An Investigation of the Hoeffding Adaptive Tree for the Problem of Network Intrusion Detection

Diego Guarnieri Corrêa*, Fabrício Enembreck† and Carlos N. Silla Jr.†
*Graduate Program in Informatics (PPGI) – Federal University of Technology of Paraná (UTFPR)
Cornélio Procópio, PR, Brazil 86300–000
Email: diego.guarnieri@gmail.com
†Graduate Program in Computer Science (PPGIa) – Pontifical Catholic University of Paraná (PUCPR)
Curitiba, PR, Brazil 80215–901
Email: fabricio@ppgia.pucpr.br, carlos.sillajr@gmail.com

***Abstract*—Intrusion detection in computer networks is a important topic in information security. Due to numerous cases of security breaches that caused economic and social losses in recent years, this topic has been the subject of several studies in order to mitigate problems related to network intrusion and computer attacks. Information security systems have been using different techniques for network intrusion detection. However, with the development of communication mechanisms and consequently with the increase in data traffic, some techniques used for intrusion detection lost their information processing capability. The emergence of new forms of attacks on computer systems also contribute to the depreciation of some of the existing tools. In this scenario, new techniques capable of processing large amounts of information and that perform proactive discovery of new attack vectors are necessary. This paper presents a study on the use of a data stream mining technique known as Hoeffding Adaptive Tree to create a predictive model for network intrusion detection. The experiments performed in this work show the effectiveness of this technique when applied to a database of computer network attacks.**

I. Introduction

The popularization of networked services and the Internet leave room for the creation and dissemination of malicious character tools. These tools seek loopholes in computer systems to create channels of undue and unrestricted access.

In recent years, the protection of computer systems has become a key factor for the sovereignty of a country [1]. Attacks on such systems can lead to the collapse of basic services such as health and transport or result in economic losses.

Computer faults, also known as attack vectors, can be found in low critical systems such as mobile devices and even in operating systems of large servers. Such vulnerabilities can bring different consequences, from issues related to banking fraud to disclosure of confidential and restricted information. Amzi *et al.* [2] show the organization's flagship event called *WikiLeaks*, which from various computer failures exploited by remote workers, exposed thousands of secret documents from different countries.

Most of the vulnerabilities currently found are explored via a network connection. This is due to the great amount of exposure to Internet services. Thus, various tools and techniques are created and refined in order to deter attacks on network level and not allow the attacker to get access to that service.

Intrusion detection systems allow the containment and response to unauthorized access [3]. Over the years, various tools have been developed and improved. Some of the best known and most used tools are: Snort [4] and OSSEC (Open Source Security HIDS) [5].

Several heuristic for intrusion detection in computer networks have been tested in recent years. But with the passage of time and the evolution of information systems, some techniques lost their effectiveness. The growing volume of network data traffic and the constant need for quick responses contributed to this issue.

Another factor that contributes to the loss of efficiency of existing intrusion detection techniques today is the evolution of the attacks and tools used by the attackers.

Some approaches to intrusion detection employ data mining techniques to discover attack patterns [6], [7], [8]. In recent years the number of studies of Data Stream Mining (DSM) techniques has also grown. The DSM techniques seek to process large amounts of data using limited computer resources.

Another motivation for the use of DSM techniques with network attack data, is that DSM techniques assume that the predictive model that they build is not stationary, i.e. that the data distribution changes over time.

Another point that makes the use of DSM algorithms interesting is the implementation of these techniques in embedded devices. Currently, low-cost devices such as Arduino[1] and Raspberry PI[2] can be adapted to become control mechanisms and mitigation of attacks.

The main contribution of this work is to present an investigation of DSM techniques to the problem of intrusion detection in computer networks. The remainder of this paper is organized as follows. Sections II, III and IV summarizes the main concepts needed to understand this study. Section V presents the methodology, while Section VI discusses the experiments and results achieved by the proposed approach. Finally, Section VII presents the conclusions of this work.

[1]https://www.arduino.cc
[2]https://www.raspberrypi.org

978-1-5090-6182-2/17/$31.00 ©2017 IEEE

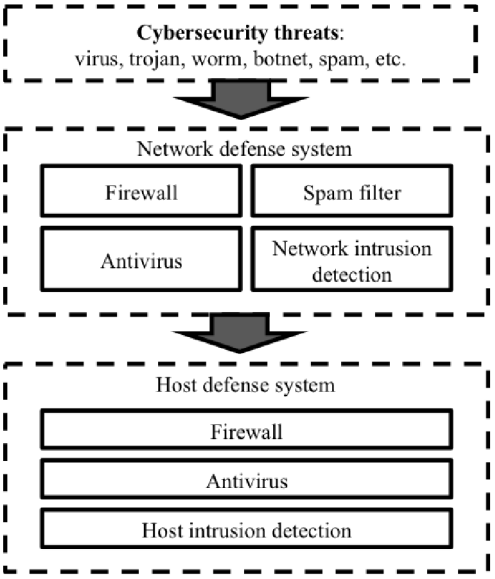


Fig. 1. Cybersecurity process [1]

II. Cybersecurity

The defense of information systems aims to ensure three basic pillars: privacy, integrity and availability [1]. Yan *et al.* [9] define these principles as:

Privacy: Users data and private information must be stored with the guarantee that they will not be disclosed without permission.

Integrity: Integrity refers to the prevention of unauthorized changes to the data. Unauthorized data changes can occur from the injection of malicious code or even via network delivery delays.

Availability: Should ensure full access to information systems for legitimate users and also prevent unauthorized users from using valid credentials. Denial of service attacks, for example, are a violation of availability.

Cybersecurity systems operate on two levels, network and host. Figure 1 shows this process. At the network level elements such as firewall, spam filters and Network Intrusion Detection System (NIDS) are used to contain unauthorized access and attacks. In the host layer, anti-virus systems are widely deployed to contain local threats such as viruses and trojans.

Intrusion detection systems use various techniques for identifying attacks and unauthorized access. Misuse detection and anomaly are the most used.

1) Misuse Detection: Misuse-based intrusion detection is characterized by analyzing network traffic for known attack signatures. This type of approach is also known as Intrusion Detection System (IDS) and it is characterized by having a low number of false positives [10].

In this approach, 0-day attacks, for example, are difficult to detect. This type of attack is characterized by using non-disclosed new faults thereby the base intrusion signature database is not yet aware of this new threat [11].

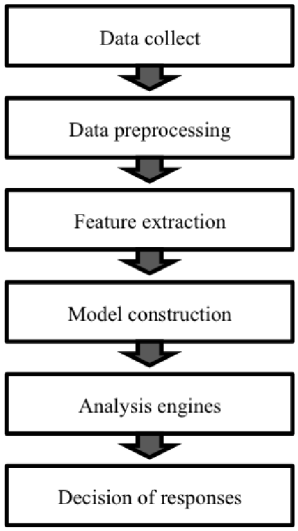


Fig. 2. Data Mining for intrusion detection

2) Anomaly Detection: Anomalies are not in accordance of what is defined as normal behavior patterns and are usually referred to as outliers, discordant observations, exceptions, surprises or quirks [12]. In the level of computer networks, all access that deviates from established normal behavior is labeled as an attack [13].

An intrusion detection system based on anomalies must build a model of normal behavior. The normal model creation process can be seen in Figure 2. This process, which can also be called data mining, initially depicts the data acquisition. For intrusion detection systems, the collected data is usually network traffic flows.

After the collection process, the data must be prepared to go through a feature extraction step, at this stage, only relevant attributes of the collected flows are considered.

In the next stage, the normal behavior model is built. Several techniques are listed in the literature for the production of this model.

The next step is the analysis of data collected from the previously created template. Finally an answer is found for the case of normal access or attack.

III. Data Stream Mining

Data Stream Mining (DSM) is receiving increasing interest from the data mining and related research communities who are interested in addressing issues related to processing large volumes of data [14], [15].

Financial systems, network monitoring, security, telecommunications and Web are examples of applications that have DSM characteristics and that can benefit from the use of such techniques [16], since these applications generate large amounts of data in a continuous and uninterrupted manner.

Unlike conventional data mining techniques, that employ a concept known as batch processing, that is, all the data to be analyzed must be known a priori, DSM is characterized by allowing real-time data processing.

DSM techniques use a concept of incremental processing [17], [18], thus, new data is analyzed as it is received. This flow of data is known as a data stream.

A data stream S can be defined as a sequence of objects $\vec{x}_i$ where $S = \{\vec{x}_1, \vec{x}_2, ..., \vec{x}_\infty\}$. Each instance $\vec{x}_i$ consists of a feature vector v with dimensionality d. Each instance is received at a time $T_1, T_2, \ldots, T_k$.

The aim is that the data streams flow S has to be processed and for each sample $\vec{x}_i$ has a class prediction associated with it. This can be represented by the tuple $(\vec{x}_i, y_i)$, where the predicted class is represented by y_i'. In this case, the prediction may have been correct $(y_i = y_i')$ or incorrect $(y_i \neq y_i')$. Figure 3 shows the model of knowledge extraction used by Data Stream Mining.

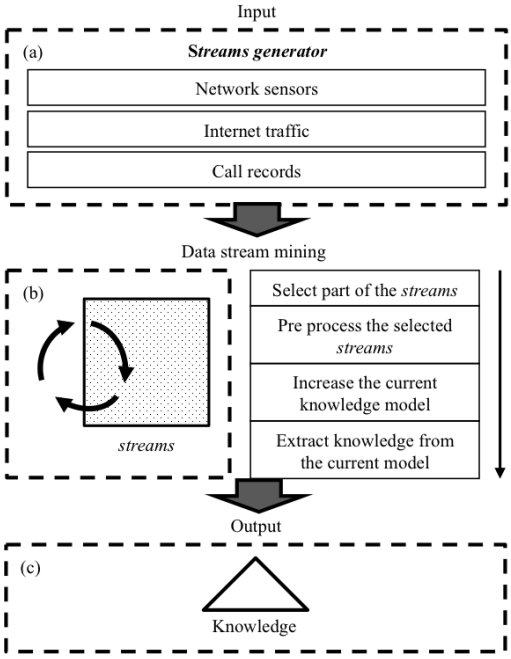


Fig. 3. General model of DSM, adapted from [18]

In Figure 3 (a) it is possible to see the input data, the source of information, can occur from various sources. In (b), a portion of the input data is selected using techniques such as sampling, load shedding, sketching or aggregation [19]. This data is then delivered to an offline algorithm that is responsible for updating the model prediction used in (b). Finally, in (c) the classification result is known for each processed sample.

IV. Hoeffding Adaptive Tree

Within the different DSM algorithms, the first type of decision tree-based data stream mining algorithm is known as the Very Fast Decision Tree (VFDT) [20]. This algorithm is also known as Hoeffing Tree and has essential characteristics for processing data streams:

1) The decision tree is generated incrementally, the contents of the collected data are known dynamically over processing.
2) It has compact data structures, thereby, the processed data is not stored, only relevant statistical information is recorded.
3) Each record is processed only once.

An innovative feature in the VFDT was the use a statistical metric called Hoeffding bound [21]. This limit provides information about when a sample group is large enough to represent a population that is not yet fully known. The Hoeffding bound has an estimated error factor that can be parameterized. Formally it can be defined by Equation 1.

$$\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}} \tag{1}$$

where:

- R is a random variable belonging to the set of real values;
- n is the number of samples $\vec{x}$ processed to date;
- δ is a configurable value that describes the acceptable margin of error of calculation.

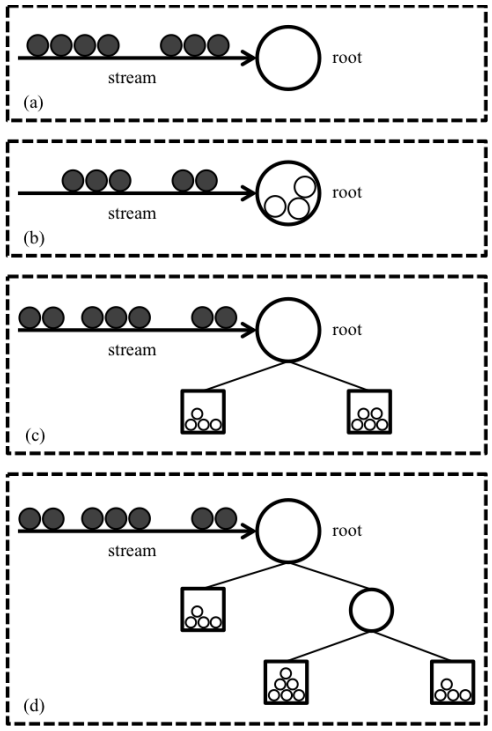


Fig. 4. VFDT construction

The process of construction and growth of a VFDT is shown in Figure 4 and is described as follows: In Figure 4 (*a*) the tree is initialized empty from the root node. In Figure 4 (*b*) each new sample delivered by the stream S, some statistical measures such as mean and standard deviation are calculated, these metrics are always calculated incrementally considering the samples that have been through the same node. Immediately after the sample is processed it is discarded. Figure 4 (*c*) shows that the VFDT has a parameter called n_{min}, when the number of samples in the node exceeds this value, the Hoeffding bound is calculated by the statistics accumulated in

the node. If the condition $\overline{G}(X_a) - \overline{G}(X_b) > \epsilon$ is satisfied, that is, the information gain from the best attribute subtracted from the information gain second best attribute is greater than ϵ, the leaf is split into a decision node. Figure 4 (*d*) shows that the tree grows incrementally with the arrival of new samples.

A VFDT can be compared to classical decision tree algorithms such as C4.5 [22] about their accuracy and error rates [20].

An inherent feature of data stream processing is the concept-drift. Gama *et al.* [23] shows that a data series processed over time will change its behaviour pattern.

One of the limitations found in the VFDT algorithm is the inability of detecting and adapting to these changes. In the work of Bifet and Gavald [24], the authors adapt the VFDT algorithm for detecting concept-drifts and adjusting the prediction model. This technique is known as Hoeffding Adaptive Tree (HAT).

The HAT uses a window to define which data is relevant to the decision nodes. This window is defined based on statistical estimators collected in the processing of the samples. In summary, over time, the oldest samples are forgotten by the algorithm and the new samples renew the knowledge model, adapting the model to the new data.

V. Experimental Details

In this section we describe the materials and methods applied to the experiments conducted in this work. Also, the parameters and settings will be presented.

A. Kyoto Dataset

The *Kyoto 2006* dataset [25] was created from attacks of records related to *Honeypot's* at Kyoto University network.

Spitzner [26] mentions that *Honeypots* are computational resources dedicated to being probed, attacked and committed to an environment that allows the control of these activities. The services offered by *Honepots* should not be disclosed, thereby, all access records directed to these resources can be considered as attacks.

The record of the activities allow the creation of jointless attacks databases, not being necessary the manual analysis of each record.

In the *Kyoto 2006* dataset, these records were collected from November 2006 to August 2009. The collected records of attacks in this period were merged to legitimate access records (mail, web, ftp) to the University servers.

Attacks samples and normal hits have been compiled in records with 24 attributes (categorical and numerical). These attributes represent the network traffic characteristics, such as source IP address and destination, source port and destination and accessible service. Other attributes show analytic characteristics of the database such as the number of sequential samples for the same destination or service.

Table I shows the total records present in the *Kyoto 2006* database divided into classes.

This database can be used to DSM experiments because it contains the attribute "Start time". With this attribute we can sort the records of this base and simulate a data stream. This version of the Kyoto database as used in our experiments is available[3]. As shown in Table I, this database has little imbalance problems, as the two classes have nearly the same amount of samples. In real situations, such as in production networks, this scenario may be different, the normal and attack classes may be completely unbalanced. However, the problem of class imbalance will not be addressed in this study.

TABLE I
Number of Records *Kyoto 2006 dataset*

Class	Total Samples	Average of Sample per Day
Normal	50,033,015	50,335
Attacks	43,043,255	42,874
Total	93,076,270	93,638

B. Tools and Parameter Settings

Some currently available tools implement some DSM algorithms. The framework known as MOA [27] was used in this work to carry out all the experiments. This tool enables the execution of HAT and other DSM algorithms and also capture the necessary information for calculating the evaluating the result.

All algorithms implemented in MOA can have their parameters configured. The VFDT has a number of parameters that can be changed. The parameters used in this work are based on the experimental results obtained in [20] and are described in Table II.

TABLE II
Parameters of HAT

Parameter	Description	Default Value
δ	Maximum error rate	10^7
τ	User-specified threshold for split the leaf	5%
n_{min}	Minimum number of samples for the analysis of leaf	200

C. Evaluation Measures

Traditional data mining evaluation methodologies such as *x-fold cross-validation*, can not be used for the validation of DSM algorithms [28]. DSM have some peculiarities such as potentially infinite data, vast amounts of information and adaptive prediction models.

Some approaches may be used. Kirkby [29] developed a measure for evaluating performance where the amount of memory used by the prediction model is recorded during processing. Bifet *et al.* [30] name this measure as *RAM-Hour*.

A measure of assessment for DSM proposed by Gamma et al. [31], [28] is called *Predictive Sequential Method* (*Prequential*). In this measure the number of samples incorrectly classified is accumulated within a window w of total processed samples.

[3]Available at: https://sites.google.com/site/carlossillajr/resources/

$$P_e(i) = \frac{1}{i} \sum_{k=1}^{i} F(y \neq y') \quad (2)$$

Equation 2 shows that at every time i the classification error can be calculated.

From the use of *Prequential Evaluation Method* several statistical measures can be calculated for the classification of data streams. In the MOA tool only some validation metrics are implemented, such as accuracy.

For this work, a change was made in this tool to generate other performance measures, among them: precision, recall, g-mean and f-measure. With these metrics, it is possible to perform a more detailed analysis of the obtained results. With the changes made in MOA it was also possible to collect detailed results for each class of the database.

VI. Experimental Results

The methodological procedures applied in the experiments performed in this work are as follows:

1) Preparing the *Kyoto 2006* database in the format accepted by the MOA tool.
2) Parameterization of the HAT algorithm described in Section V.
3) Application of the HAT algorithm across the database.
4) Analysis of the results generated by the *Prequential* evaluation method.
5) Quality analysis and understandability of the generated tree.
6) Comparison of the HAT algorithms with other DSM algorithms.

A. Window Size Analysis

As the validation window used by the *Prequential* can be changed, other experiments to determine and evaluate the size of the validation window for *Prequential* were performed. Table III shows a comparison of the obtained results. For these experiments the HAT was executed 4 times with the *Kyoto 2006* database. At each execution the *Prequential* validation window was changed.

TABLE III
Comparative Validation Window Size

Window	Average Precision	Average Recall	Processing Time (s)
1,000	98.59%	98.90%	1,255.15
10,000	98.61%	98.93%	1,222.40
100,000	98.63%	98.94%	1,223.58
200,000	98.64%	98.96%	1,181.60

From the results in Table III we can see that the values of precision and recall suffer very minor variations. Since the processing time can range from 5 to 6% between the largest and smallest validation window. For large databases, larger validation windows do not significantly impact the result and can reduce processing time and resource utilization as a CPU.

B. Tree-Based Model Growth

A key feature of HAT is the ability to perform continuous data processing. Consequently, the growth of the decision tree will also be constant. In Table IV it is possible to observe the growth in the number of nodes present in the decision tree model at each stage of evaluation using *Prequential*.

TABLE IV
Growth of the Tree

Samples	Tree Nodes
10,000,000	3,536
20,000,000	6,457
30,000,000	8,840
40,000,000	10,730
50,000,000	12,028
60,000,000	13,143
70,000,000	14,625
80,000,000	15,984
$\approx$ 90,000,000	16,826

We can see that the end of the 90 million records processed, the tree has size of approximately 17,000 nodes. The graphical representation of this type of structure is compromised with this amount of leaves and decision nodes.

C. Comparison

In order to validate the performance of the HAT algorithm we performed experiments with the following algorithms: kNN, Naive Bayes and Perceptron.

These algorithms were adapted to be used as DSM techniques using the concept of incremental processing. All algorithms are included in the MOA framework.

kNN: kNN is an iterative process that seeks to assign labels to unclassified samples. Labels are attributes the samples that are closer based on their distances [32].

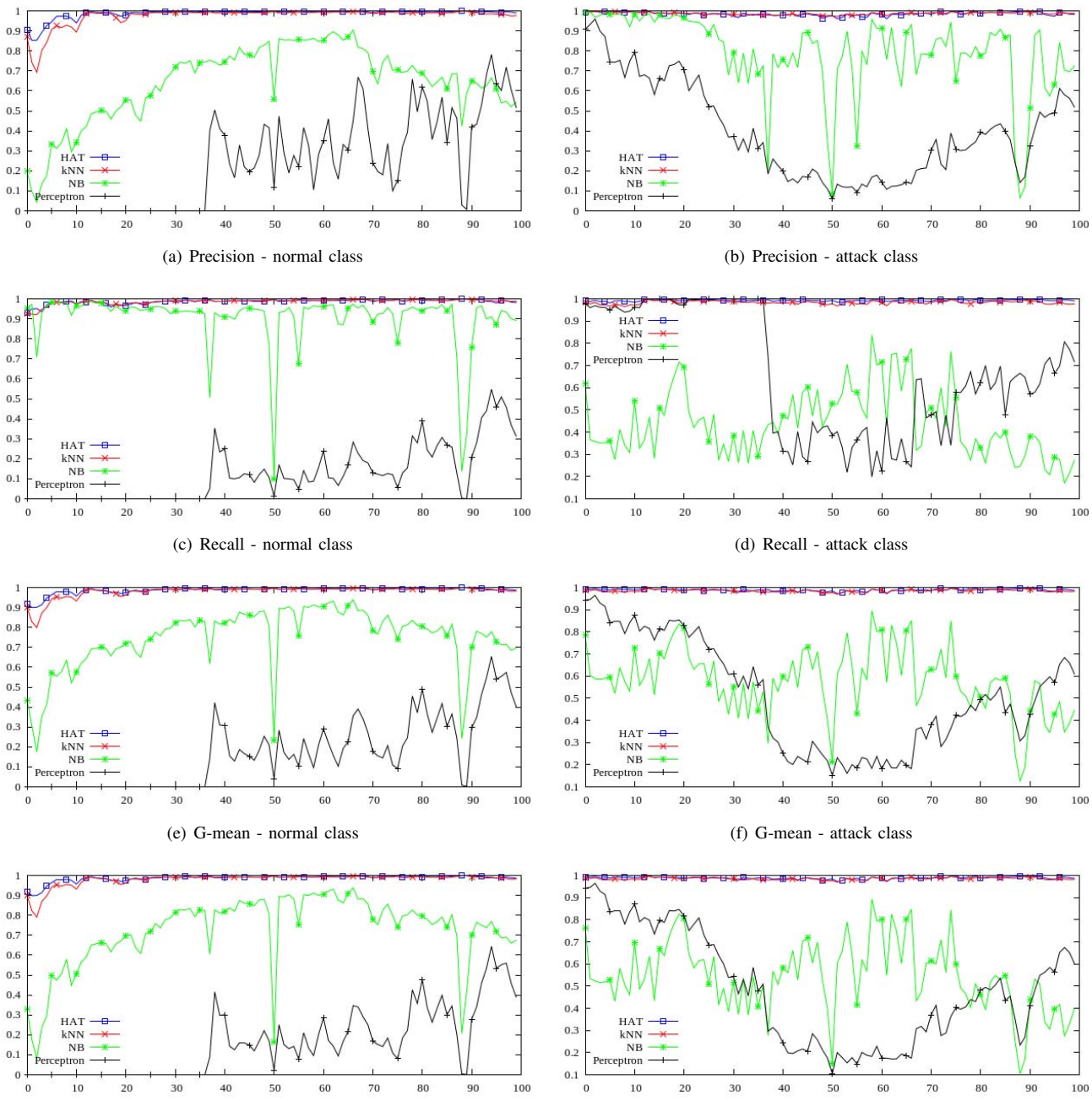
Naive Bayes: Naive Bayes algorithm is based on the principle of Bayes theory [33]. This approach is established in the quantification of the exchanges between the various classification decisions using probability and costs that accompany such decisions. This approach shows the relationship of the conditional probability and its inverse, so tried to estimate a relation of cause and effect.

Perceptron: Perceptron is an algorithm for supervised learning and a kind of artificial neural network. That maps the input data x to output data $f(x)$ by the matrix [34]. The value $f(x)$ is used to classify x.

Table V shows a comparison with the 4 algorithms used to classify the *Kyoto 2006* dataset. The evaluation method used was *Prequential* with 100,000 samples validation window.

Table V shows a unique view of the result obtained by processing nearly 90 million records from the *Kyoto 2006* dataset. However, as the *Prequential* validation principle is the instantaneous analysis of the result of the classification at every i processed samples, a single result value may compromise the understanding of the results.

A more specific analysis of the results can be obtained in Figure 5. These graphics *(a-h)* show the evolution of four



(a) Precision - normal class

(b) Precision - attack class

(c) Recall - normal class

(d) Recall - attack class

(e) G-mean - normal class

(f) G-mean - attack class

(g) F-measure - normal class

(h) F-measure - attack class

Fig. 5. Validation Metrics Results

measures of performance: presicion, recall, g-mean and f-measure. During the processing of the *Kyoto 2006* database using the four algorithms compared in this phase: Hoeffding Adaptive Tree, kNN, Naive Bayes and Perceptron.

As already detailed in Section V, the Hoeffding Adaptive Tree algorithm was parameterized with the standard values found in the literature. The other algorithms did not have their standard parameters altered. For the kNN algorithm, the value of k is equal to 3. For standardization reasons the Hoeffding Adaptive Tree and Naive Bayes algorithms will be called HAT and NB respectively in the graphs and result tables.

Figure 5 *(a-h)* have the x-axis divided into 100 units. For this comparative experiment, *Prequential* was used with a validation window of 1% of the database records. For each 1% of records processed by the each of the algorithms, we compute the evaluation measures. The y-axis represents the value found for each evaluation measure, ranging from 0 to 1.

Dividing the results obtained into a separate chart by the

TABLE V
ALGORITHM COMPARISON

Algorithm	Average Precision	Average Recall
HAT	98.63%	98.94%
kNN	98.07%	98.51%
Naive Bayes	71.66%	67.31%
Perceptron	31.65%	39.61%

database classes, the visualization of the results becomes clearer and more consistent. The use of several evaluation measures also provides insight into different aspects of classification and prediction results.

The analysis of the results presented in Figure 5 show that both HAT and kNN algorithms have high values (normally higher than 0.95%) for the evaluation measures of precision, recall, g-mean and f-measure. It should be noted that for the normal class, during the evaluation of the first 15 million records, all methods (including HAT and kNN) have a lower performance. The performance of the NB classifier varies with the evaluation measure being used, but even when using the recall of the normal class, which provides the best result for the NB, it is clear that its performance is inferior to HAT and kNN. The performance of the Perceptron algorithm obtained the worst results, as up to about 35 million processed samples it classified all samples as belonging to the attack class.

Another point of strong impact on DSM algorithms that should be analyzed is the total processing time. As mentioned before, the analysis of data stream should be online. Thus, a high processing time may compromise the final results. Table VI shows the processing times (in seconds) for each algorithm with a partial every 10 million records processed.

TABLE VI
TIME COMPARISON

Samples	HAT	kNN	NB	Perceptron
10,000,000	128.20s	3746.77s	46.10s	41.99s
20,000,000	249.21s	7233.54s	91.86s	84.61s
30,000,000	374.77s	10861.54s	137.40s	127.87s
40,000,000	499.68s	14543.53s	182.98s	171.89s
50,000,000	624.52s	18324.00s	228.60s	215.64s
60,000,000	747.98s	21983.73s	271.04s	259.29s
70,000,000	874.24s	25745.44s	316.47s	302.58s
80,000,000	1002.79s	29469.97s	361.77s	345.88s
$\approx$ 90,000,000	1072.21s	31458.94s	386.53s	370.02s

VII. CONCLUSIONS

Based on the needs and problems of intrusion detection in computer networks and from experiments performed in this work, it is clear that the use of data stream mining techniques can contribute to the given problem.

The characteristics of the network failure detection problem, as large amount of data, the need for immediate responses and compact models for prediction memory economy fit the attributes of DSM techniques.

Analyzing the various algorithms used in this work there is a clear advantage of HAT and kNN algorithms over others when analyzing different evaluation measures. However, for DSM algorithms, using only the evaluation measures to evaluate the results might not be enough. Issues such as processing time and memory consumption should also be observed.

Following these considerations and according to the results presented in the Section VI, the HAT algorithm outperformed the other algorithms. The results show that the accuracy of HAT prediction model approaches 95% on average using different evaluation measures for the network intrusion detection database used in the experiments. Furthermore, the processing time of each sample in the database is about 10^{-5}, which shows the speed of the algorithm.

One aspect that should be addressed in future research is how to reduce the size of the decision tree model generated by HAT, as in our experiments the number of generated nodes is too high to allow network administrators to understand the generated models.

REFERENCES

[1] S. Dua and X. Du, *Data Mining and Machine Learning in Cybersecurity*. Taylor & Francis, 2011.

[2] I. M. Azmi, S. Zulhuda, and S. Jarot, "Data breach on the critical information infrastructures: Lessons from the wikileaks," in *Cyber Security, 2012 International Conference on Cyber Warfare and Digital Forensic (CyberSec)*, June 2012, pp. 306–311.

[3] J. Allen, A. Christie, W. Fithen, J. McHugh, and J. Pickel, "State of the practice of intrusion detection technologies," Carnegie Mellon University, Tech. Rep. CMU/SEI-99-TR-028, ESC-99-028, 2000.

[4] M. Roesch, "Snort - lightweight intrusion detection for networks," in *Proceedings of the 13th USENIX Conference on System Administration*, ser. LISA '99. Berkeley, CA, USA: USENIX Association, 1999, pp. 229–238. [Online]. Available: http://dl.acm.org/citation.cfm?id=1039834.1039864

[5] R. Bray, D. Cid, and A. Hay, *OSSEC host-based intrusion detection guide*. Syngress, 2008.

[6] R. Rajput, A. Mishra, and S. Kumar, "Optimize intrusion prevention and minimization of threats for stream data classification," in *Proceedings of the 2014 Fourth International Conference on Communication Systems and Network Technologies*, ser. CSNT '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 408–413. [Online]. Available: http://dx.doi.org/10.1109/CSNT.2014.87

[7] J. Song, H. Takakura, Y. Okabe, and Y. Kwon, *Correlation Analysis Between Honeypot Data and IDS Alerts Using One-class SVM*. INTECH Open Access Publisher, 2011. [Online]. Available: https://books.google.com.br/books?id=QE3DoAEACAAJ

[8] G. Xiaoqing, G. Hebin, and C. Luyi, "Network intrusion detection method based on agent and svm," in *Information Management and Engineering (ICIME), 2010 The 2nd IEEE International Conference on*, April 2010, pp. 399–402.

[9] Y. Yan, Y. Qian, H. Sharif, and D. Tipper, "A survey on cyber security for smart grid communications," *IEEE Communications Surveys Tutorials*, vol. 14, no. 4, pp. 998–1010, May 2012.

[10] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on Software Engineering*, vol. SE-13, no. 2, pp. 222–232, 1987.

[11] G. Kumar, K. Kumar, and M. Sachdeva, "The use of artificial intelligence based techniques for intrusion detection: A review," *Artificial Intelligence Review*, vol. 34, no. 4, pp. 369–387, 2010. [Online]. Available: http://dx.doi.org/10.1007/s10462-010-9179-5

[12] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Survey*, vol. 41, no. 3, pp. 15:1–15:58, jul 2009. [Online]. Available: http://doi.acm.org/10.1145/1541880.1541882

[13] Z. Jinquan, L. Xiaojie, L. Tao, L. Caiming, P. Lingxi, and S. Feixian, "A self-adaptive negative selection algorithm used for anomaly detection," *Progress in Natural Science*, vol. 19, no. 2, pp. 261–266, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1002007108003407

[14] C. C. Aggarwal, Ed., *Data Streams - Models and Algorithms*, ser. Advances in Database Systems. Springer, 2007, vol. 31. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-47534-9

[15] J. Gama, *Knowledge Discovery from Data Streams*, 1st ed. Chapman & Hall/CRC, 2010.

[16] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and issues in data stream systems," in *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, ser. PODS '02. New York, NY, USA: ACM, 2002, pp. 1–16. [Online]. Available: http://doi.acm.org/10.1145/543613.543615

[17] S. Muthukrishnan, "Data streams: Algorithms and applications," in *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '03. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2003, pp. 413–413. [Online]. Available: http://dl.acm.org/citation.cfm?id=644108.644174

[18] M. Kholghi, H. Hassanzadeh, and M. R. Keyvanpour, "Classification and evaluation of data mining techniques for data stream requirements," in *International Symposium on Computer Communication Control and Automation (3CA)*, vol. 1, may 2010, pp. 474–478.

[19] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "Mining data streams: A review," *SIGMOD Rec.*, vol. 34, no. 2, pp. 18–26, jun 2005. [Online]. Available: http://doi.acm.org/10.1145/1083784.1083789

[20] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '00. New York, NY, USA: ACM, 2000, pp. 71–80. [Online]. Available: http://doi.acm.org/10.1145/347090.347107

[21] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 13–30, March 1963. [Online]. Available: http://www.jstor.org/stable/2282952

[22] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[23] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Survey*, vol. 46, no. 4, pp. 44:1–44:37, mar 2014. [Online]. Available: http://doi.acm.org/10.1145/2523813

[24] A. Bifet and R. Gavaldà, *Adaptive Learning from Evolving Data Streams*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 249–260. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-03915-7_22

[25] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao, "Statistical analysis of honeypot data and building of kyoto 2006+ dataset for nids evaluation," in *Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*, ser. BADGERS'11. New York, NY, USA: ACM, 2011, pp. 29–36. [Online]. Available: http://doi.acm.org/10.1145/1978672.1978676

[26] L. Spitzner, *Honeypots: Tracking Hackers*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[27] A. Bifet, G. Holmes, B. Pfahringer, P. Kranen, H. Kremer, T. Jansen, and T. Seidl, "Moa: Massive online analysis, a framework for stream classification and clustering." in *Discovery Science*, 2010.

[28] J. Gama, R. Sebastio, and P. Rodrigues, "On evaluating stream learning algorithms," *Machine Learning*, vol. 90, no. 3, pp. 317–346, 2013. [Online]. Available: http://dx.doi.org/10.1007/s10994-012-5320-9

[29] R. Kirkby, "Improving hoeffding trees," Ph.D. dissertation, Department of Computer Science, University of Waikato, 2007. [Online]. Available: http://adt.waikato.ac.nz/public/adt-uow20080415.103751/index.html

[30] A. Bifet, G. Holmes, B. Pfahringer, and E. Frank, "Fast perceptron decision tree learning from evolving data streams," in *Proceedings of the 14th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining - Volume Part II*, ser. PAKDD'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 299–310. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-13672-6

[31] J. Gama, R. Sebastio, and P. P. Rodrigues, "Issues in evaluation of stream learning algorithms," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '09. New York, NY, USA: ACM, 2009, pp. 329–338. [Online]. Available: http://doi.acm.org/10.1145/1557019.1557060

[32] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Computing Survey*, vol. 31, no. 3, pp. 264–323, sep 1999. [Online]. Available: http://doi.acm.org/10.1145/331499.331504

[33] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York: Wiley, 2001.

[34] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.