# *Enhancing IoT Network Security with Concept Drift-Aware Unsupervised Threat Detection*

Article

Accepted version

V. Agate, A. De Paola, S. Drago, P. Ferraro, G. Lo Re

2024 IEEE Symposium on Computers and Communications (ISCC)

Publisher: IEEE

# Enhancing IoT Network Security with Concept Drift-Aware Unsupervised Threat Detection

Vincenzo Agate[†], Alessandra De Paola[†], Salvatore Drago[‡], Pierluca Ferraro[†] and Giuseppe Lo Re[†]
[†]Department of Engineering, University of Palermo, Palermo, Italy.
[‡]IMT School for Advanced Studies Lucca, Italy
Email: vincenzo.agate@unipa.it, alessandra.depaola@unipa.it, salvatore.drago@imtlucca.it,
pierluca.ferraro@unipa.it, giuseppe.lore@unipa.it

***Abstract*—The dynamic characteristics of Internet of Things (IoT) systems create major challenges for threat detection systems that rely on machine learning models. Over time, shifts in the statistical distribution of data can lead to drastic performance degradation. This phenomenon is known as concept drift. When this problem occurs, traditional static systems require human intervention to manually retrain, leaving the network vulnerable in the meantime. In this paper, we propose an unsupervised system for online detection of anomalous traffic generated by malware-infected IoT devices. The proposed multi-tier system explicitly accounts for concept drift, automatically retraining only when necessary. We thoroughly tested the system by performing an extensive experimental evaluation using the real-world IoT-23 dataset, which includes network traffic generated by IoT devices as well as malicious network traffic generated by devices infected with different types of malware. We also compared our approach with other state-of-the-art work, and the results showed the remarkable performance achieved by the system using key metrics such as F1 score, accuracy, false positive rate and false negative rate.**

***Index Terms*—Concept Drift, Online Threat Detection, IoT, Unsupervised Learning, Cybersecurity**

## I. Introduction

The amount of information collected by sensors and Internet-connected devices is growing fast with the recent and rapid adoption of Internet of Things (IoT) and related technologies, such as smart cities [1], smart homes, and industrial IoT. At the same time, efficient data analytics and machine learning techniques to make predictions and support decision-making are in high demand by private enterprises and governments. Meanwhile, there has been a dramatic increase in cyber-attacks that affect networks, computers, information systems, and IoT devices. This is evidenced by Kaspersky's 2021-22 report [2], which documents how the number of malware attacks targeting IoT devices increased by nearly 60% year-over-year. In addition, collected data are subject to numerous factors that inevitably reduce their quality and reliability: sensory data are affected by noise and, if they come from users' devices, could also be manipulated by selfish users [3]–[5]. Moreover, the data is also naturally subject to the phenomenon of concept drift [6], [7], i.e., its statistical properties may change unexpectedly over time, invalidating any previous training of machine learning systems and paving the way for a new category of errors and inaccuracies. For example, imagine a threat detection system that monitors network traffic from IoT devices which measure certain user health parameters, such as heart rate or blood oxygen levels. Such devices exchange alerts with each other and with fog nodes when any of these parameters deviate from the normal range for a healthy person. In the system's training phase, it learns to distinguish between benign traffic (normal health alerts) and suspicious traffic. Suspicious traffic could be identified as, for example, a device that repeatedly sends the same alert values outside the normal range, which could suggest a malware infection that is generating false readings. However, threat detection systems like this faced an unexpected challenge during the COVID-19 pandemic. Many users had persistently unusual, yet stable readings due to the virus, leading their devices to repeatedly issue identical alerts for a prolonged period. A threat detection system not calibrated for this type of scenario might mistakenly flag such devices as infected and their traffic as malicious. In reality, what occurred was a concept drift, the alteration of statistical properties of the target variable. This change required a retraining phase with the new data. Once the health emergency is over, a system like this should be able to adapt back to its original mode of operation, ideally without retraining again.

In this paper, we propose an online unsupervised anomaly-based threat detection system for IoT network environments which explicitly handles concept drifts. The architecture of the system is multi-tiered and employs drift detection modules and an ensemble of unsupervised models to detect anomalies. Our goal is to minimize the number of model re-training in case of concept drift, while still ensuring high accuracy. To this end, we propose to explicitly handle the phenomenon of *recurring* concept drift, which is inevitable in many real-world cases. Our system does not discard old models. On the contrary, it keeps them because they can be reused in the future in case of recurring drifts. The system will only train a new model if the new data cannot be handled by any of the previous models.

To validate our approach, we have performed a comprehensive evaluation of our system. This is done through a variety of tests on a recent real-world dataset containing data from actual IoT devices (IoT-23 dataset [8]). The system is also compared to other state-of-the-art work to demonstrate its effectiveness, and the results show that it is remarkably capable of maintaining high accuracy while being completely unsupervised and reducing the frequency of model retraining.

The key contributions of this paper are as follows: (1) we designed an innovative unsupervised anomaly-based threat detection system, which specifically addresses recurring concept drifts; (2) we modified several known static unsupervised anomaly detection techniques to enhance their efficiency in analyzing data streams in real time; (3) we worked towards minimizing the quantity of anomaly detection models that require re-training, while ensuring that their performance levels remain high; (4) we extensively validated our system with a real-world IoT dataset and compared to other state-of-the-art work to demonstrate its effectiveness.

The remainder of the paper is structured as follows. Section II discusses related work. Section III proposes our novel architecture for unsupervised threat detection with concept drift handling. Our experimental evaluation is presented in Section IV, while Section V draws our conclusions.

## II. Related Work

The IoT ecosystem consists of a wide range of Internet-connected smart devices such as home appliances, light bulbs, network cameras, and sensors that can contribute to the realization of complex intelligent environments [9], [10]. As mobile and IoT devices continue to grow in popularity, they are becoming increasingly targeted by attackers. Likewise, studies on this particular type of malware detection are increasing and becoming more specialized for IoT devices as opposed to detection systems for traditional computer networks [11], [12].

In environments that are constantly evolving, such as IoT networks, the distribution of input or output data may shift unpredictably over time. This change is referred to as concept drift [13], [14] and can originate from various sources. The first form of drift, often referred to as virtual drift, happens when there is a shift in the input data distribution without a corresponding change in the predictions. The second form of drift, known as actual drift, occurs when the distribution of the input data remains consistent but the predictions vary, leading to changes in the decision boundary. In practical scenarios, however, these two forms of drift are likely to occur together, which happens when both the input data distribution and the predictions change at the same time. Another commonly referenced categorization for concept drift scenarios relates to how the data distribution evolves over time. Typically, four potential types of drift are identified: (1) *sudden*, which occurs due to unexpected events; (2) *incremental*, which can result from continuous sensor degradation or user preferences that evolve over time; (3) *gradual*, where the new concept initially alternates with the previous one; (4) *recurring*, where previous concepts return cyclically.

Concept drift in this particular domain can be triggered by anomalous traffic produced by a new type of malware that the system has not yet encountered, or by a change in the distribution of benign traffic generated by the devices.

Although growing in number in recent years, papers that explicitly address both IoT malware detection and concept drift are not very common. In [15], the authors propose a system for detecting malware and concept drift in IoT sensor network traffic using statistical techniques. However, one of the limitations of this approach is the use of supervised techniques. The number of real data labels required by the system and the time frame in which these labels must be available is unrealistic in an online context. In fact, for the system to work properly, it would require constant labeling by a domain expert as new data comes in for analysis, at superhuman speeds. This problem is common to other work in the intrusion detection and concept drift literature, such as [16]. To overcome this limitation, the proposed system adopts an unsupervised approach to detect both concept drifts and anomalous traffic caused by malware in IoT devices. As regards unsupervised drift detection, IForestASD [17] evaluates incoming data in batches, employing a parameter that indicates the anomaly rate that should not be exceeded in these batches. However, in an unsupervised setting without additional data knowledge, selecting a suitable value for this parameter becomes challenging. The authors of [18] circumvent this problem by suggesting a number of strategies. Their final recommendation is to use the KSWIN [19] drift detection method on incoming data features to detect concept drift and then adjust the model before the evaluation stage.

The key shortcoming of these approaches is neglecting potential recurrence of concept drifts. In contrast to other works, our system is able to detect and handle recurring concept drifts, thereby minimizing the number of retraining phases. Furthermore, a common limitation of many anomaly detection systems discussed in the literature is their limited suitability for online applications. To address this issue, we have also adapted state-of-the-art anomaly detection techniques to improve real-time analysis of streaming data.

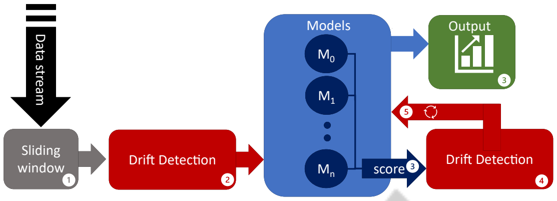

Figure 1. Architecture of the proposed system.

## III. System Architecture

This section outlines the multi-layer design of our online unsupervised threat detection system, which is shown in Fig. 1. The system is based on two concept drift (CD) detection modules and an ensemble of anomaly detection (AD) models. One of these models is currently in use, while the rest encapsulate the history of past AD models, and can be reused when recurring concept drift occurs. This is possible because the system has the ability to reconfigure itself by independently selecting which models to use depending on the context [20]. The proposed system can be deployed to a server that monitors the traffic flow of an IoT network and tries to detect suspicious events generated by infected devices. Working directly on data streams poses additional challenges that must be addressed. These include time and memory constraints, and the aforementioned concept drift management. To address time constraints, the data are analyzed sequentially. This approach minimizes the amount of time elapsed from the receipt of new data to its evaluation and allows the system to make its predictions on new samples that have not yet been used for training. This strategy, called test-then-train, is known in the literature as Prequential Evaluation [21].

A fixed-size sliding window approach is used to deal with memory constraints, whereby only the last two batches of the most recent data are stored at any given time. Specifically, the system uses a sliding window (*W*) of fixed size *w* containing the data batches $B_{n-1}$ and $B_n$. It also uses a list called *ml* of fixed size *N*, which is composed of AD models. In addition to providing the predicted outcome, each AD model includes a measure of the confidence associated with that particular prediction [22]. We use the term $CD_d$ to represent the concept drift detection module that operates on the input data, and $CD_m$ to signify the module that works on the confidence level of the model's predictions. Let us define *M* as the current model and $cl_{m_i}$ as the confidence list for each model $m_i$. All the $cl_{m_i}$ confidences are inserted in a list called *lcl*.

The workflow of the system is shown in Fig. 1 and detailed step by step in Algorithm 1. Some utility variables are initialized in lines 1-3. Specifically, *nc* is used to check whether a new batch of data has been received. *bf*, *idf*, and *cdf* are boolean variables that report whether the system has received the first batch of data, whether drift has been detected in the input data features, and whether drift has been detected between the confidence lists of the model training data and the confidence list of the last batch of data, respectively.

In lines 4-9, according to the prequential evaluation strategy, new input data is evaluated immediately. The predictions are added to the list of predictions of the $\hat{Y}$ system. In lines 10-13, each incoming value is added to the list *W* using the sliding window mechanism; *nc* is also updated. If a new batch of data has not yet been evaluated and saved (line 14), the workflow resumes from line 4. Otherwise, two possible scenarios may occur. If this is the first batch of data ($B_0$), the first model *m* with that batch of data is created and trained in lines 16-21; then, *m* is added to *ml*, its confidence list on the training data is added to *lcl*, and finally *m* is set as the current model. Alternatively, if this is not the first batch of data ($B_n$), $CD_d$ checks for concept drift between $B_{n-1}$ and $B_n$ in lines 23-25. If no drift is detected, *nc* is reset and the workflow continues from line 4. If any drift is detected, the confidence list of the batch that caused the drift is extracted for each model $m_i$, and it is then compared with the confidence list of the model training data saved in *lcl* (lines 27-30). If one of the models does not cause drift (lines 31-33), it is set as the current model, *M*, after which *nc* is reset and the workflow continues from line 4. Conversely, if no model is suitable to evaluate the new concept, a new model *m* is created and trained from the last batch of data (lines 34-41). If *ml* has not reached its maximum size *N*, *m* is added to *ml*, and its confidence list on the training data is added to *lcl*; otherwise, the first model is discarded from *ml* and its confidence list is removed from *lcl* before proceeding with these operations. Finally, *m* is set as the current model (*M*). To implement both concept drift detection modules ($CD_d$ and $CD_m$), we adopt the KSWIN [19] method. KSWIN utilizes the principles of the Kolmogorov-Smirnov (KS) statistical test [23]. This is a non-parametric test which functions without the need for presuming anything about the data distribution from which the samples are drawn. However, this test is restricted to dealing with one-dimensional data, as it works by calculating the absolute difference between the distributions of two sets of one-dimensional data.

**Algorithm 1** System workflow

```
Require: $\Delta$: data-stream; $N$: max number of models in ml;
    $w$: max length of $W$; mt: model type;
    $CD_d$: concept drift detection module for input data;
    $CD_m$: concept drift detection module for model confidence.
Ensure: $\hat{Y}$ : the list of system predictions.
 1: $\hat{Y} \leftarrow []$; $W \leftarrow []$; ml$\leftarrow []$; lcl $\leftarrow []$; nc$\leftarrow 0$
 2: bf $\leftarrow$ True; idf $\leftarrow$ False; cdf $\leftarrow$ True; $M \leftarrow$ None
 3: for $x \in \Delta$ do
 4:   if $M \neq$ None then
 5:     $\hat{y} \leftarrow M$.predict($x$)
 6:   else
 7:     $\hat{y} \leftarrow 0$
 8:   $\hat{Y}$.append($\hat{y}$)
 9:   if len($W$) == $w$ then
10:     $W$.remove_head()
11:   $W$.append($x$) ; nc $\leftarrow$ nc+1
12:   if nc $\geq \lceil w/2 \rceil$ then
13:     if bf == True then
14:       $m \leftarrow$ new model of type mt ; $m$.partial_fit($W$)
15:       $cl_m \leftarrow m$.predict_confidence($W$)
16:       ml.append($m$) ; lcl.append($cl_m$)
17:       $M \leftarrow m$ ; bf = False
18:     else
19:       $B_{n-1} \leftarrow W[0 : \lfloor w/2 \rfloor]$ ; $B_n \leftarrow W[\lceil w/2 \rceil : w]$
20:       idf $\leftarrow CD_d$.detect_drift($B_{n-1}$,$B_n$)
21:       if idf == True then
22:         for $m_i \in$ ml do
23:           $cl_{B_n} \leftarrow m_i$.predict_confidence($B_n$)
24:           $cl_{m_i} \leftarrow$ lcl[i]
25:           cdf $\leftarrow CD_m$.detect_drift($cl_{m_i}$,$cl_{B_n}$)
26:           if cdf == False then
27:             $M \leftarrow m_i$ ; break
28:         if cdf == True then
29:           $m \leftarrow$ new model of type mt ; $m$.partial_fit($B_n$)
30:           $cl_m \leftarrow m$.predict_confidence($B_n$)
31:           if len(ml) == $N$ then
32:             ml.remove_head() ; lcl.remove_head()
33:           ml.append($m$) ; lcl.append($cl_m$) ; $M \leftarrow m$
34:     nc $\leftarrow 0$
35: return $\hat{Y}$
```

As our work involves multiple features, it is crucial to modify the KSWIN method to fit our case, especially for $CD_d$. The workaround suggested in [18], [24] is to run the KSWIN method on each feature individually. However, this strategy has an inherent limitation: it is possible for small variations to occur simultaneously on many features, leading to concept drift. This drift may go undetected by a system that focuses on only one feature at a time. To tackle this issue, we have chosen to construct a new synthetic feature that considers the changes across all the other features simultaneously. The KSWIN method is then employed to examine the shifts in the

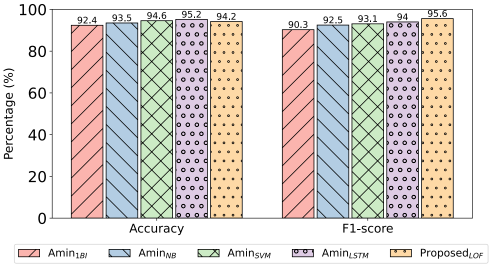

Figure 2. Accuracy and F1-score of the proposed system vs. Amin et al.

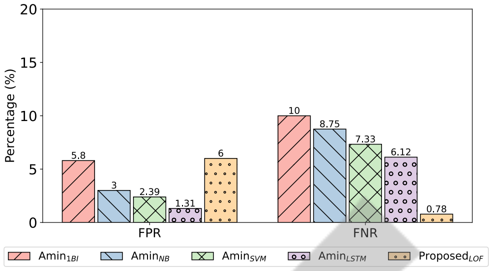

Figure 3. FPR and FNR of the proposed system vs. Amin et al.

distributions of all features, including the synthetic one. If a drift is observed in at least one feature, the system considers training a new model or restoring a previously trained model. For this study, we have experimentally selected the normalized sum of squares of the base features as our synthetic feature.

The anomaly detection module consists of an ensemble of unsupervised models. In this paper, the unsupervised anomaly detection models LOF, IF, OCSVM and ELEN (selected from the most widely used and studied models in the literature [25]), have been used and compared. The Local Outlier Factor (LOF) is an anomaly detection algorithm that primarily relies on density measurements, making it especially effective for handling datasets characterized by irregular distributions [26]. Isolation Forest (IF) is an unsupervised anomaly detection algorithm based on the assumption that anomalous data correspond to points that are rare and far from the center of clusters of normal data [27]. One Class Support Vector Machine (OCSVM) [28] derives a function that yields positive values in areas with high point densities, and negative values when the densities are low. Elliptic Envelope (ELEN) [29] constructs a hypothetical elliptical boundary around a specified dataset. Any values within this envelope are deemed as standard data, conversely, other values are classified as outliers.

These are static methods designed to operate when the entire dataset is available or when a statistically significant train set has been collected. Many approaches have been proposed in the literature to use these methods with streaming data, but they all present shortcomings and are not suitable for handling data with concept drifts, as demonstrated in [7].

Our system addresses these challenges by adopting a hybrid strategy. We assume that, for each batch of data used for retraining, the frequency of anomalies is relatively low compared to the overall set of observations. In this context, unsupervised models can be employed to learn the statistical distribution of normal traffic and identify malicious traffic that deviates from this distribution. However, a shift in data distribution due to concept drift can lead to a degradation in system performance, as shown in Fig. 6 and later discussed in Sec. IV. To avoid this, the proposed system proactively tackles concept drift before the anomaly detection algorithm comes into play, using specially designed detection modules. In addition, the system retains a historical record of models to dramatically reduce the need to train new ones. The quantity of models retained in the historical record carries significant weight: a larger pool of models increases the likelihood that one will be suitable for the input data, which significantly cuts down on the number of models requiring training. This is particularly advantageous in scenarios involving recurring concept drift. For the proposed system, the computational complexity of the $CD_d$ module is *O(w)* and, in case of drift detection, the complexity of the $CD_m$ module results in $N\cdot$ *(O(w) + CCP(w))* cost, where *CCP* represents the computational cost of model prediction, which is generally much lower than the computational cost of model training.

## IV. Experimental Evaluation

This section will detail the performance of the proposed system, presenting a series of tests performed on real-world datasets and comparing our approach with state-of-the-art work to test its validity. The system is evaluated with the main metrics used in the literature, namely accuracy, F1 score, False Positive Rate (FPR) and False Negative Rate (FNR).

The experiments reported below have been performed on datasets extracted from IoT-2023 [8], which is a collection of multiple datasets that provide a wide variety of network traffic data from actual IoT devices, both uninfected and infected with various malware. Each experiment was performed with the maximum number of stored models in the history set to 5. All experiments were repeated for each of the unsupervised models listed earlier in Section III, while varying the relevant hyperparameters and the window size *w* (from a minimum of 25 to a maximum of 300). Furthermore, the experiments were repeated with and without the concept drift detection to test their actual effectiveness. The first set of experiments we present concerns the comparison between the model that obtained the best results with our proposed system and four different models described in [15]. This work was chosen for comparison due to its recency and relevance. It provides a comprehensive set of evaluation metrics for assessing the performance of an online system in the presence of concept drift and uses the IoT-2023 dataset, as we do. To perform our experiments coherently with the results reported in [15], we used a dataset created by merging four sub-datasets of IoT-2023, with 5075 records. 2.5% of these records belong to the malicious class. All attacks encountered in the experiments are treated as unknown, similar to zero-day attacks.

For the setting under consideration, the anomaly detection model that achieved the best performance for our system is the LOF with window size *w=200*. Fig. 2 and 3 show the comparison between this unsupervised system and the four

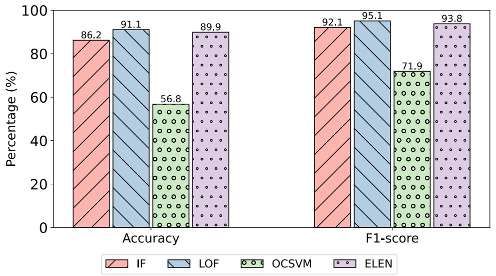

Figure 4. Accuracy and F1-score of anomaly-detection models for our system in a dataset with severe concept drift.

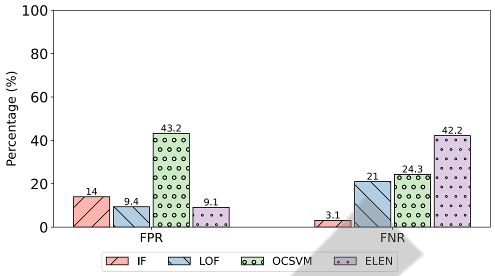

Figure 5. FPR and FNR of anomaly-detection models for our system in a dataset with severe concept drift.

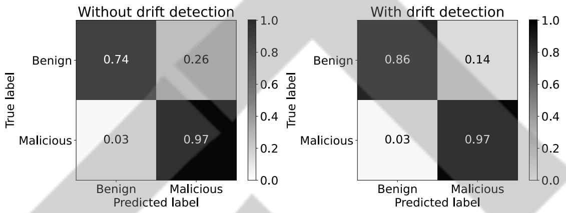

Figure 6. Confusion matrices of our best system (IF) without and with concept drift detection.

different supervised models presented in [15]. Please note that the y-axis range of Figure 3 was adjusted to 0-20 instead of 0-100 to enhance the visual representation of the results.

The proposed system, although using an unsupervised model, achieves an accuracy of 94.2%, surpassing two of the compared supervised models and obtaining results very close to Amin$_{\mathrm{SVM}}$ and Amin$_{\mathrm{LSTM}}$. Remarkable results were also obtained by our system with respect to the F1 score (95.6%), outperforming all other systems. The same is true for the FNR (in this case, the lower the better), where the proposed system achieves 0.78%, outperforming all other systems. In contrast, such accurate detection of the malicious class results in a slightly higher value for the FPR, where the system achieves 6%. This trade-off seems inevitable, and can be attributed to the unsupervised nature of the model. The dataset used in this comparison presents only a moderate amount of drift. Nevertheless, the proposed system has proven to be remarkably robust even with little to no drift, resulting in high performance under all circumstances. We conducted a second set of experiments comparing the four types of unsupervised models proposed for our system in Section III. This set of experiments was performed on a modified version of the previous dataset, which was obtained by merging it with additional sub-datasets, also from the IoT-2023 dataset collection. It explicitly presents both sudden and recurring concept drift in its data distribution, caused by changes in the network structure, the types of IoT devices in use, and the emergence of new malicious traffic from novel malware. This scenario is both realistic and suitable for further testing the system under more challenging conditions and evaluating the need for concept drift detection and adaptation mechanisms [30]. Fig. 4 and Fig. 5 show the results obtained by each model in our system with drift detection enabled, when using the best possible parameters for each model. The two models which achieved the best results are LOF and IF (both with window size *w=75*). IF obtained 86.2% accuracy and 92.1% F1 score with a low FNR of 3.1% and a relatively low FPR of 14.0%, which is still remarkable, considering that the modified dataset contains a considerable amount of drift, which poses a major challenge to anomaly detection systems. LOF, on the other hand, obtains 91.1%, 95.1%, 21%, 9.4% for accuracy, F1 score, FNR, and FPR, respectively. Although accuracy and F1 score are higher for LOF, the low FNR obtained by IF makes it the most reliable system, in the considered scenario, since false negatives must be avoided at all costs. The other two models tested (OCSVM and ELEN) exhibit low performance in the detection of malicious and benign classes, respectively, as can be seen from the FPR and FNR values obtained. Interestingly, for the dataset with moderate presence of drift, performance is better with a large window. The large window is more representative but slower to detect drift than the small window, which is less representative but reduces the time between the occurrence of concept drift and its detection and adaptation. A small window is preferable when the dataset has a strong presence of drift. Finally, Fig. 6 shows the confusion matrices obtained by IF without drift detection (on the left) and with drift detection (on the right). For the system without concept drift detection, the appearance of new malicious traffic is not particularly harmful because its statistical distribution is still different from that of the benign traffic on which the model was originally trained, but concept drift on benign traffic is particularly harmful because it causes misclassifications that result in high FPR. Unsurprisingly, given that the dataset contains severe concept drift, explicit drift management results in significantly improved performance, especially for the benign class. By automatically adapting to new benign network traffic coming from a new type of IoT device, the system avoids mistaking it for malicious traffic, in contrast to the system without drift detection.

The entire dataset was processed in 83 batches of size *w=75*. Concept drift in the incoming data was detected 24 times. In such cases, the system reused an existing model 75% of the time. This confirms the importance of handling recurring concept drift, which allowed the system to avoid unnecessary retraining when an already known concept occurred.

## V. CONCLUSIONS

In this work, we introduced a novel system that fuses unsupervised anomaly detection and concept drift detection methodologies, employing traditionally static algorithms in a hybrid manner to effectively work with streaming data. Moreover, we devised a unique approach to expressly handle recurring concept drift, by retaining a historical record of previous models which drastically reduces the necessity for re-training. To verify the correctness of our methodology, we performed an extensive evaluation of our system through a series of experiments on two real-world datasets. The results prove the effectiveness of the proposed system in identifying malicious traffic generated by malware, regardless of the presence or absence of concept drift, by always achieving high performance. These results are comparable and sometimes superior to supervised systems in the literature. However, the proposed system is much more realistic than its supervised counterparts, which often make assumptions that are not feasible in practice in terms of obtaining instantaneous feedback on previous data provided by experts who are supposed to work at inhuman speeds. As future work, the system could be extended to dynamically use data windows of variable size, in order to exploit the trade-off between fast response to drift and representativeness of the training set discussed in Section IV.

## Acknowledgments

This work is partially funded by the European Union Next-Generation EU (Piano Nazionale di Ripresa e Resilienza (PNRR) – Missione 4 Componente 2, Investimento 1.3) – D.D. 1551.11–10-2022, PE00000004) - MICS (Made in Italy – Circular and Sustainable) Extended Partnership, partially funded by the European Union - PON Ricerca e Innovazione 2014-2020 - DM 1062/2021, and partially funded by the European Union Next-Generation EU (Piano Nazionale di Ripresa e Resilienza (PNRR) – Missione 4 Componente 2, Investimento 1.3) – Progetto ”Future Artificial Intelligence - FAIR” PE00000013, CUP J73C24000060007.

## References

[1] V. Agate, A. De Paola, G. Lo Re, and A. Virga, “Reliable reputation-based event detection in v2v networks,” in *International Conference on Advanced Research in Technologies, Information, Innovation and Sustainability*. Springer, 2023, pp. 267–281.

[2] Kaspersky, “Pushing the limits: How to address specific cybersecurity demands and protect IoT,” *Kaspersky*, 2022.

[3] F. Concone, P. Ferraro, and G. L. Re, “Towards a smart campus through participatory sensing,” in *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE, 2018, pp. 393–398.

[4] F. Restuccia, P. Ferraro, S. Silvestri, S. K. Das, and G. Lo Re, “IncentMe: Effective mechanism design to stimulate crowdsensing participants with uncertain mobility,” *IEEE Transactions on Mobile Computing*, vol. 18, no. 7, pp. 1571–1584, 2018.

[5] V. Agate, A. De Paola, G. Lo Re, and M. Morana, “Vulnerability evaluation of distributed reputation management systems,” in *Proceedings of the 10th EAI International Conference on Performance Evaluation Methodologies and Tools*, ser. VALUETOOLS’16, 2017, p. 235–242.

[6] F. Bayram, B. S. Ahmed, and A. Kassler, “From concept drift to model degradation: An overview on performance-aware drift detectors,” *Knowledge-Based Systems*, vol. 245, p. 108632, 2022.

[7] V. Agate, S. Drago, P. Ferraro, and G. Lo Re, “Anomaly detection for reoccurring concept drift in smart environments,” in *18th International Conference on Mobility, Sensing and Networking (MSN)*. IEEE, 2022, pp. 113–120.

[8] S. Garcia, A. Parmisano, and M. J. Erquiaga, “IoT-23: A labeled dataset with malicious and benign IoT network traffic (1.0.0) [Data set]. Zenodo.” [Online]. Available: https://doi.org/10.5281/zenodo.4743746

[9] A. De Paola, P. Ferraro, G. Lo Re, M. Morana, and M. Ortolani, “A fog-based hybrid intelligent system for energy saving in smart buildings,” *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 7, pp. 2793–2807, 2020.

[10] F. Concone, G. L. Re, and M. Morana, “A fog-based application for human activity recognition using personal smart devices,” *ACM Transactions on Internet Technology (TOIT)*, vol. 19, no. 2, pp. 1–20, 2019.

[11] V. Agate, F. M. D’Anna, A. De Paola, P. Ferraro, G. Lo Re, and M. Morana, “A behavior-based intrusion detection system using ensemble learning techniques.” in *CEUR Workshop Proceedings, 6th Italian Conference on Cybersecurity, ITASEC 2022*, vol. 3260, 2022, pp. 207–218.

[12] Ö. A. Aslan and R. Samet, “A comprehensive review on malware detection approaches,” *IEEE Access*, vol. 8, pp. 6249–6271, 2020.

[13] 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.

[14] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, “Learning under concept drift: A review,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2346–2363, 2018.

[15] M. Amin, F. Al-Obeidat, A. Tubaishat, B. Shah, S. Anwar, and T. A. Tanveer, “Cyber security and beyond: Detecting malware and concept drift in ai-based sensor data streams using statistical techniques,” *Computers and Electrical Engineering*, vol. 108, p. 108702, 2023.

[16] D. Mulimani, S. G. Totad, P. Patil, and S. V. Seeri, “Adaptive ensemble learning with concept drift detection for intrusion detection,” in *Data Engineering and Intelligent Computing: Proceedings of ICICC 2020*. Springer, 2021, pp. 331–339.

[17] Z. Ding and M. Fei, “An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window,” *IFAC Proceedings Volumes*, vol. 46, no. 20, pp. 12–17, 2013.

[18] M. U. Togbe, Y. Chabchoub, A. Boly, M. Barry, R. Chiky, and M. Bahri, “Anomalies detection using isolation in concept-drifting data streams,” *Computers*, vol. 10, no. 1, p. 13, 2021.

[19] C. Raab, M. Heusinger, and F.-M. Schleif, “Reactive soft prototype computing for concept drift streams,” *Neurocomputing*, vol. 416, pp. 340–351, 2020.

[20] V. Agate, P. Ferraro, and S. Gaglio, “A cognitive architecture for ambient intelligence systems,” in *6th International Workshop on Artificial Intelligence and Cognition, AIC 2018. CEUR Workshop Proceedings*, vol. 2418, 2019, p. 52 – 58.

[21] J. Vinagre, A. M. Jorge, C. Rocha, and J. Gama, “Statistically robust evaluation of stream-based recommender systems,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 7, pp. 2971–2982, 2019.

[22] L. Perini, V. Vercruyssen, and J. Davis, “Quantifying the confidence of anomaly detectors in their example-wise predictions,” in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2020, pp. 227–243.

[23] F. J. Massey Jr, “The Kolmogorov-Smirnov test for goodness of fit,” *Journal of the American statistical Association*, vol. 46, no. 253, pp. 68–78, 1951.

[24] A. De Paola, S. Drago, P. Ferraro, and G. Lo Re, “Detecting zero-day attacks under concept drift: An online unsupervised threat detection system,” in *CEUR Workshop Proceedings, 8th Italian Conference on Cybersecurity, ITASEC 2024*, 2024.

[25] S. Nõmm and H. Bahşi, “Unsupervised anomaly based botnet detection in iot networks,” in *2018 17th IEEE international conference on machine learning and applications (ICMLA)*. IEEE, 2018, pp. 1048–1053.

[26] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, “LOF: identifying density-based local outliers,” in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, pp. 93–104.

[27] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation forest,” in *2008 eighth ieee international conference on data mining*. IEEE, 2008, pp. 413–422.

[28] A. J. Smola and B. Schölkopf, “A tutorial on support vector regression,” *Statistics and computing*, vol. 14, pp. 199–222, 2004.

[29] P. J. Rousseeuw and K. V. Driessen, “A fast algorithm for the minimum covariance determinant estimator,” *Technometrics*, vol. 41, no. 3, pp. 212–223, 1999.

[30] J. Wang, P. Li, E. Weitkamp, Y. Satani, and A. Omundsen, “Malbuster: Scalable, real-time, and concept drift-adaptive malware detection for smart environments,” in *2024 IEEE 21st Consumer Communications & Networking Conference (CCNC)*. IEEE, 2024, pp. 352–355.