2014 Eighth International Conference on Software Security and Reliability - Companion

Classification of Partially Labeled Malicious Web Traffic in the Presence of Concept Drift

Goce Anastasovski*
Alarm.com
Vienna, VA, USA
Email: ganasta2@mix.wvu.edu

Katerina Goseva-Popstojanova
Lane Department of Computer Science and Electrical Engineering
West Virginia University, Morgantown, WV, USA
Email: Katerina.Goseva@mail.wvu.edu

***Abstract*—Attacks to Web systems have shown an increasing trend in the recent past. A contributing factor to this trend is the deployment of Web 2.0 technologies. While work related to characterization and classification of malicious Web traffic using supervised learning exists, little work has been done using semi-supervised learning with partially labeled data. In this paper an incremental semi-supervised algorithm (CSL-Stream) is used to classify malicious Web traffic to multiple classes, as well as to analyze the concept drift and concept evolution phenomena. The work is based on data collected in duration of nine months by a high-interaction honeypot running Web 2.0 applications. The results showed that on completely labeled data semi-supervised learning performed only slightly worse than the supervised learning algorithm. More importantly, multiclass classification of the partially labeled malicious Web traffic (i.e., 50% or 25% labeled sessions) was almost as good as the classification of completely labeled data.**

***Keywords*-Web 2.0 security; Malicious Web traffic classification; Multiclass classification; Semi-supervised learning; Concept drift; Concept evolution.**

I. Introduction

Since the mid-1990s, Web has had a revolutionary impact on our culture, commerce, and the way we live our lives. When Web was in its infant stage, the content used to be unchangeable and same for all users. In contrast to these Web 1.0 sites where users were limited to the passive viewing of content, today Web 2.0 technologies allow users to interact and collaborate with each other as creators of user-generated content. In addition, Web applications today are primary software solutions of most businesses and individuals. However, Web applications have many hacker-exploitable vulnerabilities that are increasing with the functionality and complexity of Web sites.

The 2012 SANS report [18] ranked the attacks to Web applications among the most frequent type of attacks. The popularity of these applications and their frequent exploitation motivated us to analyze attackers activities on Web systems. Therefore, over a period of several years, our research group developed and deployed several different high-interaction honeypot systems with a three-tier Web architecture (i.e. Web server, application server, and database server) [6], [7]. Since our honeypots had meaningful functionality, attackers were easily attracted, which allowed us to collect several datasets composed of only malicious HTTP traffic.The work presented in this paper is based on a dataset collected in duration of nine months, which consists of only malicious traffic distributed among 19 different vulnerability scan and attack classes.

* This work was done while Goce Anastasovski was affiliated with West Virginia University.

In general, assuming that different malicious activities exhibit different behavioral patterns provides bases for using machine learning methods for their classification. In our previous work [8], [9] we used several batch[1] supervised machine learning algorithms with 10 cross validation to classify malicious traffic and extract rules and patterns that characterize it. However, supervised machine learning algorithms require the data to be completely labeled, which is a tedious, costly and error prone process that in some cases may not even be feasible. Therefore, in this paper we explore whether partially labeled malicious Web traffic can be classified as good as completely labeled data. Furthermore, instead of batch algorithms, we use stream-based algorithm that takes an incremental approach for learning. That is, we divide the data, as it arrives to our honeypot without altering the order, into equally sized windows that are used for incremental learning. Last but not least, we are concerned with the classification of the malicious data when the classes and number of instances in each class change. Specifically, we consider the presence of *concept drift* [10], [21], i.e., changes that occur in the data over time, and *concept evolution* [14], i.e., an appearance of novel classes in the data. The concept drift and concept evolution are important phenomena to address because data are seldom stationary. In the area of cybersecurity, cyber attacks change over time in unforeseen ways which is an example of concept drift, while an emergence of a new, previously unseen attack is an example of concept evolution. (It should be noted that the traditional batch learning with 10 cross validation is unable to deal with concept drift and concept evolution.) In this paper we address the following research questions:

RQ1: Is supervised classification better than semi-supervised classification on completely labeled data?

[1]A batch algorithm stores the whole dataset in the main memory and learns a model from the data, either by cross validation or dividing the data into test and training datasets.

978-1-4799-5843-6/14 $31.00 © 2014 IEEE
DOI 10.1109/SERE-C.2014.31

RQ2: Is there a significant difference in the performance of semi-supervised classification with completely and partially labeled data?

RQ3: How is the classification affected by concept drift and concept evolution?

The main findings of this paper are as follows:

- Supervised learning algorithm J48, which was used as a baseline, was only slightly better than the semi-supervised machine learning algorithm CSL-Stream [15] with completely labeled data.
- CSL-Stream can be used to classify partially labeled data (i.e., 50% and 25%) without significant degradation in accuracy (i.e., at most 8% on average) compared to completely labeled data. This is very important because data labeling is an expensive process, which in some cases may not even be not feasible.
- The classification of bigger classes, which together accounted for 90% of the traffic, depends on the number of instances per class in each window, the distinctive features of each class and the arrival of instances (i.e., concept drift).
- Neither semi-supervised nor supervised learner were able to classify well the very small classes, which together accounted for less then 10% of the malicious traffic.
- The semi-supervised algorithm could not detect the new classes when they first appeared (i.e., concept evolution), but was able to recognize and classify them if they prevailed in future windows.

II. Related Work

We present the related works in two groups based on whether they did or did not consider concept drift. Our focus is on semi-supervised and unsupervised learning approaches used for two-class or multiclass classification of cybersecurity data. Related works that used supervised machine learning methods for intrusion detection or classification of malicious activities can be found, for example, in [9], [11].

We start with the related works which did not consider the presence of concept drift in the data. The first five papers [3], [5], [12], [17], [20] considered the intrusion detection problem, whereas the sixth paper [19] focused on determining the optimal amount of labeled data needed for good classification of normal and malicious executable files. The work in [5] used the DARPA dataset with semi-supervised learning to build an alert filter for intrusion detection. With 10% partially labeled data semi-supervised learning was better in detecting true attacks and reducing false alarms than supervised learning. Two papers [3] and [17] used the KDD 1999 Cup dataset, which is a derivative of the DARPA dataset, with semi-supervised learning to distinguish among normal and malicious classes. A semi-supervised approach to anomaly and misuse detection using partially observable Markov decision process was proposed in [12]. The amount of labeled legitimate user data (0.1%, 1%, and 10%) and attacker data (1%, 10%, 50%) were varied, and high accuracy with the least amount of labeled data was reported. In [20] two supervised and two non-parametric semi-supervised algorithms were trained and tested using 10 cross validation on the Kyoto2006+ dataset. The work in [19] used semi-supervised learning with amounts of labeled data from 10% to 90%. In terms of accuracy, the best results were achieved with 65% labeled data.

Next, we address the related works that considered concept drift in the data [1], [2], [4], [14], [15], [16]. All six papers used the KDD 1999 Cup dataset to perform semi-supervised [14], [15] or unsupervised classification [1], [2], [4], [16]. Two semi-supervised algorithms with a window size of 1,000 instances on completely and partially labeled (i.e., 50%, 25%, 10%) data were used in [15]. Another work was concerned with the detection of novelty (i.e., concept evolution) and used three metrics to evaluate the performance of the semi-supervised learners [14]. Three papers [2], [4] and [16] clustered the KDD Cup 1999 data into one normal and four malicious clusters. In [16] the data were divided into nine 16 MByte windows. A window size of 1,000 instances was used in [2] and [4]. Similarly, in [1] the data were divided into windows of size 2,000 instances.

Our work differs from the related works in several ways. First, none of the related works was based on only malicious data collected by advertised, fully functional, three-tier honeypot system, with standard security settings. Most of the papers [1], [2], [3], [4], [5], [14], [15], [16], [17] used the outdated DARPA and KDD 1999 datasets, which in addition to the normal traffic include four classes of simulated attacks.

Second, while we use semi-supervised learning as several papers which dealt with intrusion detection, we only use malicious traffic and classify attacker activities with a goal to distinguish among ten vulnerability scan classes and nine attack classes. This classification is important because attacks are much more critical events than vulnerability scans. Other works distinguished among one normal and four malicious classes [1], [2], [3], [4], [5], [14], [15], [16], [17], or among one normal and one malicious class [12], [20], [19].

Third, none of the related works which considered concept drift assessed the classification of individual classes. Accuracy, which is a misleading metric for imbalanced datasets, was used in [4] and [15]. Similarly, SSQ was used in [1], [4] and [16], while purity was used in [2]. Three metrics that measured the overall classification were used in [14].

Finally, we study both concept drift and concept evolution whereas some of the related work papers did not study either of them [3], [5], [12], [17], [19], [20], or studied only concept drift [1], [2], [4], [15], [16] or only concept evolution [14].

III. Data Collection and Extraction

The malicious Web traffic data used in this paper was collected by a high-interaction honeypot [7], which ran off-the shelf operating system and applications, which followed typical security guidelines and did not include user accounts with nil or weak passwords. Furthermore, instead of a set of independent applications typically installed on honeypots, the honeypot had meaningful functionality and followed a three-tier architecture consisting of a front-end Web server (i.e., Microsoft IIS 5.1), application server (i.e., PHP 5.0.2 server), and a back-end database (i.e., MySQL database version 4.1) and included two Web 2.0 applications (i.e., MediaWiki version 1.9.0 and Wordpress version 2.1.1).

The honeypot was advertised using a technique called 'transparent linking', which involved placing hyperlinks pointing to the honeypot on public Web pages, so that it was indexed by search engines and Web crawlers, but could not be accessed directly by humans. Therefore, after removing the legitimate crawlers, such as Google and MSNbot, the collected traffic consisted of only malicious Web sessions. From the honeypot with this configuration, the data was collected in duration of nine months (i.e., 273 days) and consisted of 5,902 sessions. Our research group has labeled these malicious Web sessions using a semi-automated process [7], which resulted in 19 different classes of malicious activities (i.e, ten vulnerability scan and nine attack classes) shown in Table I. Overall, out of 19 classes only five have high frequency. The remaining fourteen classes account for a little over 9% collectively, with most of them contributing individually less than 1%. These labels were used as a ground truth in evaluation of the performance of multiclass classification presented in this paper.

Table I: Breakdown of malicious Web sessions

Malicious activity	sessions	%
Vulnerability scans: Total	**2,833**	**48.84%**
DFind	44	0.74%
Other Fingerprinting	2	0.03%
Static	508	8.60%
Blog	797	13.50%
Wiki	1,307	22.14%
Blog & Wiki	150	2.54%
Static & Blog	11	0.18%
Static & Wiki	22	0.37%
Static & Blog & Wiki	28	0.47%
Other Vulnerability Scan	14	0.23%
Attacks: Total	**3,019**	**51.15%**
Denial of Service (DoS)	4	0.06%
Password cracking Blog	10	0.16%
Password cracking Wiki	71	1.20%
Spam on Blog	1,434	24.29%
Spam on Wiki	1,304	22.09%
Remote File Inclusion (RFI)	9	0.15%
SQL injection	2	0.03%
Cross-site Scripting (XSS)	13	0.22%
Other Attacks	172	2.91%
Total	**5,902**	**100%**

Each malicious Web session was characterized with a vector of 43 different features (i.e., session characteristics), inspired by feature sets used in articles on network intrusion detection, host intrusion detection, and Web crawlers identification. The complete list of 43 features is as follows: (1) number of requests; (2) bytes transferred; (3) duration (in seconds); (4)-(8) mean, median, minimum, maximum, and standard deviation of the time between requests; (9)-(14) number of requests with a specific method type (i.e., GET, POST, OPTIONS, HEAD, PROPFIND, and other); (15) number of requests to picture files (e.g., .jpeg, .gif, .ico, .png); (16) number of requests to video files (e.g., .avi, .wmv); (17) number of requests to static application files (e.g., .html, .htm); (18) number of requests to dynamic application files (e.g., .php, .asp); (19) number of requests to text files (e.g., .txt, .css); (20)-(24) number of requests with specific status code (i.e., Informational (1xx), Success (2xx), Redirect (3xx), Client error (4xx), and Server error(5xx)); (25)-(29) mean, median, minimum, maximum, and standard deviation of the length of requests' substrings within a session; (30)-(34) mean, median, minimum, maximum, and standard deviation of the number of parameters passed to application within a session; boolean indications of whether: (35) robots.txt file was accessed in any request of that session; (36) it was a night session (from 12 am to 8 am local time); (37) there was a remote site injection in at least one request; (38) a semicolon was used to divide multiple attributes passed to an application in at least one request; (39) a string containing suspicious encoding in any of the requests; (40) a reserved character was used in any of the requests; (41) an ASCII control character was used in any of the requests; (42) a non-ASCII control character was used in any of the requests; and (43) an invalid encoding was used in any of the requests.

IV. Data Mining Approach

In this paper we use the semi-supervised learning algorithm CSL-Stream, which is an incremental algorithm that is able to process data streams [15]. Because the window size is an adjustable parameter that affects the classification, we experimented with two different window sizes: 1,000 and 500 sessions (i.e., instances). First, the data were divided into six equally sized windows, five windows with exactly 1,000 instances, and the sixth window with 902 instances. Then, we divided the data into twelve equally sized windows, each window with exactly 500 instances and the twelfth window with 402 instances. The data in each odd window were used for training the learner, while the data in each even window were used for testing the learner. For example, the data in the first window (Window 1 Train) were used for training the learner and the data in the second window (Window 1 Test) were used for testing the learner. Note that the order of arrival of the data was not altered in any way, i.e., the first 1,000 (or 500 instances depending on the window size) were included in the first window and the second 1,000 (or 500) instances were included in the second window, and so on. CSL-Stream has seven other parameters which were set to the values recommended in [15].

The goal of RQ1 is to establish a baseline. Because supervised learning algorithms work only with completely labeled data, we compared the performance of the semi-supervised algorithm CSL-Stream with completely labeled data to the performance of the supervised learning algorithm J48, which had the best performance of several supervised algorithms we used in our previous work [9]. For a fair comparison, the same training and testing process was used for the supervised algorithm J48, that is, it was trained on odd windows and tested on even windows.

Because labeling malicious traffic is a tedious and expensive process, RQ2 is concerned with exploring the performance of the semi-supervised learning for different amounts of partially labeled data (i.e., 50% and 25%). We also ran experiments with only 10% labeled data, but the results were not satisfactory and therefore they are not included in this paper.

To evaluate learners' performance we used (1) accuracy as a metric that tell us how good is the overall classification (i.e., classification for all classes together) and (2) per class metrics that tell us how good the individual classes were classified. To compute these metrics we first computed the confusion matrix E with mxm elements, where m is the number of classes (i.e., 19). For each class i out of the total m classes, the diagonal element E_{ii} is equal to the number of sessions that actually belong to i-th class and are correctly assigned to this class, which is referred to as true positives TP_i. Each off-diagonal element $E_{ij}, i \neq j$ gives the number of sessions that actually belong to the i-th class that have been incorrectly assigned to class j. For each individual class i let FP_i denote false positives (the number of the sessions that do not belong to i-th class, but are erroneously assigned to this class); FN_i denote false negatives (the number of the sessions that belong to i-th class, but are falsely not assigned to this class); and TN_i denote true negatives (the number of the sessions that do not belong to i-th class and correctly are not assigned to this class).

A metrics for all classes. We use the *overall accuracy*, defined by (1), to evaluate the learners' performance for all m classes together. Because the overall accuracy gives equal weight to each malicious session it tends to be dominated by the classifier's performance on larger classes.

$$\text{accuracy} = \text{TP}/(\text{TP}+\text{FN}) = \sum_{i=1}^{m} \text{TP}_i / \sum_{i=1}^{m} (\text{TP}_i + \text{FN}_i) \quad (1)$$

Per class metrics. For each class i we compute the recall (R_i) and precision (P_i) given by:

$$R_i = \text{TP}_i/(\text{TP}_i + \text{FN}_i) \quad (2)$$

$$P_i = \text{TP}_i/(\text{TP}_i + \text{FP}_i) \quad (3)$$

Recall (R_i), defined by (2), which is often called *probability of detection*, represents the probability of correctly classifying the sessions of each class (i.e., the ratio of detected sessions of class i to all sessions in class i). *Precision* (P_i), given by (3), determines the fraction of sessions correctly classified to belong to class i out of all sessions classified as class i. The *F_i-Score* of each class i is then computed as a harmonic mean of the R_i and P_i as:

$$F_i = 2R_iP_i/(R_i + P_i) \quad (4)$$

R_i, P_i, and F_i have values in the interval $[0, 1]$. Larger values indicate better classification. The ideal classification have both recall and precision values of 1, which results to F-score equal to 1. By definition, F_i-score is 0 if either the recall or the precision is 0.

V. Main Observations

A. Comparison based on the overall accuracy

In this subsection, we use the overall accuracy given with (1) as an evaluation criterion to address RQ1 and RQ2. For this purpose, we use Figure 1 and Figure 2 which present the accuracy for CSL-Stream and J48, in each pair of train and test windows, with completely and partially labeled data for window sizes of 1,000 and 500 instances, respectively.

First, we focus on RQ1, that is, explore whether supervised classification performs better than semi-supervised classification on completely labeled data. As shown in Figure 1, when the window size was set to 1,000 instances and the data were completely labeled the accuracy of J48 was on average 13% higher than the accuracy of CSL-Stream with completely labeled data. With a window size of 500 instances and completely labeled data (see Figure 2), the classification results were improved for both CSL-Stream and J48. In this case, the accuracy of J48 was on average 10% higher than the accuracy of CSL-Stream. In other words, **in terms of accuracy, J48 classified the completely labeled data slightly better than CSL-Stream.**

Next, we focus on RQ2, that is, explore whether there is a significant difference of the classification performance with completely and partially labeled data. As it can be seen in Figures 1 and 2 when the data were partially labeled, there was a small drop in the classification accuracy compared to when the data were completely labeled. In particular, for window size of 1,000 instances, the accuracy (averaged across the three pairs of windows) with completely, 50% and 25% labeled data was 73%, 70% and 67%, respectively. When the size of the window was reduced (i.e., set to 500 instances), the average accuracy when the data were completely, 50% and 25% labeled improved to 84%, 82% and 76%, respectively.

These results indicate that CSL-Stream is able to achieve good accuracy with partially labeled data. When the data were 50% labeled the drop of the average accuracy was not worse than 3% and, even more, when only 25% of

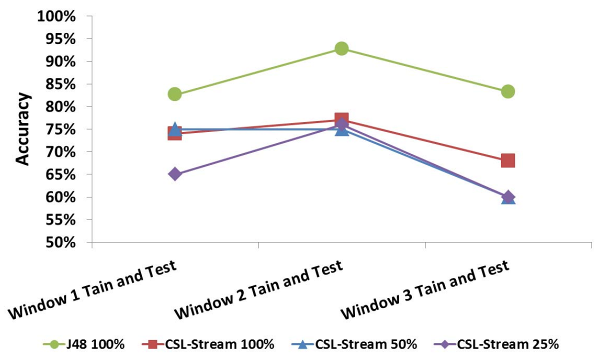

Figure 1: Accuracy of J48 and CSL-Stream for window size of 1,000 instances.

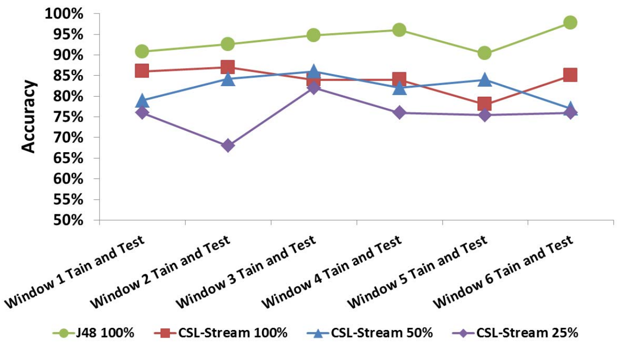

Figure 2: Accuracy of J48 and CSL-Stream for window size of 500 instances.

data were labeled the average accuracy dropped not more than 8% compared to the case when the malicious traffic was completely labeled. This is an important result because completely labeling the data is very expensive, time consuming, error prone and in some occasions impossible. With respect to RQ2 we conclude that **semi-supervised machine learning algorithms, such as CSL-Stream, can be used to classify partially labeled data without significant degradation in accuracy.**

B. Comparison based on the per class metrics, i.e., F_i-Scores

In this subsection, we revisit RQ1 and RQ2 using the per class metrics as evaluation criteria, first for a window size of 1,000 instances and then for a window size of 500 instances.

1) Per class performance with a window of 1,000 instances: Figure 3 presents the F_i-Scores averaged over the three pairs of windows for each of the 19 malicious classes with 100%, 50% and 25% labeled data. With respect to RQ1, it is clear from Figure 3 that, **based on the average F_i-Scores over the three pairs of windows, J48 classified the completely labeled data slightly better than CSL-Stream;** the lowest difference in the average F_i-Scores was 6% for the *Blog & Wiki* class, and the highest difference was 28% for the *Static* class. Furthermore, when J48 was used 11 out of 19 classes had average F_i-Scores greater than 0%; in case of CSL-Stream 10 out of 19 classes had average F_i-Scores greater than 0%. If we consider average F_i-Score greater than 50% to be a good classification, 9 classes had F_i-Score greater than 50% in case of J48, while only 6 classes had average F_i-Score greater than 50% in case of CSL-Stream.

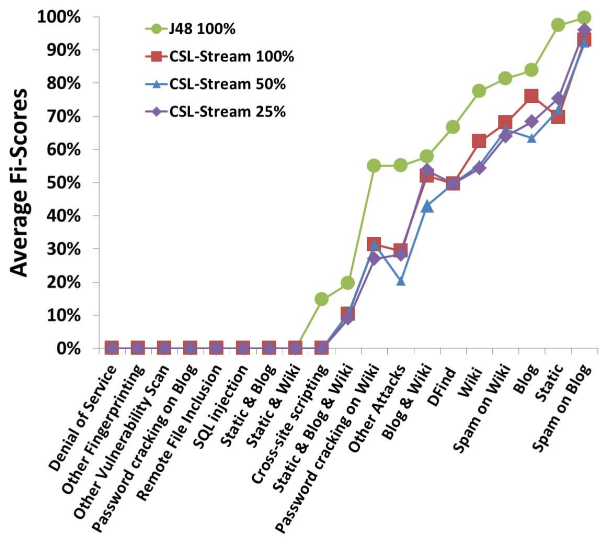

Figure 3: F_i-Scores averaged over the three pairs of windows for each of the 19 malicious classes with 100%, 50% and 25% labeled data. In each window there were 1,000 instances. The classes are ordered from the lowest to the highest values of the average F_i-Scores produced by J48.

The best classified classes, shown on the right side of the x-axis in Figure 3, had many instances. Each of the five largest classes – *Spam on Blog*, *Spam on Wiki*, *Wiki*, *Blog* and *Static* – had more than 175 instances per window, while the classes *Static & Blog & Wiki*, *Password cracking on Wiki*, *Other Attacks*, *Blog & Wiki*, and *DFind* each had less than 175 instances per window. It is easily seen that the classes with more than 175 per window instances were better classified than classes with less than 175 instances per window. This shows that **the classification was highly dependent on the number of class instances present in the training and testing windows.**

Nine classes with CSL-Stream and eight classes with J48, that appear on the left side of the x-axis in Figure 3 were not classified at all, i.e., the average F_i-Scores for these classes were 0%. These classes were misclassified because they were very small, together contributing only 1.4% of all data. The fact that these small numbers of instances were divided across different windows made the classification impossible, even with completely labeled data and using supervised learning.

To address RQ2 using per class metrics as evaluation criterion, we took the F_i-scores averaged across the three pairs of windows between completely and partially labeled data and subtracted them from each other, which resulted in a set of values that represent the difference in classification

Table II: Mean and standard deviation of the difference in F_i-Scores, averaged across the three pairs of windows, between different percentages of labeled data (i.e., 100%, 50%, and 25%)

Amount of labeled data	100% and 50%	100% and 25%	50% and 25%
Mean difference	2%	1%	-1%
Standard deviation	4%	3%	3%

with completely and partially labeled data. From Table II, which presents the average and standard deviation of the values in this set, it is obvious that the mean difference in F_i-scores, averaged across the three pairs of windows, between completely and partially labeled data did not exceed 2%, with a standard deviation of at most 4%. We conclude that, **based on per class F_i-Scores, CSL-Stream classified the partially labeled data (i.e., 25% and 50%) as good as the completely labeled data.** As we noted previously, this is an important observation because labeling the data completely is a very expensive, time consuming, error prone and in some occasions unfeasible process.

2) Per class performance with a window of 500 instances: The F_i-Scores for window size of 500 instances, averaged over the six pairs of windows, for 100%, 50% and 25% labeled data are shown in Figure 4. With respect to RQ1 **for window size of 500 instances, similarly as when the window size was 1,000 instances, J48 classified the completely labeled data better than CSL-Stream.** In case of J48, 11 out of 19 classes had average F_i-Scores greater than 0%, while when CSL-Stream was used 10 out of 19 classes had average F_i-Scores greater than 0%. J48 resulted in 8 classes with F_i-Scores over 50%, while when CSL-Stream was used on completely labeled data 7 classes had F_i-Scores greater than 50%. The lowest difference in average F_i-Scores between J48 and CSL-Stream was -6% for the *Spam on Wiki* class, and the highest difference was 45% for the *Other attacks* class.

Again, the classes on the right side of the x-axis in Figure 4 (i.e., *Spam on Wiki*, *Wiki*, *Blog*, *Spam on Blog*, *Static*) were classified well because they accounted for 90% of the malicious traffic. Nine classes with CSL-Stream and eight classes with J48, which appear on the left side of the x-axis on Figure 4 had F_i-Scores equal to 0%. Similarly as when the window size was 1,000 instances, these classes were not classified at all because they were very small, together contributing only 1.4% of all data, across all windows.

To address RQ2 related to the performance of semi-supervised learning on partially labeled data, using per class metrics as evaluation criterion we observe, similarly as in case of window size of 1,000 instances, that **CSL-Stream is capable of classifying 50% or even 25% labeled data without significant decrease of F_i-Scores compared to when the data were completely labeled.**

3) Compassion of the classification with a window of size 500 to the classification with a window of size 1,000 instances: As discussed in section V-A, window size of 500 instances led to higher overall accuracy than window size of 1,000 instances. Here we compare the performance with respect to per class metrics (i.e., F_i-Scores). The smaller window improved the average F_i-Scores for six classes, while it worsened the average F_i-Scores for four classes. The average F_i-Score improved by at most 23% for the *Password cracking on Wiki* class with completely labeled data and it worsened by at most 30% for the *Blog & Wiki* class with 25% labeled data. The classification for the nine very small classes that had zero F_i-Scores neither improved nor worsened with the decrease of the window size.

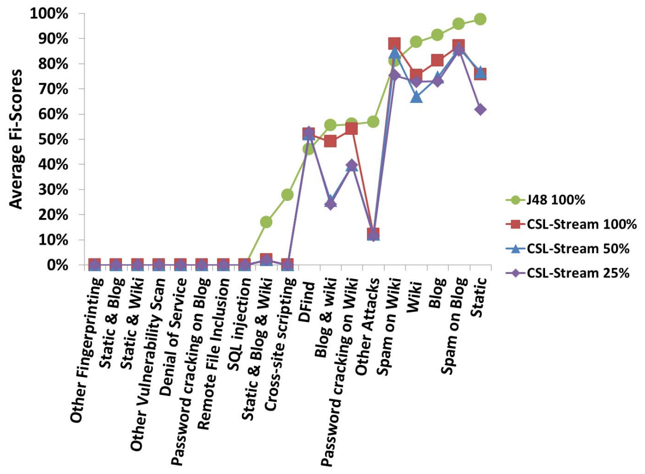

Figure 4: F_i-Scores averaged, for each of the 19 malicious classes, over the six pairs of windows with 100%, 50% and 25% labeled data. In each window there were 500 instances. The classes are ordered from the lowest to the highest values of the average F_i-Scores produced by J48.

Table III presents the percentages of increase and decrease in average per class F_i-Scores, averaged across all pairs of windows, when the window size was decreased from 1,000 to 500 instances with 100%, 50% and 25% labeled data. We start the discussion with the six classes whose average classification was improved. Two of these classes *DFind* and *Password cracking on Wiki* were small and contributed 0.7% and 1.2% of the whole dataset, respectively. The improvement in the case of *DFind* class was due to the better classification in windows 4 and 6. Similarly, the improvement in the average F_i-Scores of the *Password cracking on Wiki* class was due to the better classification in windows 3, 4, 5, and 6. The remaining four classes whose classification was improved (i.e., *Wiki*, *Static*, *Blog*, and *Spam on Wiki*) were bigger. Their F_i-Scores improved because of the extra training and testing windows with varying, but sufficient, number of instances. Thus, CSL-Stream tuned its model to favor instances of these four classes for classification. The biggest increase of the average F_i-Scores, over the 6 pairs of windows, among these four big classes was by 20% for the *Spam on Wiki* class with completely labeled data. (The -14% value in case of *Static* class with 25% labeled data indicates

Table III: Changes of the per class F_i-Scores, averaged across all pairs of windows, when the window size was decreased from 1,000 to 500 instances with 100%, 50% and 25% labeled data. Positive (negative) sign annotates better (worse) performance in case of 500 instances window size.

Class	100 % labeled	50% labeled	25% labeled
DFind	2%	2%	3%
Password cracking on Wiki	23%	8%	13%
Wiki	13%	12%	9%
Static	6%	5%	-14%
Blog	5%	11%	5%
Spam on Wiki	20%	19%	11%
Static & Blog & Wiki	-8%	-9%	-7%
Other Attacks	-17%	-8%	-17%
Blog & Wiki	-3%	-18%	-30%
Spam on Blog	-6%	-6%	-11%

that the classification was worse for the window size of 500 instances than for the window size of 1,000 instances.)

Three of the four classes for which the classification worsened (i.e., *Static & Blog & Wiki*, *Other Attacks*, *Blog & Wiki*) when the window size was decreased from 1,000 to 500 instances were small classes. This happened because with smaller window the number of training examples of these classes in each window was reduced significantly, which led to lower F_i-Scores. The average F_i-Score for the big class (i.e., *Spam on Blog*) worsened because of the smaller number of training instances in the first two windows.

Even though the classification with smaller window improved for some and worsened for other classes, as discussed in section V-A, the overall accuracy of the classification improved. This can be explained by the fact that smaller window has a potential to create more accurate model when there is a big concept drift in the data, while a larger window is beneficial when the data generation process is stable (i.e., the distribution of classes does not change much) [13], [22]. The concept drift phenomenon is explored in details in the following section.

C. Concept drift

This section is focused on RQ3, that is, it explores how the multiclass classification of malicious traffic is affected by the concept drift and concept evolution, that is, when the classes and number of instances in each class change. We specifically analyze several factors that affect the performance of the classification: (1) number of instances in each class, (2) set of features (i.e., session characteristics) used to describe the data, and (3) the existence of concept drift and concept evolution. Due to space limitation we only present the results for the window size of 1,000 instances.

We already concluded that the classification of individual classes is highly dependent on the number of training and testing instances present in a class. Thus, the classes that have many instances are among the best classified (and are shown on the right side of the x-axis in Figure 3). As shown in Table I the *Spam on Wiki* and *Wiki* classes had 1,304 and 1,307 sessions (i.e., instances), respectively, which made them bigger than the *Static* and *Blog* classes, which had 508 and 797 sessions, respectively. However, even though the *Static* and *Blog* classes were smaller than the *Spam on Wiki* and *Wiki* classes, they were classified better by the CSL-Stream algorithm. It follows that, in addition to the number of instances in each class, the classification was dependent on other factors too. In this section we explore these factors.

Smaller classes could have been classified better than larger classes either because (1) *Static* and *Blog* classes had better predictive features than *Spam on Wiki* and *Wiki* classes or because (2) the distribution of the *Static* and *Blog* classes in each training and testing window was more stable (i.e., the concept drift was smaller) than the distribution of the *Spam on Wiki* and *Wiki* classes. From our previous work [9] we know that these classes share the same most predictive features, which means that the smaller classes were classified better than the bigger classes due to smaller concept drift.

In order to explore how the concept drift affected the classification of individual classes we present in Figure 5 the distribution of arriving instances in each training and testing window, for each class. Based on this figure we identified the following four scenarios of concept drift.

- Classes that appeared in all windows with different degree of concept drift: *Spam on Blog*, *Blog*, *Spam on Wiki*, *Wiki*, *Static*, *Blog & Wiki*, and *Cross-site scripting*.
- Classes that existed in the past, seized to exist and appeared again in the future: *DFind*, *Static & Blog*, *Static & Wiki*, *Static & Blog & Wiki*, *Remote File Inclusion* and *Password cracking on Blog*.
- Classes that did not exist in the past and then started to exist in the future (i.e., classes that exhibited concept evolution rather than concept drift): *Other Attacks*, *Password cracking on Wiki*, *Other Fingerprinting*, and *Other Vulnerability Scan*.
- Classes that existed in the past and then seized to exist in the future: *Denial of Service* and *SQL Injection*.

The classes from the first scenario were among the best classified classes with F_i-Scores from 50% to 99%. The only class that was not classified well was the *Cross-site scripting*, which was due to the fact that this class was very small, contributing only 0.2% of the whole dataset distributed across the three pairs of training and testing windows. A closer look at the distribution of instances in Figure 5 and the classification for individual classes in Figure 3 showed that smaller classes with a smaller concept drift (i.e., *Blog* and *Static*) were classified better than larger classes with bigger concept drift (i.e., *Spam on Wiki* and *Wiki*). **Therefore, we can conclude that the classification of individual classes was dependent on the number of instances in each window and the amount of concept drift.**

The second scenario, where the number of instances in a class decreased up to a point where there were no instances present in some windows, and then the instances from that

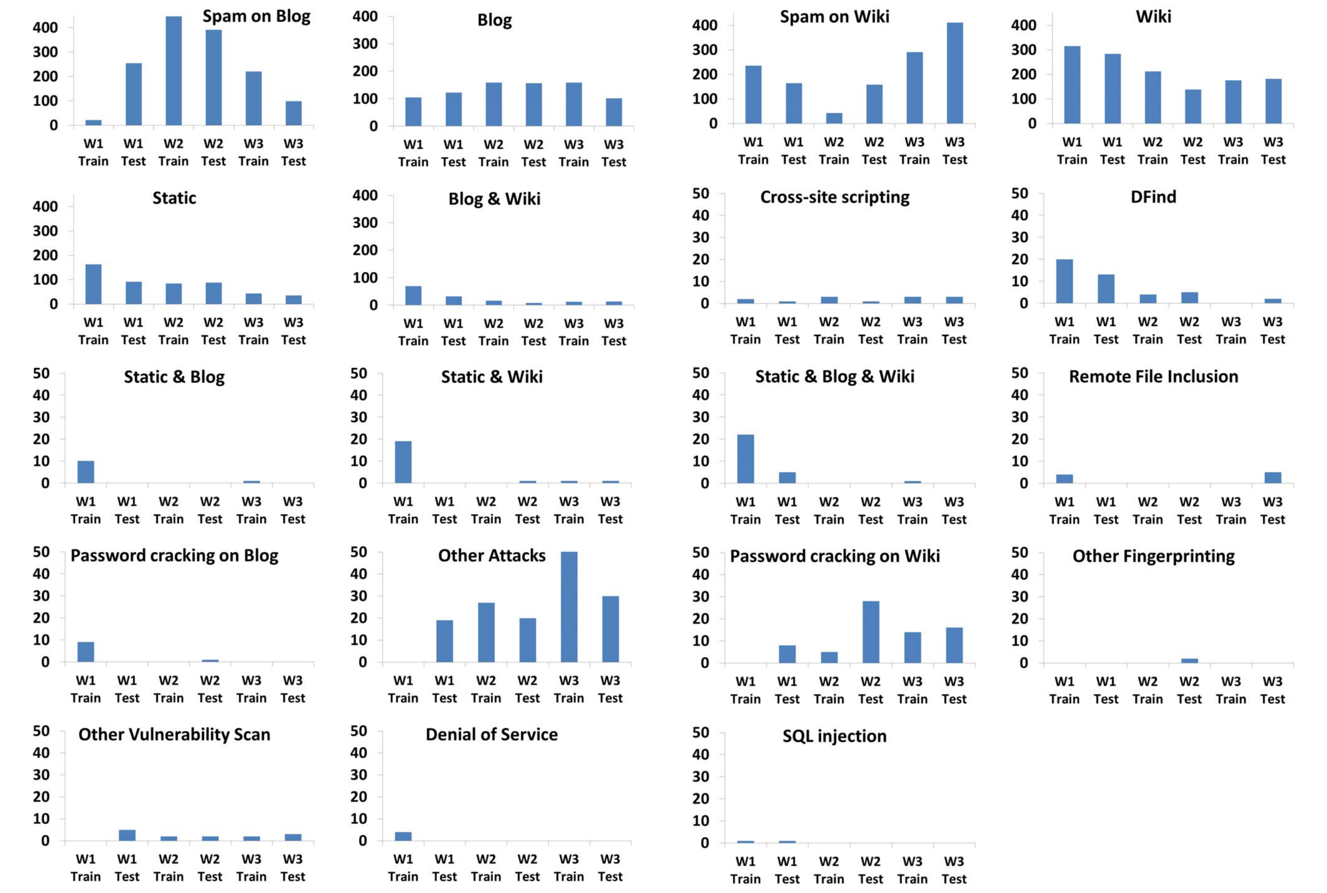

Figure 5: Distribution of instances in each training and testing window by class, with 1,000 instances in each window. The y-axes present the number of instances. Note that there are two different ranges on y-axes: at most 50 and at most 400 instances.

class started appearing again, consisted of six classes. *DFind* was the only class which for some pairs of windows was classified well. Specifically, in the first pair of windows (Window 1 Train and Window 1 Test), the F_i-Score for this class was 90% with 100%, 50% and 25% labeled data, even though the number of *DFind* class instances in Window 1 Train and Window 1 Test were 20 and 13, respectively, out of total 1,000 instances in each window. The *DFind* class was classified well in spite of the small number of training and testing instances (as small as 1% to 2%) because of the characteristics of its features. Thus, the majority of sessions in this class had requests with the same length of the request substrings (i.e., 29), the same number of bytes transferred (i.e., 4,167) and zero number of parameters. As the number of training and testing instances dropped in the future pairs of windows, so did the performance of the classification of this class. Specifically, in the second and third pairs of training and testing windows there were 4 and 0 training instances, and 5 and 2 testing instances, respectively. This resulted in F_i-Scores of 59% and 0%, respectively. Therefore, even thought the classification of *DFind* class in the first pair of windows was very good, the average F_i-Score over the three pairs of windows was low (i.e., 50%). The remaining five classes from the second scenario (i.e., *Static & Blog*, *Static & Wiki*, *Static & Blog & Wiki*, *Remote File Inclusion*, and *Password cracking on Blog*) were very small, together contributing to only 1.5% of the whole dataset, which was not sufficient for the semi-supervised learning algorithm to learn from.

We observed four classes that belonged to the third scenario, which is characterized by concept evolution or discovery of novelty, that is, classes that did not exist in the past and then started to appear in the future. This is an every day scenario in real world Web servers, which are attacked by new types of attacks on regular basis. Two classes (*Other Attacks* and *Password cracking on Wiki*) in the third scenario had average F_i-Scores greater than 0%. When the learner was trained on Window 1 Train and tested on Window 1 Test, the F_i-Scores for these two classes were 0%. This happened because, for both classes, there were no instances present to train the learner in Window 1 Train, but there were instances present to test the learner in Window 1 Test. Since the learner has not been trained to recognize these classes, it missclassified them. However, as the number of training and testing instances increased, so did the F_i-Scores, especially for the *Password cracking on Wiki* class. It appears that **CSL-Stream could not detect the novelty when it first appeared, but it was able to recognize and classify the novelty if it prevailed in future windows.** The other two classes (*Other Fingerprinting* and *Other Vulnerability Scan*) in the third scenario had only a handful of instances (i.e., together only 0.2% of the whole dataset), which were not sufficient to learn, thus resulting in a poor classification.

The classes from the fourth scenario, *Denial of Service* and *SQL Injection*, had in total four and two instances, respectively. As expected, for both classes F_i-Score was 0%.

In summary, the classification of large classes (i.e., classes that together accounting for 90% of the malicious traffic) depends on the number of instances a specific class has in each window, how distinctive are its features, and the pattern of arrivals (i.e., concept drift). For very small classes (each around or less than 1% of the whole dataset, distributed across multiple windows) we were not able to test the factors that influence the classification because these classes were too small to be recognized by the machine learning algorithms that use incremental learning, both semi-supervised and supervised.

VI. Discussion

In this section we compare the results of using incremental semi-supervised learning presented in this paper to the results of our previous work [9] where batched supervised classification with 10 cross-validation was used. In our previous work only 3 small classes (i.e., *SQL injection*, *Other Fingerprining*, and *Blog & Wiki*) were not classified well (with F_i-Scores below 50%) because they were too small (i.e., together had 15 out of 5,902 instances). The other classes were distributed among three classification groups with F_i-Scores from 50% to 60%, from 60% to 80%, and from 80% to 100%. Interestingly, in addition to large classes, several small classes (i.e., *Denial of Service*, *Password cracking on Wiki* and *DFind*) were among the best classified classes. These small classes were classified as good as the large classes because batch learning with 10 cross validation was used [9]. In our previous work [9] we extracted rules that could be used in firewalls and intrusion detection systems to identify malicious classes, including some small classes. However, batch learning with 10 cross validation is not always possible or desirable.

In this paper, we classified the malicious traffic as it arrives, using incremental semi-supervised learning. Even though the overall accuracy was very good (i.e., above 90% in each pair of train and test windows), some of the small individual classes were not classified well neither with supervised nor with semi-supervised classification. On the other side, larger classes were classified well, with supervised learning performing slightly better than semi-supervised learning but with a higher price to be payed because it requires completely labeled data. The fact that the semi-supervised learning was able to classify the partially labeled datasets (with as little as 25% of malicious traffic being labeled) almost as good as the completely labeled dataset is very important because labeling is an expensive and error prone process, which in some cases is not feasible. Another important characteristic of this paper is the fact that we studied the effect of concept drift and concept evolution on the classification of malicious traffic. It should be noted that we did not alter the order or number of instances in our datasets, neither we removed classes no matter how small they were. This allowed us to have a realistic representation of malicious traffic that typically contains new types of attacks (i.e., concept evolution), as well as attacks that occur from time to time (i.e., concept drift) or attacks that persist over time.

VII. Concluding Remarks

Motivated by the fact that data labeling is a very tedious, expensive and error prone process, in this paper we used an incremental semi-supervised machine learning algorithm which is able to process partially labeled data streams and account for concept drift in the data. The results are based on a dataset collected from a high-interaction honeypot that had a three tier architecture and ran Web 2.0 applications. The collected malicious traffic consisted of close to six thousand sessions, which were divided among ten vulnerability scan classes and nine attack classes. Our findings, as they pertain to the research questions, are summarized as follows.

RQ1: Is supervised classification better than semi-supervised classification on completely labeled data?

- The supervised algorithm (i.e., J48) performed slightly better than the semi-supervised CSL-Stream algorithm with completely labeled data.

RQ2: Is there a significant difference in the performance of semi-supervised classification with completely and partially labeled data?

- In terms of overall accuracy, the semi-supervised algorithm CSL-Stream classified the partially labeled data (i.e., 50% and 25%) almost as good as completely labeled data. The biggest drop in the average difference in accuracy was 8%.
- In terms of the per class metrics (i.e., F_i-Scores) the classification with partially labeled data was almost as good as the classification with completely labeled data. This is very important observation since labeling the data is a tedious and expensive process.
- Regardless of the window size (i.e., 1,000 or 500 instances), the difference in classification with completely and partially labeled data was not significant. Smaller window size resulted in better classification for some and worse classification for other classes, but the overall accuracy of the classification improved for completely as well as for partially labeled data.

RQ3: How is the classification affected by concept drift and concept evolution?

- The classification of large classes (which together accounted for 90% of the malicious traffic) depended on the number of instances per class in each window, how distinctive were the features that characterized that class, and the arrival pattern of the class instances (i.e., the existence and the extent of the concept drift).
- The classification of very small classes (each with around or less than 1% of the total traffic) was not good in general, regardless of the window size, learner used (i.e., J48 or CSL-Stream), amount of labeled data or the presence of concept drift or concept evolution.
- CSL-Stream could not detect the appearance of novelty when it appeared for the first time, but was able to recognize and classify the novelty (i.e., concept evolution) if it prevailed in future windows.

This paper showed that semi-supervised learning can classify partially labeled malicious traffic almost as good as completely labeled traffic. It also explored the effect of concept drift and concept evolution on the classification, which is very important because these are common phenomena observed in cybersecurity. We note that, as in any work using machine learning, the results are specific to the used dataset and algorithms. The generalizability of the findings using other datasets and other semi-supervised algorithms is a focus of our future work. We also point out that, when incremental learning is used, neither supervised nor semi-supervised algorithms can classify successfully malicious behaviors consisting of a very small number of instances. Future research efforts of the cybersecurity research community should be focused on solving the challenge of successfully classifying very small classes when using incremental learning.

Acknowledgments

This work was funded in part by the WVU ADVANCE Sponsorship Program funded by the NSF ADVANCE IT Program award HRD-100797. The authors thank J. Alex Baker, David Krovich, Jonathan Lynch, Brandon Miller, and Risto Pantev for their support with the experimental setup and data collection, and Hai-Long Nguyen for sharing his implementation of the CSL-Stream algorithm.

References

[1] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *29th Int. Conf. Very Large Data Bases*, vol. 29, 2003, pp. 81–92.

[2] F. Cao, M. Ester, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *2006 SIAM Int. Conf. Data Mining*, 2006, pp. 328–339.

[3] C. Chen, Y. Gong, and Y. Tian, "Semi-supervised learning methods for network intrusion detection," in *IEEE Int. Conf. Systems, Man and Cybernetics (SMC)*, 2008, pp. 2603–2608.

[4] Y. Chen and L. Tu, "Density-based clustering for real-time stream data," in *13th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2007, pp. 133–142.

[5] C.-Y. Chiu, Y.-J. Lee, C.-C. Chang, W.-Y. Luo, and H.-C. Huang, "Semi-supervised learning for false alarm reduction," in *Advances in Data Mining. Applications and Theoretical Aspects*, 2010, pp. 595–605.

[6] K. Goseva-Popstojanova, B. Miller, R. Pantev, and A. Dimitrijevikj, "Empirical analysis of attackers activity on multi-tier Web systems," in *24th IEEE Int. Conf. Advanced Information Networking and Applications (AINA)*, 2010, pp. 781–788.

[7] K. Goseva-Popstojanova, R. Pantev, A. Dimitrijevikj, and B. Miller, "Quantification of attackers activities on servers running Web 2.0 applications," in *9th IEEE Int. Symp. Network Computing and Applications (NCA)*, 2010, pp. 108–116.

[8] K. Goseva-Popstojanova, G. Anastasovski, and R. Pantev, "Classification of malicious Web sessions," in *21st Int. Conf. Comp. Communications & Networks (ICCCN)*, 2012, pp. 1–9.

[9] ——, "Using multiclass machine learning methods to classify malicious behaviors aimed at Web systems," in *23rd IEEE Int. Symp. Software Reliability Eng. (ISSRE)*, 2012, pp. 81–90.

[10] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *7th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*. 2001, pp. 97–106.

[11] K. Julisch, "Data mining for intrusion detection," *Applications of Data Mining in Computer Security*, pp. 33–58, 2002.

[12] T. Lane, "A decision-theoritic, semi-supervised model for intrusion detection," in *Machine Learning and Data Mining for Computer Security*. 2006, pp. 157–177.

[13] M. Last, "Online classification of nonstationary data streams," *Intelligent Data Analysis*, vol. 6, no. 2, pp. 129–147, 2002.

[14] M. M. Masud, J. Gao, L. Khan, J. Han, and B. Thuraisingham, "Classification and novel class detection in data streams with active mining," in *Advances in Knowledge Discovery and Data Mining*. 2010, pp. 311–324.

[15] H.-L. Nguyen, W.-K. Ng, Y.-K. Woon, and D. H. Tran, "Concurrent semi-supervised learning of data streams," in *Data Warehousing and Knowledge Discovery*. 2011, pp. 445–459.

[16] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani, "Streaming-data algorithms for high-quality clustering," in *18th Int. Conf. Data Engineering*, 2002, pp. 685–694.

[17] V. K. Pachghare, V. K. Khatavkar, and P. A. Kulkarni, "Pattern based network security using semi-supervised learning," *Int. Journal of Information and Network Security (IJINS)*, pp. 228–234, 2012.

[18] SANS, Dec 2012, http://www.sans.org/reading-room/analysts-program/sans-survey-appsec.

[19] I. Santos, J. Nieves, and P. G. Bringas, "Semi-supervised learning for unknown malware detection," in *Int. Symp. Distributed Computing and Artificial Intelligence*, 2011, pp. 415–422.

[20] C. T. Symons and J. M. Beaver, "Nonparametric semi-supervised learning for network intrusion detection: combining performance improvements with realistic in-situ training," in *5th ACM Workshop on Security and Artificial Intelligence*, 2012, pp. 49–58.

[21] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *9th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*. 2003, pp. 226–235.

[22] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Machine Learning*, vol. 23, no. 1, pp. 69–101, 1996.