# Stream Learning and Anomaly-based Intrusion Detection in the Adversarial Settings

Eduardo Viegas, Altair Santin, Vilmar Abreu
Graduate Program in Computer Science
Pontifical Catholic University of Parana
Curitiba, Parana - Brazil
{eduardo.viegas, santin, vilmar.abreu}@ppgia.pucpr.br

Luiz S. Oliveira
Informatics Department
Federal University of Parana
Curitiba, Parana - Brazil
luiz.oliveira@ufpr.br

***Abstract*—Despite existing many anomaly-based intrusion detection studies in the literature, they are not frequently adopted by the industry in production environments (products). Such a usage gap occurs mainly due to the difficulty to maintain the detection rate in acceptable level, given the occurrence of false alarms. In general, the literature does not consider the *adversarial settings,* when an opponent attempt to evade the detection system, thus possibly rendering the system unreliable over time. In this paper, we propose and evaluate a new approach to reliably perform real time stream learning for anomaly-based intrusion detection. We employ a class-specific stream outlier detector to automatically update the intrusion detection engine over the time, and a rejection mechanism, which makes it possible to obtain indications that an evasion attempt might being happening. Furthermore, the proposal is resilient to *causative* attacks, providing a secure intrusion detection mechanism even when the attacker can inject misclassified instances in the training dataset. The evaluation tests show that the proposed approach is resilient to *exploratory* attacks, allowing the system administrator to know when an evasion attempt might be occurring.**

***Keywords—Adversarial Settings, Outlier Detection, Stream Learning, Intrusion Detection.***

## I. INTRODUCTION

A common approach used for the detection of attacks in the network is through an Intrusion Detection System (IDS). The IDS allows the detection of attacks, malicious or inadequate usage of a computational system or a network of computers [1]. In general, the detection of intrusion attempts in most IDSs is performed through the usage of machine learning techniques [2]. In such approach, the system is trained with a set of known profiles/behaviors. Afterwards, during its deployment in production, any event, e.g. a network packet, which significantly deviates from the previously known profiles is classified as an intrusion attempt [3]. In this way, the anomaly-based intrusion detection can detect new attacks.

However, despite the promising results reported in the literature, the anomaly-based intrusion detection approach is rarely considered in production (real-world) environments [4]. Thereby, the signature-based still is the most used approach [5], in which the classification is performed by pattern matching a set of well-known attacks. This discrepancy between the literature and real-life occurs mainly because the high number of false alarms [4] and the need to update the intrusion detection algorithm over time [6] due to the changes in the environment behavior, such as the occurrence of new network traffic and new attacks [5].

In light of this, over the last years, a great amount of research effort has been performed on stream learning techniques [7]. Such a technique is often employed in scenarios in which the set of target concepts (classes) changes over time [7]. For instance, in the network-based intrusion detection context, the legitimate client (normal) behavior changes over time with the usage of new services, whilst the attacker behavior (intrusion) also changes gradually due to the generation of new attacks [4]. Therefore, in an evolving context the detection mechanism update is often performed based on time intervals (*sliding window*) [8]. Thereby, due to the characteristics of the data, recent events are given a greater importance during the detection stage, whilst older events are often discarded [8].

Because of its capacity to deal with evolving data, several works have argued that the stream-based learning techniques enables the usage of anomaly-based intrusion detection approaches in production environments [2]. However, in the literature, there is a lack of proper evaluation of such systems, especially in the intrusion detection field [4]. In general, the works in literature does not consider the adversarial settings, where an attacker will attempt to evade the intrusion detection mechanism, either by perverting the intrusion detection mechanism properties or by injecting attacks during the training stage [9].

In this paper, we present an approach to reliably deal with evolving data streams to perform anomaly-based intrusion detection. The proposal relies in a class-specific stream outlier detector to automatically and reliably update the intrusion detection engine over time. The proposal considers the adversarial settings by rejecting potentially evasion attempts or non-reliable decisions.

## II. BACKGROUND

In this work, two approaches are considered for anomaly-based intrusion detection: traditional machine learning and stream learning.

### A. Traditional Machine Learning (TML)

The traditional machine learning refers to the pattern recognition algorithms [10], in which its process commonly relies in an immutable database. In this case, the intruder and normal user behavior are obtained and stored in a dataset. The dataset, usually referred as training dataset, is used during the classifier (pattern recognition algorithm) training stage. During this process, a false-positive (FP) and false-negative (FN) rate are estimated, through another dataset commonly referred to as

testing dataset. The FP rate refers to the rate in which normal events are wrongly classified as an intrusion attempt, whilst the FN rate refers to the rate that intrusion events are wrongly classified as normal activity. In the traditional machine learning process, when the attacker behavior changes, the pattern recognition algorithm must be retrained. However, the training process is often a costly task, as the environment must be monitored, the new behaviors must be identified, often manually, the classifier must be retrained and the FP and FN rates must be estimated. Moreover, identifying the environment behavior change is a challenging task, given the identification is often based in the increase of the FP and FN rates, which is also identified manually through experts' assistance [4].

The easiness to update becomes relevant in the anomaly-based intrusion detection field. New attacks are discovered in a daily-basis, along with new services and their contents. Thus, regardless of the used intrusion detection mechanism, the system must be continually updated [4].

### B. *Traditional Stream Learning (TSL)*

Traditional Stream Learning techniques aim at automatically dealing with the environment changes over time [7]. In such a case, the stream learning algorithm is executed in a repeated cycle, in which the algorithm is updated according to the incoming events from the network data stream [8]. Thereby, such process, occurs per memory and processing bounds. In this case the memory bounds are defined according with the number of events considered during the classification process. For this purpose, most strategies rely in a *sliding window* approach, in which a window (pre-defined range) of recent events is maintained, and the older events are discarded according to a set of rules [8].

Due to its adaptive nature, a data stream classification algorithm must present the following properties [8]: (i) process and inspect one example at a time, at most; (ii) use a limited amount of memory; (iii) classify in a restricted amount of time and (iv) predict new events at any time.

### C. *Adversarial Settings*

Over the last years, the traditional machine learning and stream learning techniques have been successfully applied to several fields [4], such as image recognition, product recommendation systems, natural language translation [11] amongst others. However, in the intrusion detection field, there still a lack of applicability in production usage [4].

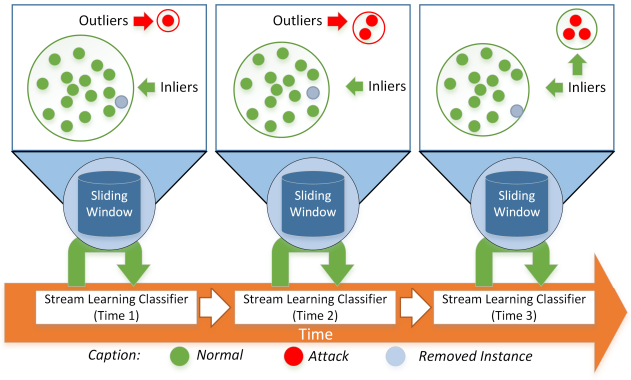


Figure 1 – Sliding Window behavior in Traditional Stream Learning Classifier. Outliers (Attacks) are added in the Sliding Window regardless of their class.

An emerging field of research, known as *adversarial machine learning* [9] considers the use of *machine learning* in the settings of an adversary – called *adversarial settings*. In such cases, the adversary (attacker) will attempt to evade the intrusion detection mechanism using sophisticated types of attacks, called *causative* and *exploratory* [9]. The *causative* attacks refer to attacks that occurs during the training process [9], e.g. the attacker inject misclassified intrusions into the training dataset as normal events. On the other hand, the *exploratory* attacks aim at exploring the machine learning algorithm properties [9], e.g. craft the intrusion attempt in a manner that the detection engine classifies it as a normal activity.

An example of *causative* attack at a traditional Stream Learning classifier is shown in Figure 1. In such scenario, the classifier is classifying attacks (Outliers) events over time. At Time 1 (Figure 1, Time 1) the *sliding window* is fully populated with inliers (normal) events, thereby the attack is classified as outlier. However, regardless of its assigned class, the attack, initially classified as outlier, is added in the *sliding window*, whilst an inlier (oldest event in the *sliding window*) is removed. The same property occurs at Time 2 (Figure 1, Time 2), however the new attack is still classified as outlier, as there are not enough events in its neighborhood. Finally, at Time 3 (Figure 1, Time 3) the attacker can inject enough attacks in the *sliding window*, as the number of neighbors is close enough to form a cluster and be classified as inliers (normal) events.

Thus, the usage of machine learning techniques, specially Stream Learning, in the *adversarial settings* seeks at designing detection mechanisms that can resist or to be resilient to the before mentioned (sophisticated) attacks.

## III. PROPOSAL

The objective of this proposal is to provide a reliable stream learning approach for anomaly-based intrusion detection that can automatically update the intrusion detection engine over time. Therefore, the proposed approach relies in a class-specific stream outlier detection algorithm to be resilient to both *causative* and *exploratory* attacks. The overall proposed method is shown in Figure 2 and described in the next subsections.

### A. *Detection Scheme*

The proposed method considers that the detection scheme relies in a class-specific stream outlier detection algorithm. For example, an outlier detection for normal events and an outlier detection for attack events. The detection is performed accordingly to Figure 2, (*i*) the set of features are extracted from the considered event, e.g. a network packet; (*ii*) a feature vector

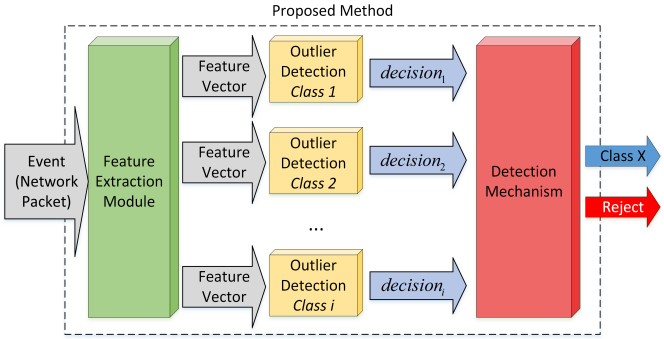


Figure 2 – Proposed method for anomaly-based intrusion detection through stream learning algorithm in adversarial settings.

is supplied to each outlier detection algorithm; (*iii*) each outlier detection perform its detection, assigning a class either outlier (event does not belong to outlier detection class group) or inlier (event does belong to outlier detection class group); (*iv*) the detection engine receives the decision from each outlier detection and attempts to find an consensus among the decisions; (*v*) if a decision unanimity is found the class is assigned, otherwise, the event decision is rejected.

When receiving an event decision, the detection engine decides whether the event classification is reliable or not. The class assignment reliability of an event classification (output 'class X' in Figure 2) comes from the nullity of intersection of decision from all classifiers. The reliability computation process is shown in Eq. 1, where $decision_i$ denotes to each Outlier Detection classifier output.

$$\bigcap_{i=1}^{n}(decision_i) = \emptyset \quad (1)$$

As an example, consider two outlier detection algorithms (Eq. 1), one for normal events and one for attacks; an event which is classified as an inlier for normal and outlier for attack is reliable – the decision is an unanimity, because there is not intersection between classification classes in the different detection engine for the same event. However, an event which is classified as inlier for more than one outlier detection should be rejected, as the decision is not reliable. Rejected classifications indicates that a potential evasion attempt or a false alarm might be occurring and another detection mechanism should be used, for instance, a signature-based intrusion detection mechanism or manual inspection.

### B. Ensuring Adversarial Machine Learning - Exploratory

Unlike the traditional stream learning algorithms, our approach provides resilience to *exploratory* attacks, by relying in the *immutable* behavior of each outlier detection algorithm. The immutable behavior is defined by a restriction that do not allow an outlier to become an inlier in the outlier detection algorithm over time (in a considered sliding windows). Our approach considers that in the anomaly-based intrusion detection field an event that is initially classified as outlier will not become an inlier at any moment in time. For example, an attack that was classified as an outlier (attack) by the normal outlier detection algorithm, must not be classified as a normal event afterwards, even if its occurrence increases in the *sliding window* over time.

By using the immutable behavior, the attacker will not be able to exploit the sliding window range to pervert (pollute) the classification of events being analyzed by an outlier detector. It is important to note that events classified as inlier continue to be added into the stream learning *sliding window*, thus the algorithm is still able to adapt to changes in the stream. But, the proposal mitigates a possible evasion attack, when the number of outlier events become predominant in a sliding window, therefore they will trigger the behavior mutation from outlier to inlier.

### C. Ensuring Adversarial Machine Learning - Causative

In order to provide resilience to *causative* attacks, the proposal relies in both *immutable* behavior (Section III.B) and class-specific outlier detectors. It considers that resilience to *causative* attacks must be provided at two stages: *initial training* and *ongoing readapting (retraining)*.

The *initial training* is related to the initial outlier detector *sliding window* population – the filling of events in a sliding window. In this stage, the outlier detectors *sliding window* are still being populated, thus susceptible to *causative* attacks. Thereby, the proposed approach assumes that at least there are an initial population allowing the correct classification for one outlier detector, since the *sliding window* will be updated according to the initial events. A way of assuring the reliability of the initial training is preset the sliding windows with a predominant number of copies of the same inlier event. Thus, given the outlier detector is reliable, the classification outputs can be trusted if decision unanimity is reached, otherwise the classification is rejected.

In order to provide a secure *ongoing readapting,* the proposal relies in both class-specific single class detection mechanism and *immutable* behavior (Section III.B). The single class detection mechanism provides resilience to event behavior manipulation. For example, the attacker must manipulate the event behavior in a manner that it behaves as a normal event, while also being an outlier for the attack outlier detection mechanism. Whilst, the *immutable* behavior difficult the attack over the *sliding window*, since the attacker must have skills to manipulate the events in a manner that pervert all outlier's detectors.

## IV. EVALUATION

Due to the well-known limitations of the current approaches regarding datasets for network-based intrusion detection evaluation [4], the approach proposed in our previous work [19] was used to gather the network traffic. In this way, two classes of network traffic compose the dataset: normal and attack. The testbed consists of 100 interconnected client machines, 3 attacker hosts and a single server.

To generate the *normal* traffic, the services provided in the testbed scenario were HTTP (*Hypertext Transfer Protocol*), SSH (*Secure Shell*), SMTP (*Simple Mail Transfer Protocol*), SNMP (*Simple Network Management Protocol*) and name resolution requests (DNS, *Domain Name System*). Three set of attacks were generated: SYNFlood, UDPFlood and ICMPFlood. The testbed described in [19] was executed for 30 minutes, the total amount of generated network traffic for each service and attack is reported in Table 1. For each network packet 23 features are extracted [19].

For the tests purposes the well-known Micro-cluster-based Continuous Outlier Detection (*MCOD*) [12] algorithm has been considered in our proposed method (Figure 2, Section III). For comparison purposes two other approaches were considered: the Traditional Machine Learning (TML, Section II.A) and Traditional Stream Learning (TLS, Section II.B).

TABLE I. NETWORK TRAFFIC DISTRIBUTION

| Class | Traffic | Number of Packets |
|---|---|---|
| Normal | HTTP | 20,238,802 |
| | SMTP | 2,298,222 |
| | SSH | 1,048,482 |
| | SNMP | 3,017,731 |
| | DNS | 135,188 |
| Attack | SYNFlood | 471,288 |
| | UDPFlood | 121,645 |
| | ICMPFlood | 130,698 |

### A. Model Obtainment Process

For the proposed method (Section III), two classes were considered: *normal* and *attack*. Thereby, for each test, two outlier detectors were used, one for *normal* and one for *attack*. A *sliding window* of 10,000 events was considered. A total of 50 events were established as neighbor (k) parameter. Each class outlier detection has its own radius parameter. A series of tests were conducted to establish the radius parameters; the choosing criteria was to minimize the *fitness* value in Eq. 2.

$$fitness = error_{rate} + rejection_{rate} \quad (2)$$

The $error_{rate}$ and $rejection_{rate}$ were defined through the detection of the initial 10,000 *normal* events followed by detecting 10,000 *attack* events from the training dataset (Table 1). The radius values for each class outlier detector, *normal* and *attack*, was varied in a 0.01 interval from zero to 2.00.

The *k-Nearest Neighbor* (kNN) classifier was used for the TML (Section II.A). To allow comparison, a total of 5000 events for each class, *normal* and *attack*, are used for the classifier neighbor computation. The 5000 events of each class are defined by the *k-means* clustering algorithm [13], using the training dataset (25% of randomly chosen events from Table 1). The kNN neighbors set are not updated during the classification process. Finally, for the TSL (Section II.B2), the MCOD is used. However only the *normal* class is considered, as commonly performed in related works [12], whilst the radius obtainment process was established only by the $error_{rate}$ minimization.

### B. Traditional Evaluation

Initially, the traditional evaluation process was considered for the evaluated approaches. In the traditional evaluation, the *adversarial settings* (Section II.C) are not considered.

For the kNN classifier, the dataset was divided into: *training*, *validation* and *testing*, containing, 25%, 25% and 50%, respectively of the whole dataset (Table 1). Due to the adaptive nature of the considered stream learning algorithm, the whole dataset is used for the traditional evaluation. The events are replayed in the exact same order as they appear in the original dataset (Table 1). Table 2 shows the accuracy rates regarding each of the evaluated approaches, where the *method* column refers to the used approach during the detection stage. Each approach is tested with a different set of attacks used during the training stage, shown in brackets in the method column.

TABLE II. PROPOSAL AND TRADITIONAL EVASION EVALUATION

| ***Method*** | ***Accuracy (Reject)*** | | | |
|---|---|---|---|---|
| | ***Normal Accuracy (Rejection)*** | ***SYNFlood Accuracy (Rejection)*** | ***UDPFlood Accuracy (Rejection)*** | ***ICMPFlood Accuracy (Rejection)*** |
| Proposed Approach MCOD (SYNFlood) | 100.00% (0.04) | 100.00 (0.00) | - (100.00) | - (100.00) |
| Traditional kNN (SYNFlood) | 99.83 (N.A.) | 100.00 (N.A.) | 100.00 (N.A.) | 0.01 (N.A.) |
| Proposed Approach MCOD (UDPFlood) | 100.00 (0.98) | - (100.00) | 100.00 (0.10) | - (100.00) |
| Traditional kNN (UDPFlood) | 99.93 (N.A.) | 49.97 (N.A.) | 100.00 (N.A.) | 0.01 (N.A.) |
| Proposed Approach MCOD (ICMPFlood) | 100.00 (0.97) | - (100.00) | - (100.00) | 100.00 (0.12) |
| Traditional kNN (ICMPFlood) | 100.00 (N.A.) | 3.23 (N.A.) | 100.00 (N.A.) | 100.00 (N.A.) |
| Traditional Stream Learning MCOD | 99.19 (N.A.) | 0.81 (N.A.) | 0.69 (N.A.) | 0.22 (N.A.) |

One can notice that both the proposed approach and the traditional machine learning (kNN) can detect the same set of attacks, with a significantly high accuracy rate. Regarding the detection of attacks, both the proposed approach and the kNN presented a FN rate of zero percent, when detecting the same set of attacks the system has been trained with. Considering the FP rate, the kNN classifier achieved 0.17, 0.07 and zero percent when trained with SYNFlood, UDPFlood and ICMPFlood attacks, respectively. The proposed approach achieved a FP rate of 0.00 percent in all tested cases. However, the proposed approach rejected potentially wrong classifications. In such a case, 0.04, 0.98 and 0.97 percent of normal events were rejected for SYNFlood, UDPFlood and ICMPFlood attacks, respectively. It can be observed that the proposed approach presents a similar detection accuracy when compared to the traditional machine learning approach. However, the proposed approach rejects potentially wrong classifications, which can be observed by comparing the kNN FP rate and the proposed approach rejection rate.

Considering the traditional stream learning approach, it was possible to notice that when events are replayed in the exact same order as they appear in the original dataset (Table 1), the method can detect only the initial attacks – when the *sliding window* is almost fully populated with normal events. However, as the attacks occurrence increases, the further attack events are classified as inlier (*normal*). Such a property occurs due to the adaptive nature of stream learning algorithms, which allows that an event, initially classified as outlier (*attack*), to be added in the *sliding window*, hence, allowing that an attack to become an inlier over time, perverting the outlier detector behavior. In this manner, the traditional stream learning algorithms, must consider such property – in the intrusion detection field, which is dealt in this work by considering the *immutable* behavior (Section III.B).

### C. Adversarial Settings – Exploratory attacks

Two types of attacks were evaluated in this experiment: the *traditional evasion* and the *window interval exploit*.

#### 1) Traditional Evasion

The *traditional evasion* refers to the detection of attacks with a different kind of behavior to the attack that the system was trained with, however with the same or similar outcome. For example, attacks aiming at generating a significant amount of network traffic at the targeted victim, regardless of the considered protocol, e.g. UDP, TCP or ICMP floods. In this way, each of the considered approaches were also tested with a different flood attack than the system was trained with. The obtained accuracy is shown in Table 2.

Regarding the traditional machine learning (kNN), the attacker could evade the system, while generating an attack that produces the same or similar outcome. When the kNN was used, the evasion possibility was evidenced for all evaluated attacks: SYNFlood, UDPFlood and ICMPFlood. For instance, when the system was trained with SYNFlood attacks, the attacker is still able to evade the detection system by generating ICMPFlood attacks. Moreover, the tested approach accepted only the classifications outputs regarding the attacks the system was trained with. Such a high rejection rate, and in this case reliability increase, due to the possible increase in the error rate,

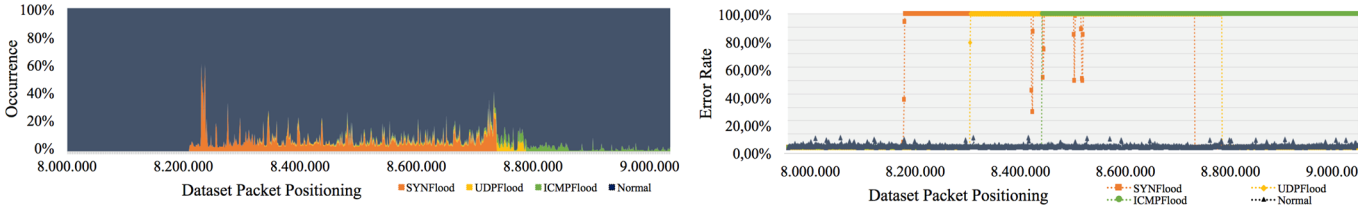


Figure 3 – Traditional Stream Learning approach behavior under network traffic intensive attacks, left chart shows the network packet classes occurrence while right chart shows the related error rate. Attack detection error rate increases according to the occurrence of attacks in the *sliding window*.

occurred due to the lack of decision unanimity between the outlier detectors, and thereby rejecting the assigned class.

### 2) *Window Interval Exploit*

The second evaluated *exploratory attack* is called as *sliding window exploit*. The *sliding window exploit* attack aims at evaluating the traditional stream learning accuracy according to the attack occurrence in a *sliding window*. As noted in Section IV.B, the increase in the attack frequency in the *sliding window* renders the stream learning algorithm unreliable. Figure 3 (right chart) shows the error rate regarding each of the evaluated attacks, during the 8 to 9 million packets in the created dataset. The error rate is evaluated in a 1,000 packets interval.

It is possible to observe that during the normal events detection, the traditional stream learning algorithm error rate remains similar to the rate obtained during the traditional evaluation (Section IV.B, 0.81 percent). However, as the attacks begins to occur (around the 8.2 millionth packet), the attack detection error rate increases, due to the increasing in the attack occurrence. In this manner, the attacker can exploit the traditional stream learning algorithm *sliding window*, by increasing the attack occurrence (Figure 3, left chart), causing the attacks to be classified as inlier (*normal*) due to their frequency increase in the *sliding window*.

The *sliding window exploit* does not occur in the proposed approach due to the *immutable* behavior (Section III.B) and the class-specific single class detection mechanism (Section III.A). The results are shown in Table 2. The attacker is not able to add attacks in the normal outlier detector *sliding window* due to the *immutable* behavior. Whilst, if the detection mechanism wrongly classifies an event, and thereby add it in its *sliding window*, the event will be rejected, because it will not be possible to establish an unanimity between the others outliers' detectors, given the outliers decision have a non-null intersection.

## D. *Adversarial Machine Learning – Causative attacks*

Finally, to evaluate the *causative* attacks resilience, a training dataset poisoning approach was adopted. The traditional machine learning (TML) and the proposed approach were evaluated regarding the influence that attacks, initially injected into the training dataset as normal events, have in the resulting accuracy. Thereby, the goal was to evaluate each of the considered methods, regarding their resilience to dataset poisoning attacks. Figure 4 shows the relation between the attack detection rate and the attacker control percentage over the training dataset, while successfully injecting attacks labeled as normal activity.

Regarding the TML, it is possible to note that the three evaluated attacks can evade the detection mechanism when injected in the training dataset as normal events. The accuracy rate for SYNFlood attacks dropped for 50% when 3% of normal events were SYNFlood injected attacks. Whilst, for ICMPFlood and UDPFlood the attacker could evade the detection system when 9% of attacks were injected. On the other side, the proposed approach could detect when attacks were injected into the training dataset and reject further classifications. Such a characteristic occurred due to class specific outlier detector, the attacks injected into the training dataset as normal events incurred in a lack of outliers unanimity in the classification decision process, thereby, the events were rejected.

# V. Related Works

The lack of usage of anomaly-based intrusion detection methods in production environments was noted over the last years by several works [4]. Such gap may be caused by several aspects; however, it is a consensus that the detection method must be at least reliable and easy to update [4].

The detection reliability is often considered in other areas [11], to this end, in general the authors [14] rely in the output class probability to reject or not the decisions, while other approaches uses an ensemble of classifiers and establishes the classification reliability through a majority voting approach [15]. Despite being often considered in other areas, to the best of our knowledge, the classification reliability has not been considered in stream learning field yet. However, it must be in our opinion, because the sliding window can be attacked to

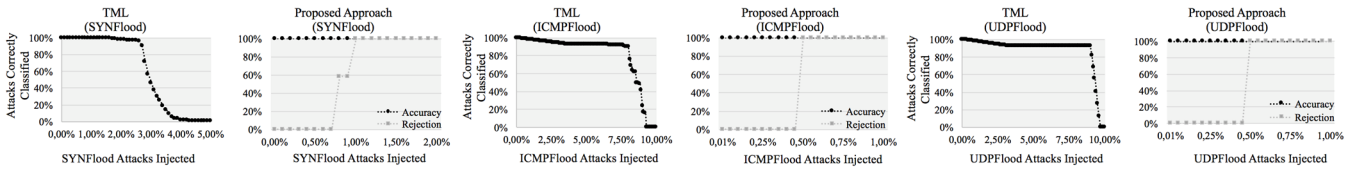


Figure 4 – Traditional Machine Learning (TML) and Proposed Approach resilience to *causative attacks* (training dataset / poisoning attacks). The horizontal axis shows the rate of attacks injected into the training dataset labeled as normal activities. The vertical axis shows the accuracy and rejection rate impact while detecting such attacks, having the infected training dataset.

deceive the outlier detector. Some authors, however, considered the adversarial settings in anomaly-based intrusion detection.

Ling Huang et al. [9] defined a taxonomy used in their work to classify possible adversarial attacks against the machine learning system. The authors also evaluated the impacts that a poisoned training dataset incur in the classifier accuracy, in all evaluated cases the classifier became unreliable when the training dataset had misclassified attacks injected.

In the spam detection scenario, Blaine Nelson et al. [15] evaluated the training dataset poisoning impact on accuracy, the authors reported a 36% misclassification increase when the attacker have control of only 1% of the training dataset. The authors also evaluated a *causative attacks* resistance approach by identifying whether the new added instance results in accuracy improvements or not, despite this approach is effective, the authors relied in a supervised dataset (when all instances are prior classified). Such an approach cannot be employed in production as the instances are not prior labeled and the accuracy cannot be estimated in real time. In the malicious PDF detection scenario, Srndic and Laskov [16] evaluated a set of attacks against a well-known learning-based classifier for malicious PDF files, the authors could drop the classification accuracy from almost 100% to 28%. The authors also suggested that a multiple classifier system should be more resilient to such adversarial attacks, due to the need to evade several complementary classifiers.

Few authors address *causative* attacks in the network intrusion detection field [17], for instance, Benjamin et al. [18] developed the ANTIDOTE which relies in a robust PCA and a robust Laplace threshold that is less sensitive to poisoning attacks. However, their approach remains susceptible to *exploratory* attacks.

To the best of our knowledge this is the first work to address both *causative* and *exploratory* attacks using stream learning algorithms for intrusion detection field. Our approach remains reliable during both attacks, *causative* and *exploratory*, by employing a rejection mechanism and a class-specific outlier detector.

## VI. Conclusion

The anomaly-based intrusion detection has been extensively studied over the last years. However, despite promising results such an approach is hardly used in production environments, mainly due to the difficulty in providing reliable and updatable detection methods. The main issue is the use of machine learning in adversarial setting, in which an attacker attempt to evade the detection mechanism.

This paper presents a novel anomaly-based intrusion detection method which addresses the use of machine learning in the adversarial settings. The proposed approach relies in class-specific single class detection mechanism. It can detect possible evasion attempts, while providing a reliable and updateable detection engine. The reliability is achieved by rejecting potentially wrong classifications or evasion attempts. Through a set of comprehensive experiments, the proposed method has shown its resistance for both *causative* and *exploratory* attacks.

As future works, we are pursuing the reduction of the rejection rate while still being resilient to adversarial attacks. To this end, we plan to employ a hybrid approach which relies in both stream learning and traditional machine learning algorithms.

## Acknowledgment

This work was partially sponsored by Intel Lab's University Research Office and the Brazilian Coordination for the Improvement of Higher Education Personnel (CAPES), grant 99999.008512/2014-0.

## References

[1] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Ferná Ndez, and E. Vá Zquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Comput. Secur.*, vol. 28, pp. 18–28, 2009.

[2] V. Jyothsna, V. V Rama Prasad, and K. Munivara Prasad, "A Review of Anomaly based Intrusion Detection Systems," *Int. J. Comput. Appl.*, vol. 28, no. 7, pp. 26–35, 2011.

[3] D. E. Denning, "An intrusion-detection model," *Proc. - IEEE Symp. Secur. Priv.*, no. 2, pp. 118–131, 2012.

[4] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," *2010 IEEE Symp. Secur. Priv.*, pp. 305–316, 2010.

[5] S. Axelsson, "The Base-Rate Fallacy and the Difficulty of Intrusion Detection," *ACM Trans. Inf. Syst. Secur.*, pp. 186–205, 2000.

[6] F. Maggi, W. Robertson, C. Kruegel, and G. Vigna, "Protecting a moving target: Addressing web application concept drift," *Lect. Notes Comput. Sci.*, vol. 5758 LNCS, pp. 21–40, 2009.

[7] H. He, S. Chen, K. Li, and X. Xu, "Incremental learning from stream data.," *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 1901–14, 2011.

[8] A. Bifet, R. Gavaldà, "Learning from time-changing data with adaptive windowing", *Proc. 7th SIAM Int. Conf. Data Mining*, pp. 443-448, 2007.

[9] L. Huang, A.D. Joseph, B. Nelson, B. Rubinstein, and J.D. Tygar, "Adversarial Machine Learning," Proc. Fourth ACM Workshop Artificial Intelligence and Security, pp. 43-57, 2011.

[10] I. Corona, G. Giacinto, and F. Roli, "Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues," *Inf. Sci. (Ny).*, vol. 239, pp. 201–225, Aug. 2013.

[11] L. S. Oliveira, R. Sabourin, F. Bortolozzi, and C. Y. Suen, "Automatic recognition of handwritten numerical strings: A Recognition and Verification strategy," IEEE Trans. Pattern Anal. Mach. Intell., vol. 24, no. 11, pp. 1438–1454, 2002.

[12] M. Kontaki, A. Gounaris, A. N. Papadopoulos, K. Tsichlas, and Y Manolopoulos, "Continuous monitoring of distance-based outliers over data streams," *IEEE Int. Conf. on Data Eng. (ICDE)*, pp. 135-146, 2011.

[13] K. Alsabti, S. Ranka, and V. Singh, "An efficient k-means clustering algorithm," *Electical Eng. Comput. Sci.*, vol. 43, pp. 2-7, 1997.

[14] G. D. C. Cavalcanti, L. S. Oliveira, T. J. M. Moura, and G. V. Carvalho, "Combining diversity measures for ensemble pruning," Pattern Recognit. Lett., vol. 74, pp. 38–45, 2016.

[15] B. Nelson, M. Barreno, F.J. Chi, A.D. Joseph, B.I.P. Rubinstein, U. Saini, C. Sutton, J.D. Tygar, K. Xia, "Exploiting Machine Learning to Subvert Your Spam Filter", *Proc. First Workshop Large-Scale Exploits and Emergent Threats*, pp. 1-9, 2008.

[16] N. Srndic and P. Laskov, "Practical evasion of a learning-based classifier: A case study," Proc. - IEEE Symp. Secur. Priv., pp. 197–211, 2014.

[17] G. Wang, S. Barbara, T. Wang, H. Zheng, and B. Y. Zhao, "Man vs . Machine : Practical Adversarial Detection of Malicious Crowdsourcing Workers," 23rd USENIX Secur. Symp., pp. 239–254, 2014.

[18] B. I. P. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S. Lau, S. Rao, N. Taft and J. D. Tygar, "ANTIDOTE : Understanding and Defending against," SIGCOMM, no. November, pp. 1–14, 2009.

[19] E. Viegas, A. Santin, A. França, R. Jasinski, V. Pedroni, and L. Oliveira, "Towards an Energy-Efficient Anomaly-Based Intrusion Detection Engine for Embedded Systems," *IEEE Transactions on Computers,* vol. 66, no. 1, pp. 1–14, 2017.