# A Method Aware of Concept Drift for Online Botnet Detection

**Bruno Henrique Schwengber**, **Andressa Vergütz**, **Nelson G. Prates Jr.**, **Michele Nogueira**
NR2/CCSC - Federal University of Paraná, Brazil
Emails: {bhschwengber, avergutz, ngpjunior, michele}@inf.ufpr.br

GLOBECOM 2020 - 2020 IEEE Global Communications Conference | 978-1-7281-8298-8/20/$31.00 ©2020 IEEE | DOI: 10.1109/GLOBECOM42002.2020.9347990

***Abstract*—Botnets deeply threaten cybersecurity due to their distributed and dynamic nature, causing attacks with severe consequences for users and companies, such as Distributed Denial of Service. Detecting botnets is challenging once they constantly evolve, resulting in fast behavior changes in network. Current techniques usually detect botnets without considering these changes and their fast adaptation to new behavior. Hence, this paper presents CONFRONT, a method aware of concept drift (fast changes in network behavior) for online botnet detection. Different from the literature, this paper introduces a new technique to detect concept drift and optimize botnet classification. CONFRONT employs features from network flow on the unsupervised concept drift detector and a supervised incremental botnet classifier. Results show CONFRONT feasibility, reaching 95% of accuracy in less than 1 ms.**

***Index Terms*—Concept drift, Botnet Detection, Security**

## I. Introduction

The increasing number of heterogeneous applications and device results in security issues [1]. Botnets pose a huge cybersecurity threat once they spread malwares and coordinate malicious activities in the Internet. The average loss resulted from attacks generated by botnets is around US$ 3.92 million [2]. Cisco estimates the number of connected devices will reach 29.3 billion by 2023 [3], and most of these devices are vulnerable. For instance, the devices in the Internet of Things (IoT) have vulnerabilities (e.g., authentication failures), being susceptible to become an infected device and, then, a bot in a botnet. Hence, the increasing number and heterogeneity of devices intensify botnets security concerns [4].

Botnets rely on connected devices to carry out a vast variety of cyberattacks (e.g., Distributed Denial of Service and phishing), causing severe consequences, such as financial and reputation losses [2]. However, botnet detection techniques use traditional classification tasks (training and testing) considering static data [5]. Nevertheless, botnets act very quickly and employ streaming data collected from network traffic. Such real-world data stream evolve over time, this is known as concept drift [6]. When a concept drift occurs, the classification model becomes outdated because data distribution has changed, and then it can not correctly detect botnets [7].

Studies in botnet detection, in general, ignore fast changes in network behavior (i.e., concept drift) [8]–[10]. However, few recent works [5] lead the application of supervised methods for identifying concept drift in online attack detection. Supervised concept drift detectors only use the classifier performance metrics, without considering data distribution. Unsupervised concept drift detectors monitor data distribution over the stream. There are two types of unsupervised detection, novelty detection [11], [12] and multivariate distribution monitoring methods [6], [13], [14]. Novelty detection methods suffer from the dimensionality curse (phenomena that occur when analyzing high dimensional data). In contrast, multivariate distribution methods deal properly with multidimensional streams. Thereby, online botnet detection rely on data streaming and evolve quickly, resulting in fast changes in data distributions and making detection a challenging problem. Hence, a botnet detection with concept drift awareness is crucial to identify changes over data distribution and detect botnets.

This paper introduces CONFRONT, a method aware of CONcept driFt foR ONline boTnet detection. Concept drift detection provides an optimized input for botnet detection. CONFRONT innovates employing unsupervised concept drift detection over a sliding window to monitor the changes in data distribution. The window contains network flow features (e.g., flow volume). Therefore, the method divides the input data window into two data blocks: old and new data. Each one receives a target number (label), 0 or 1 for old or new data, respectively. CONFRONT trains and tests a decision tree classifier periodically over data blocks. When a new data block contains separable samples to the old block, the method alerts a concept drift. Based on data window provided by the concept drift detection, CONFRONT uses an incremental learning classifier to detect botnets.

The CONFRONT evaluation follows a trace-driven approach, having as input a botnet dataset [15] containing network traffic and attacks. The dataset includes network traffic from Peer to Peer (P2P), Internet Relay Chat (IRC), and HyperText Transfer Protocol (HTTP). It encompasses different botnets attacks. Results show the potential of CONFRONT on employing concept drift and successfully identifying botnets by improving the performance of classification to higher than 95%. Moreover, the method detected botnets in less than 1 ms per instance overall classification time.

This paper proceeds as follows. Section II presents the related works. Section III details CONFRONT. Section IV describes the evaluation methodology and discusses results. Finally, Section V presents conclusions and future work.

This work was supported by CNPq/Brazil, grants #309129/2017-6 and #432204/2018-0, São Paulo Research Foundation (FAPESP), grant #2018/23098-0, Capes/Brazil, grants #88882.381963/2019-01 and #88882.461738/2019-01, and the joint NSF and RNP HealthSense project, grant #99/2017.

## II. Related Work

There are studies addressing online botnet detection [8]–[10]. In the literature, they employ supervised incremental classifier [8], unsupervised detection by reputation scores [10], and similarity analysis [9]. However, botnets act quickly and employ streaming data collected from network traffic. As real-world applications evolve, data concept drift occurs [6], i.e., the classification model becomes outdated because data distribution has changed, and then it can not correctly predict data for botnet detection [7]. Therefore, it is crucial to detect concept drift to precisely detect botnets.

For online attack detection, some studies [5] compared the overall performance of concept drifts methods such as Page-Hinckley Test (PHT) [16], Adaptive Windowing (ADWIN), and Fixed Windowing (FIXWIN) [17]. The authors showed the improvements of concept drift detection for online attack detection. PHT, ADWIN, and FIXWIN are the most prevalent methods for concept drift identification considering botnet detection [5]. The PHT method is well-established in concept drift literature. It monitors the performance metrics (e.g., accuracy), obtained by classifiers, sequentially and cumulatively, to differentiate the new values from the average metrics results. The FIXWIN method employs a sliding window, and the ADWIN method employs an adaptive window size. For all classifiers, in each round, the methods update the set of employed metrics. When the average metric values differ from a given threshold, the methods detect a concept drift in data. However, supervised methods detect concept drift only monitoring the metrics obtained by classifiers, without considering data distribution. Thus, concept drift detection only occurs when the data distribution $p(X)$ affects the classification result $p(y|X)$. Moreover, they require labeled data, increasing the detection cost.

Differently, some studies employed unsupervised methods for concept drift detection, such as novelty detection methods [6], [13], [14] and multivariate distribution monitoring methods [6], [13], [14]. The novelty detection computes the distance between data distribution clusters. A new cluster on data means a concept drift. However, this method suffers from a well-known problem, the curse of dimensionality. The multivariate distribution methods monitor data characteristics based on unlabeled data [6], [13], [14]. In this context, some studies employed semi-parametric likelihood and Principal Component Analysis (PCA) to monitor data changes, using only 10% of PCA smallest eigenvalues to detect concept drifts [13]. Other studies applied the highest eigenvalues of PCA since the original data characteristics are summarized in the vector top [14]. In contrast, another study presented a Discriminative Drift Detector (D3) [6]. It uses a linear discriminative classifier based on a sliding window to distinguish two sets of streaming data. Hence, it runs repeatedly until the streaming is active. Thus, this paper introduces CONFRONT, a method aware of concept drift for online botnet detection. Based on [6], CONFRONT detects concept drift using an unsupervised method, and detects online botnets.

## III. The CONFRONT Method

This section details CONFRONT, a method aware of concept drift for online botnet detection. From the network traffic, it extracts flow features and detects data concept drift to optimize the botnets detection. The next subsections describes the concept drift in online learning and the method proposed.

### A. Concept Drift in Online Learning

In dynamic and non-stationary environments, data distribution evolves, changing its distributions. In other words, it occurs the phenomenon of data concept drift. There are two ways of concept drift: real and virtual concept drift. The first one defines a change in the classification performance when the data distribution $p(X)$ affects the classification $p(y|X)$. In contrast, the virtual drift defines a distribution change without considering the classification performance [7]. In general, real and virtual concept drift detection use supervised and unsupervised methods, respectively [12]. Therefore, for online learning, the concept drift follows supervised or unsupervised detection approaches. Moreover, for classifiers and concept drift work together correctly, it is crucial to use incremental classifiers that adapt according occur a concept drift [6]. Hence, as CONFRONT considers an unsupervised concept drift detection, it follows the sequence shown in Fig. 1.

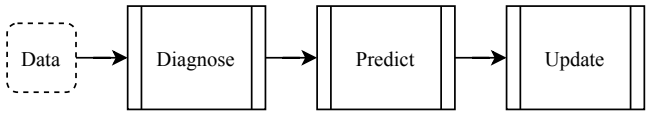



Fig. 1. Learning Sequence

The concept drift for an online learning model considers the diagnose, predict, and update phases (as shown in Fig. 1). The first phase compares two sets of network traffic data, containing old and new data in each one, to identify concept drifts. The second phase predicts the incoming new data, i.e., it detects botnets by classifiers. Then, the third phase updates the sets containing network traffic data regarding it occurs a concept drift. Thus, it keeps data always update. When a concept drift occurs, the set with old data is removed from the sliding window and is replaced with the incoming data.

### B. Detailing CONFRONT

CONFRONT identifies the traffic of different botnet classes. Thus, Fig. 2 depicts the three phases of CONFRONT and a proposal of positioning it in the network. The decision of CONFRONT positioning in a network depends, in general, on the requirements and characteristics of the network. In this work, we suggest positioning CONFRONT between the edge router and the victim (e.g., firewall). Despite not mandatory, we assume dedicated hardware to run CONFRONT. However, CONFRONT is designed to be flexible and implemented along with a firewall or an intrusion detection system (IDS). The method position may imply higher or lower efficiency. As the main contribution, CONFRONT innovates employing

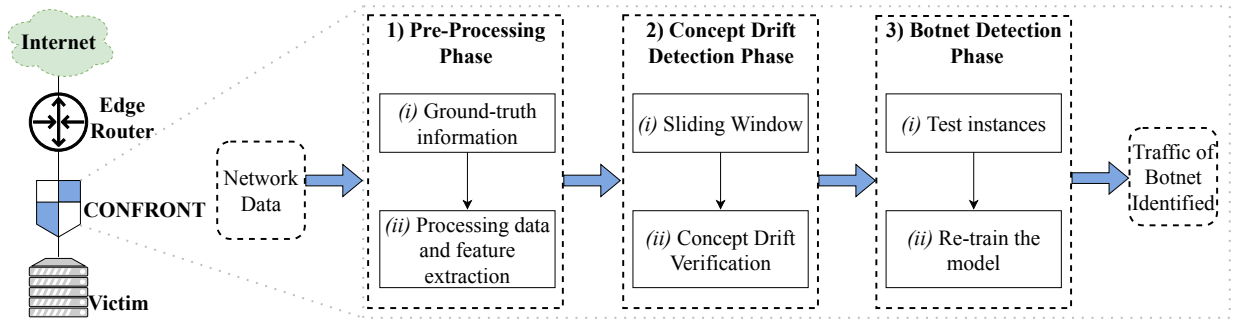



Fig. 2. The CONFRONT Architecture

unsupervised concept drift detection over a sliding window to monitor the data distribution changes.

The method follows three main phases: *(1) Pre-Processing*, *(2) Concept Drift Detection*, and *(3) Botnet Detection*, as shown in Fig. 2. The first phase pre-processes data filtering network traffic according to certain features (called ground-truth information). In this study, the ground-truth information is a traffic information. From ground-truth information, CONFRONT filters the network traffic into flows. As CONFRONT runs online, for feature extraction it collects only a sample from the traffic (e.g., one sample from several flows). The second phase builds the sliding window and detects concept drift. The third phase identifies botnets and re-trains the incremental algorithm. Next we detail specifically each phase.

*1) The Pre-Processing Phase:* This phase handles the network traffic in two steps: *(i)* the collection of ground truth-information and *(ii)* data processing to filter into flows ($F$) and feature extraction. In this study, CONFRONT uses the 5-tuple $\langle$ Source and Destination IP, Source and Destination MAC, and communication port $\rangle$ as ground-truth to correctly indicates what flow each traffic packet belongs to. Thus, based on the ground-truth, CONFRONT splits the network traffic into flows. As result, CONFRONT creates a subset of network data for each flow, denoted by a set $F = \{f_1, f_2, f3, ..., f_t\}$, where each $f$ means a specific set of packet in the network traffic. For processing data and feature extraction, the method considers as input the set of flows $F$. Afterwards, for each $f_t \in F$, CONFRONT extracts flow features, defined as a set $X^F = \{x_1, x_2, x_3, ..., x_i\}$, where each $x$ means a specific statistical feature from the flow $F$. CONFRONT employs as flow features the number of packets sent and received, the number of bytes sent and received, start and end timestamps of sending packets, start and end timestamps of receiving packets, the time interval of sending and receiving packets. Based on these features, the method employs machine-learning algorithms to differentiate network traffics [18].

*(2) The Concept Drift Detection Phase:* This phase identifies concept drifts in data distribution and builds the sliding data window for classification. Thus, it follows the Algorithm 1 adapted from [6]. The algorithm receives as input parameters the data streaming, the minimum number of instances ($\delta$), the number of new instances ($\sigma$), and a threshold ($\beta$).

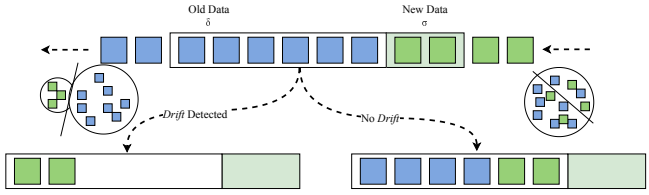



Fig. 3. Concept Drift Detection Workflow

As a result, the algorithm returns true or false (drift or not), informing the concept drift occurrence. The data streaming instances arrive sequentially for verification. In line 1, the algorithm initializes data window with size $\delta + \sigma$. It only detects a drift when the data window is fulfilled. Hence, in line 4, the algorithm verifies data window fill. If it is not full, it adds new instances to the data window. Otherwise, if the window $W$ is filled with $\delta + \sigma$, the method applies labels to the instances (label 0 for old data and label 1 for new data). Formally, in lines 8 and 9, the algorithm defines $\delta$ instances with label 0 and $\sigma$ instances with label 1. For a visually comprehension, Fig. 3 shows the data window division (next

**Algorithm 1** Concept Drift Detection

**Input:** $stream, \delta, \sigma, \beta$
**Output:** Drift or not

1: Initialize window $W$ where $|W| = \delta + \sigma$
2: Decision tree Classifier C
3: **for** $X, y$ in stream **do**
4: **if** $W$ is not full **then**
5: $W \leftarrow W \cup X$ {add X to the head of W}
6: **else**
7: $S$ is vector of $s$, $|W| = |S|$ {s is a slack variable}
8: $s = 0$ for $W[1, w]$ {label for old (0) and new (1)}
9: $s = 1$ for $W[w + 1, end]$
10: Train C$(W, S)$ { train C with S as labels of W}
11: **if** AUC(C, S) $\geq \beta$ **then**
12: drift = True {drift detected}
13: Drop $\delta$ elements from the tail of the $W$
14: **else**
15: drift = False {no drift}
16: Drop $\sigma$ elements from the tail of $W$
17: **end if**
18: **end if**
19: **end for**

we detail the figure). The method uses these instances to train and test a decision tree classifier in line 10.

Fig. 3 shows as data window construction works for concept drift detection. The full data window has a specific size defined as $\delta + \sigma$ and contains the old $\delta$ and new $\sigma$ instances of network data. As CONFRONT works with streaming data, network data are coming over time. Note that the network traffic is pre-processed before the drift detection. CONFRONT detects a concept drift when the decision tree classifier is able to separate the instances with performance higher than $\beta$. Line 11 of the algorithm runs the concept drift identification according to [6]. It points out the existence of new data instances i.e., there is a concept drift occurrence in the data distribution. Hence, when the method detects a concept drift, it removes a set of old $\delta$ instances from the window tail, updating the window for new changes (lines 12 and 13 from the algorithm). Moreover, regarding of concept drift occurrence, CONFRONT removes a set of $\sigma$ instances to substitute for new ones, keeping the data window always updated (lines 15 and 16 from the algorithm).

*3) The Botnet Detection Phase:* This phase receives as input the data window from the concept drift detection phase. It uses incremental machine learning algorithms to detect botnets because it deals with data streaming. In this work, the method applies the Hoeffding Tree algorithm based on the study [5]. Moreover, CONFRONT supports unbalanced data (e.g., dataset containing botnets and benign traffic with different amounts of data) because it works with incremental learning. The detection follows the prequential evaluation (test first, then train), unlike the traditional evaluation (train first, then test) since it is specific for online detection. Therefore, CONFRONT points out a botnet when the classifier result achieves high performance on accuracy and precision metrics.

A Hoeffding tree is an incremental classifier, it is a decision tree induction algorithm that is capable of learning from massive data streams. Hoeffding trees exploit the fact that a small representation is often enough to choose an optimal splitting attribute. This idea is supported mathematically by the Hoeffding bound, which quantifies the number of observations (e.g.,, instances) needed to estimate some statistics within a prescribed precision. A theoretically appealing feature of Hoeffding Trees not shared by other incremental decision tree classifiers is that it has reliable guarantees of performance. Using the Hoeffding bound one can show that its output is asymptotically nearly identical to that of a non-incremental classifier using infinitely many examples [19].

## IV. Performance Evaluation

As CONFRONT works between the edge router and the victim (e.g., firewall), the representative Botnet 2014 dataset [15] serves as input for CONFRONT performance evaluation. The dataset presents network traffic from 3 different datasets. It is relevant because it contains records of real network traffic from different types of botnets and different architectures. Table I shows the botnet distribution with percent of data in the dataset. The architectures include Peer to Peer (P2P), Internet Relay Chat (IRC), and HyperText Transfer Protocol (HTTP). The types of botnets include Neris, Rbot, Menti, Sogou, Murlo, Virut, NSIS, Zeus, SMTP Spam, UDP Storm, Tbot, Zero Access, Weasel, Smoke Bot, Zeus Control, ISCX, IRC Bot. Moreover, it contains regular traffic. This variety of botnets makes it possible to evaluate CONFRONT in a more realistic scenario as well as a better comprehension of data analysis and conclusions. Moreover, for being an available online dataset, it allows the easy experiment reproducibility.

TABLE I
Botnet Distribution in the dataset

| Botnet Name | Architecture | % of Data |
|---|---|---|
| Sogou | HTTP | 0,019 |
| Virut | HTTP | 13,74 |
| Neris | IRC | 17,67 |
| Rbot | IRC | 22,018 |
| Menti | IRC | 0,62 |
| Tbot | IRC | 0,283 |
| Murlo | IRC | 1,06 |
| NSIS | P2P | 2,645 |
| Zeus | P2P | 0,119 |
| SMTP Spam | P2P | 11,2 |
| UDP Storm | P2P | 9,63 |
| Zero Access | P2P | 0,221 |
| Weasel | P2P | 9,25 |
| Smoke Bot | P2P | 0,017 |
| Zeus Control | P2P | 0,016 |
| ISCX IRC Bot | P2P | 0,387 |

The dataset is the base for the extraction of the statistical flow features. As mentioned, CONFRONT employs the flow volume and timestamps extracted by Tshark tool. A flow encompasses the 5-tuple $\langle$ Source and Destination IP, Source and Destination MAC, and communication port $\rangle$. Fig. 4 shows the packet distribution and the botnet attacks over the dataset. It contains the number of packets sent and received in each flow. Moreover, it contains the botnet attack flows (red lines). In 70% and 90% of the dataset, occurred the botnet attacks overload. For botnet detection, CONFRONT receives as input ten flow features such as the number of packets sent and received, number of bytes sent and received, start and end timestamps of packets sending, start and end timestamps of packets receiving, and time interval of sending and receiving packets. Based on these features, CONFRONT applies the Algorithm 1 to build the data window and detect drifts. Further, it applies rules of prequential evaluation for the classifier (test first, then train), specifically for online training. We implemented the method and classifier by Scikit-Multiflow v0.4.1 and Pandas v1.0.3 libraries from Python v3.7.

### A. Concept Drift and Machine Learning Algorithms

This analysis deals with two concept drift detection methods (D3, ADWIN), one botnet detection method (Hoeffding Tree), and the proposed method (CONFRONT) that contains botnet and concept drift detection. Note that for the concept drift detection methods (D3 and ADWIN) we have added a botnet detection method to compare the results with CONFRONT. The concept drift detection algorithms are the base to measure the performance improvement in our proposal. As

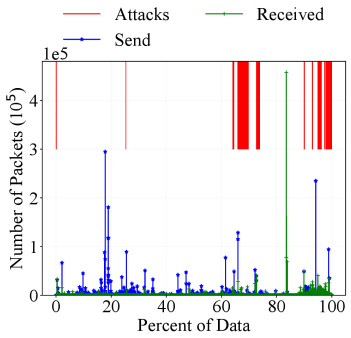



Fig. 4. Dataset Packet Distribution and Attacks

our problem lies in binary classification (e.g., botnet or benign traffic), we employ the incremental Hoeffding Tree algorithm.

For the detection using the Hoeffding Tree algorithm, we follow the prequential evaluation method. Train the classifier with $\sigma$ instances in the beginning, then test and re-train the classifier as the data window arrives, and finally present the concept drift detection. For performance evaluate the classifiers, we compute the following performance metrics: accuracy (Eq. 1), precision (Eq. 2), recall (Eq. 3), and F1 Score (Eq. 4). These metrics are based on the values about true positive *(TP)*, false positive *(FP)*, true negative *(TN)* and false negative *(FN)* achieved during classification. F1 Score is the overall classification time, given in seconds.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1\ Score = 2 * \frac{Recall * Precision}{Recall + Precision} \tag{4}$$

We compare the proposed method with two methods from the literature, ADWIN and D3 [5], following the performance metrics. Hence, D3 and CONFRONT methods received as input parameters the detailed dataset stream, $\delta$ value = 100 instances, $\sigma$ value = 10% of new instances for drift detection, and $\beta$ value = 70% in AUC score. Such values were defined as the best overall evaluation of $\delta = [100, 250, 500, 1000, 2500]$, $\sigma = [10, 20, 30, 40, 50]$ and $\beta = [60, 65, 70, 75, 80, 85, 90]$. In ADWIN method evaluation, were used the default parameters defined in scikit-multiflow library. We compared CONFRONT with ADWIN method because it achieved the best results for network security scenarios in [5]. D3 method also was compared with CONFRONT since our concept drift detection is based on it [6].

### B. Results

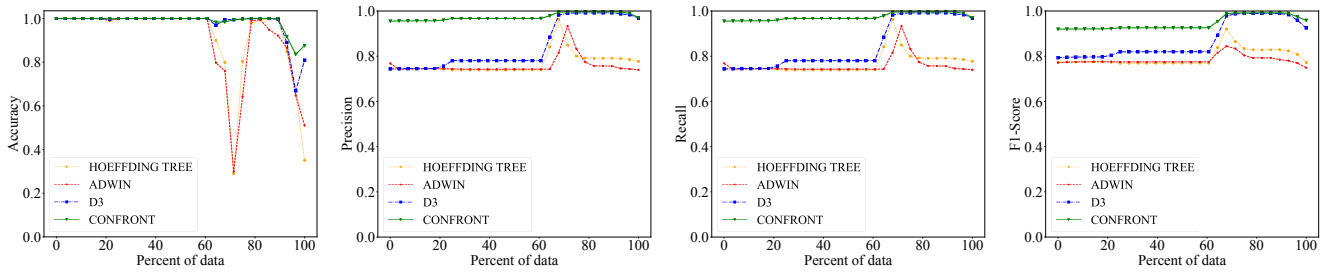
This subsection presents the results about the CONFRONT method evaluation performance. We have compared the method proposed with two concept drift detection methods (D3 and ADWIN) and the Hoeffding Tree classifier. For D3 and ADWIN we have added a botnet detection technique, and Hoeffding Tree does not consider concept drift detection. The results follow three approaches: evaluation of the classifiers metrics during execution, the average of these metrics in the general context of the dataset, and the performance of the classification time for each scenario.

Fig. 5 shows the performance metrics results for botnet detection. These results show the optimization of botnet detection metrics when supported by concept drift detection techniques. In the Fig. 5(a), the accuracy decrease in all methods due to the overload caused by the botnet attack. Such attack overload occurred around 70% of data, the same moment at accuracy values decrease. The Hoeffding Tree algorithm, without concept drift detection, achieved the worse accuracy results when compared to the D3, ADWIN, and CONFRONT. CONFRONT achieved the best accuracy result, even during the attack overload. In the precision metric results, as shown in Fig. 5(b), all methods obtained a good performance result, around 80% and 90%, even during the attack overload. However, Hoeffding Tree presented a poor result (around 70% precision) since it did not consider concept drift detection.

Figs. 5(c) and 5(d) show the recall and F-Score results, where ADWIN achieved higher false negative rates than Hoeffding Tree, decreasing its recall and F-Score rates. These results reflect the supervised concept drift detection effect since they consider only the classifier metrics results and ignore the original patterns contained in the data distributions. In contrast, D3 and CONFRONT employed unsupervised concept drift detection and reached better performance results, higher than 90% of recall and F-Score, during botnet detection. Finally, CONFRONT achieved the best result in recall and F-Score even under attack. Fig. 6(a) show the average performance metrics results. The average performance metrics considered were accuracy, precision, recall, and F1-Score. Based on the results, the use of concept drift methods increase the detection performance of the Hoeffding Tree algorithm. Nevertheless, among the methods of concept drift detection, CONFRONT outperforms the ADWIN and D3.

Fig. 6(b) show the detection time of each method. This detection time considers all dataset analysis. Among the methods, it is possible to notice a discrepancy of 100% more in detection time of the D3 method over the ADWIN method. Although the difference in detection time results, the D3 method presents detection performance results similar to the ADWIN. The CONFRONT method presents an increase of approximately 20% of detection time over the ADWIN. However, CONFRONT presents an increase in the detection performance metrics. Moreover, the CONFRONT detection time per instance is bellow 1 millisecond, being considered a significant value for online detection.

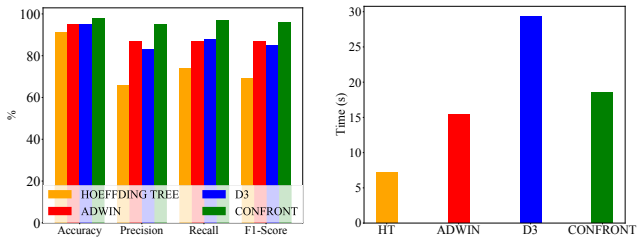
This performance evaluation shows that the concept drift detection techniques improve the overall classification metrics regarding without them. Moreover, ADWIN supervised concept drift method achieved worse performance than D3 and CONFRONT unsupervised methods. CONFRONT suc-




(a) **Acurracy** (b) **Precision** (c) **Recall** (d) **Fscore**

Fig. 5. Performance metrics over the dataset

cessfully identified the botnets and achieved the best results during the performance evaluation due to non-linear concept drift detection in data distribution that D3 method does not detect. This contrast provides support for the improved classification of CONFRONT method. Finally, the 20% increase of overall time classification presented between ADWIN and CONFRONT is overcome by the 67% classification error reduction stated by CONFRONT.




(a) Average Performance (b) Detection Time

Fig. 6. Performance of the Methods

## V. Conclusion

This paper presented CONFRONT, a novel method aware of concept drift in online botnet detection. CONFRONT evaluation employed an empirical dataset containing several types of botnets attacks and a comparison between state-of-the-art botnet identification with and without concept drift detection. Results have indicated the improvement of detection performance when comparing with ADWIN and D3 state-of-the-art methods. This work assists the research community to design network mechanisms to improve network security. As future directions, we intend to propose improvements such as implement a complete unsupervised system for online botnet detection and improve the detection time.

## References


[1] M. Ammar, G. Russello, and B. Crispo, "Internet of Things: A survey on the security of IoT frameworks," *Journal of Information Security and Applications*, vol. 38, pp. 8–27, 2018.

[2] R. SOBERS, "110 must-know cybersecurity statistics for 2020," (https://www.varonis.com/blog/cybersecurity-statistics/), 2020, last access in april/2020.

[3] Cisco, "Cisco annual internet report (2018–2023)," 2018. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

[4] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying IoT security: an exhaustive survey on IoT vulnerabilities and a first empirical look on internet-scale IoT exploitations," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2702–2733, 2019.

[5] P. Casas, P. Mulinka, and J. Vanerio, "Should I (re) Learn or Should I Go (on)? Stream Machine Learning for Adaptive Defense against Network Attacks," in *Proceedings of the 6th ACM Workshop on Moving Target Defense*, 2019, pp. 79–88.

[6] Ö. Gözüaçık, A. Büyükçakır, H. Bonab, and F. Can, "Unsupervised Concept Drift Detection with a Discriminative Classifier," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 2365–2368.

[7] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.

[8] V. Carela-Español, P. Barlet-Ros, A. Bifet, and K. Fukuda, "A streaming flow-based technique for traffic classification applied to 12+ 1 years of internet traffic," *Telecommunication Systems*, vol. 63, no. 2, pp. 191–204, 2016.

[9] P. Barthakur, M. Dahal, and M. K. Ghose, "Clusibothealer: botnet detection through similarity analysis of clusters," *Journal of Advances in Computer Networks*, vol. 3, no. 1, 2015.

[10] M. Yahyazadeh and M. Abadi, "Botgrab: A negative reputation system for botnet detection," *Computers & Electrical Engineering*, vol. 41, pp. 68–85, 2015.

[11] E. R. Faria, J. a. Gama, and A. C. P. L. F. Carvalho, "Novelty detection algorithm for data streams multi-class problems," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, ser. SAC '13, 2013, p. 795–800.

[12] T. S. Sethi, M. Kantardzic, and H. Hu, "A grid density based framework for classifying streaming data in the presence of concept drift," *Journal of Intelligent Information Systems*, vol. 46, no. 1, pp. 179–211, 2016.

[13] L. I. Kuncheva and W. J. Faithfull, "PCA feature extraction for change detection in multidimensional unlabeled data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 1, pp. 69–80, 2013.

[14] A. A. Qahtan, B. Alharbi, S. Wang, and X. Zhang, "A PCA-Based Change Detection Framework for Multidimensional Data Streams: Change Detection in Multidimensional Data Streams," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 935–944.

[15] E. B. Beigi, H. H. Jazi, N. Stakhanova, and A. A. Ghorbani, "Towards effective feature selection in machine learning-based botnet detection approaches," in *2014 IEEE Conference on Communications and Network Security*. IEEE, 2014, pp. 247–255.

[16] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1/2, pp. 100–115, 1954.

[17] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proceedings of the 2007 SIAM international conference on data mining*. SIAM, 2007, pp. 443–448.

[18] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, and J. Aguilar, "Towards the deployment of machine learning solutions in network traffic classification: a systematic survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1988–2014, 2018.

[19] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 2001, pp. 97–106.