soft Comput.*, 10(1):1–35, Jan. 2010.

APPENDIX

Table 3: Names of 30 features from expert knowledge for Web attack detection [12]

Feature Name	Feature Name
Length of the request	Length of the path
Length of the arguments	Length of the header “Accept”
Length of the header “Accept-Encoding”	Length of the header “Accept-Charset”
Length of the header “Accept-Language”	Length of the header “Cookie”
Length of the header “Content-Length”	Length of the header “Content-Type”
Length of the Host	Length of the header “Referer”
Length of the header “User-Agent”	Method identifier
Number of arguments	Number of letters in the arguments
Number of digits in the arguments	Number of ’special’ char in the arguments
Number of other char in the arguments	Number of letters char in the path
Number of digits in the path	Number of ’special’ char in the path
Number of other char in path	Number of cookies
Minimum byte value in the request	Maximum byte value in the request
Number of distinct bytes	Entropy
Number of keywords in the path	Number of keywords in the arguments