Real-Time Packet-Based Intrusion Detection on Edge Devices

Niccolò Borgioli
Scuola Superiore Sant'Anna
Pisa, Italy

Linh Thi Xuan Phan
University of Pennsylvania
Philadelphia, Pennsylvania, USA

Federico Aromolo
Scuola Superiore Sant'Anna
Pisa, Italy

Alessandro Biondi
Scuola Superiore Sant'Anna
Pisa, Italy

Giorgio C. Buttazzo
Scuola Superiore Sant'Anna
Pisa, Italy

ABSTRACT

Recently, the number of security threats targeting cyber-physical systems has continued to increase, both in quantity and in sophistication. Modern signature-based Intrusion Detection Systems (IDSs) are no longer able to keep up to date with the most recent attack techniques. This gives rise to the need for an intelligent system that is able to learn the expected network traffic and to detect not only known but also novel attacks. This paper introduces a novel autoencoder-based IDS that can detect new malicious packets with high precision. The proposed technique is general and can be used to detect a wide range of attacks, including unseen ones. Extensive experiments in simulation and on real hardware show that our technique substantially outperforms state-of-the-art solutions in terms of detection accuracy and generality. An analysis of the inference times is presented to show the predictability of the detection mechanism, as well as its practical applicability in resource-constrained edge devices.

KEYWORDS

real-time, anomaly detection, unsupervised learning, autoencoder, network traffic

ACM Reference Format:
Niccolò Borgioli, Linh Thi Xuan Phan, Federico Aromolo, Alessandro Biondi, and Giorgio C. Buttazzo. 2023. Real-Time Packet-Based Intrusion Detection on Edge Devices. In *Cyber-Physical Systems and Internet of Things Week 2023 (CPS-IoT Week Workshops '23), May 09–12, 2023, San Antonio, TX, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3576914.3587551

1 INTRODUCTION

As the reliance on networked systems in cyber-physical systems (CPS) continues to increase, so does the number of cyberattacks targeting these systems. One of the most critical issues faced by network security is the ability to detect and respond to these attacks in real time. Intrusion detection systems (IDSs) play a crucial role in identifying network anomalies that may indicate the presence of an attack. However, traditional IDS methods that rely on predefined rules or signatures can be bypassed by attackers who use new or modified attack methods.

The state of the art in network anomaly detection mainly relies on supervised learning, where a classifier is trained to detect a predefined set of known attacks. However, similar to traditional IDSs, supervised learning requires a continuous retraining of such classifiers as new attack methods are discovered. One effective solution to this issue is to use *unsupervised* learning, which can learn the expected behavior and thus recognize any kind of anomalies. This approach has shown its potential in the predictive maintenance field, where it is able to identify sub-system failures based solely on the learned knowledge of the normal operation of a system. Recent research has also begun to investigate unsupervised learning techniques for detecting anomalous network flows based on manually extracted features [3, 12], with promising results.

In this paper, we exploit unsupervised learning to develop a new generation of IDSs that are capable of detecting general and new kinds of network attacks. The idea is to utilize modern autoencoder techniques to reconstruct the received packets and detect anomalies and attack packets based on the reconstruction error. Towards this, we first present two solutions based on state-of-art autoencoder (AE) architectures widely used with sequences. Based on the results, we then introduce Multi-State Memory AE (MSM AE), a novel AE architecture that exploits multiple parallel LSTM autoencoders, each with a different embedding size, to increase detection accuracy.

To evaluate the effectiveness of the proposed architectures, an extensive experimental evaluation was conducted on a dataset of real-world network traffic. The results demonstrate that the proposed architectures outperform existing methods in terms of detecting a variety of common attack types. The proposed architecture's ability to detect network attacks in an unsupervised manner, without the need for labeled data, makes it a valuable tool for safeguarding against cyber threats. We further present an experimental study of the inference time of the proposed solutions on real hardware; this serves as a foundation for worst-case execution time (WCET) analysis, which is crucial to providing predictable timing on resource constrained real-time systems. The proposed approach aims at substituting modern firewalls, while providing predictable timing guarantees on the delay introduced in the communication.

In summary, the paper makes the following contributions:

- a novel method to detect network attacks based on unsupervised learning;
- a novel autoencoder architecture for IDSs based on parallel autoencoders to improve attack detection accuracy;
- an analysis of the classification results in terms of true positives and false negatives;

CPS-IoT Week Workshops '23, May 09–12, 2023, San Antonio, TX, USA

ACM ISBN 979-8-4007-0049-1/23/05…$15.00
https://doi.org/10.1145/3576914.3587551

- an implementation of the proposed techniques on a real edge platform; and
- an evaluation of the accuracy and inference time on our experimental platform under different realistic configurations.

2 RELATED WORK

There exist several techniques for network anomaly detection; however, the majority of existing work focuses on packet flow analysis rather than per-packet analysis. Andreas et al. [2] presented a general comparison between packet-based and flow-based anomaly detection. The authors used a Bidirectional LSTM (BLSTM) classifier for both per-packet extracted features and flow-based features. Their experiments showed that the proposed architecture achieved 76% of accuracy with the per-packet features and 96% with flow-based features. Dromard et al. [5] proposed a custom algorithm to extract features from flows to recognize different anomaly classes based on some signatures, and obtained a classification accuracy of 93%. These techniques only target known attacks which the method learned to detect.

Considering the ability to detect new (and zero-day) attacks, Tram et al. [12] used Generative Adversarial Networks (GANs) to generate a reconstruction of the input traffic flows and used the reconstruction error to discriminate between normal and anomalous traffic. Although the approach only reached 82% accuracy, thanks to the unsupervised approach, it is also capable of detecting unknown attacks and thus greatly improve the portability of the IDS without the need for periodic retraining. These existing techniques, however, focus only on the detection accuracy, while ignoring the inference time of the detection.

Some recent solutions have begun to investigate the relevance of inference time in anomaly detection. For example, Kathareios et al. [7] developed a multi-stage solution for anomaly detection that considers inference time in their evaluation. They first extracted statistical features about flow and global traffic to be used as input to an autoencoder. The features reconstruction error is then used to detect anomalous flows. To reduce the percentage of false positives, the authors introduced a second stage classifier which analyzes the detected anomalous flows. The proposed method achieved a true positive rate between 90% and 98%, with a 2% rate of false positives. In their work, the authors also reported the inference time of their method, which requires at least 1 second for each prediction. Alam et al [1] proposed a memristor-based autoencoder to perform anomaly detection in low-power devices. They first trained the autoencoder offline and reached a detection accuracy of 95%. Then, they converted it for execution in the memristor on the low-power platform reducing the accuracy to 92%. They also analyzed the impact of the proposed solution in terms of power consumption and inference time under different working configurations. Finally, Carrera et al. [3] proposed to combine multiple unsupervised approaches to perform anomaly detection. The authors performed an extensive evaluation of their proposed solutions, which achieve an accuracy ranging from 80% to 90% depending on the different model combinations. Their solutions have an inference time of up to 1.18 seconds on a desktop PC with an Intel Core i7-8665U processor, 16 GB of RAM, and the Microsoft Windows 10 Pro operating system. Such inference time is too large to be acceptable for most real-time systems on edge devices.

Overall, to the best of our knowledge, no previous work provided a per-packet anomaly detection method together with an inference time analysis performed on or suitable for a real edge device.

3 PROPOSED REAL-TIME IDS METHOD

This section provides an overview of the dataset, an exploratory data analysis, the data preprocessing, the model architectures, and the training of the models in our detection method.

3.1 Dataset

Towards a general IDS that is capable of detecting novel attacks in real time, we have evaluated several existing datasets for both training and testing our models. Since our goal is to develop a packet-based IDS to be deployed on each edge device of a network, we targeted a dataset that provides raw network packet captures (pcap files) collected from different network nodes. In our proposed approach, we use unsupervised learning to train the network using benign packets only so that it can detect anomalies in the presence of malicious packets. We considered the following existing datasets, which cover a range of representative use case scenarios for different environments: NSL-KDD [8], UNSW-NB15 [9], CIC-IDS2017 [10], and EDGE-IIOTSET [6].

NSL-KDD. This dataset is a rebalanced version of the widely used KDDCUP'99 dataset (which presented some statistical issues). However, this dataset still presents some limitations which prevented its adoption in the present work: **(i)** provides only flow features and not raw packets, **(ii)** the data contained in the dataset were generated more than two decades ago and are no longer representative of modern network traffic and attacks.

UNSW-NB15. This dataset contains about 100 GB of collected raw benign and malicious packet data generated in a controlled simulation environment. It provides both a labeled CSV file with the extracted flow features and the raw pcap files. However, this dataset provides only flow labels but not per-packet labels, making it unsuitable for our purposes.

CIC-IDS2017. This dataset contains raw packet captures of both benign and malicious packets collected in a controlled network over 5 days. The dataset is provided both in the form of a CSV file with labeled flow information and both as raw pcap files with indications of when attacks start and end for each day. However, this dataset also does not provide per-packet labeling, making it unsuitable for our purposes.

EDGE-IIOTSET. This dataset collects network traffic of a realistic network composed of more than 10 different types of IoT devices. The authors collected the traffic generated by each device in a separate pcap file and performed 14 different types of attacks. The dataset provides both the raw pcap files and the CSV files with the extracted features from each packet, along with its label.

Based on the observed characteristics of each of the considered datasets, we decided to adopt the EDGE-IIOTSET for the present work, not only because it is the only one that provides labeled raw packets, but also because it considers the widest range of device types along with one of the most recent set of attacks.

3.2 Exploratory data analysis

After selecting the dataset, we analyzed the dataset to understand the composition of its samples. We focused on the packet length distribution and the encryption protocol used because they affect the capability of generalization and the learning features.

Packet length distribution. Analyzing the distribution of the packet lengths we found that the maximum packet length is 1514 bytes (maximum size of traditional ethernet frames), but that *most of the packets exchanged by the devices are quite short* (below 100 bytes). Our analysis shows that there is a big imbalance in the dataset from the packet length point of view, due to the non-uniform distribution of packet lengths.

Encryption. To understand if the payload content of the packets could be used for classification, we analyzed the dataset to check if the exchanged packets used an encryption protocol (such as TLS or IPsec). This is important because an encrypted payload will require a dedicated feature extraction to gather significant information. From our analysis, we discovered that *all the traffic exchanged by nodes is unencrypted* (which is expected, since IoT devices typically do not provide support for encryption). This characteristic enables us to exploit the plain text payload information to gather information useful for classifying packets.

3.3 Data preprocessing

This section describes the preprocessing carried out on each data sample before training our classifier.

The unbalanced distribution of packet lenghts can cause the proposed model to overfit short (and most occurring) packets and underfit the long ones, thus reducing its generalization capability. To prevent this issue, we assign each packet a weight computed as the inverse of the number of packets in the training set with the same length as the considered packet, as follows:

$$weight(packet) = \frac{1}{num\_pkts(len(packet))}. \quad (1)$$

With this technique, the data loader will even out the distribution of packets used in the training phase based on their lengths.

Since Ethernet packets in the dataset are unencrypted we can fully exploit the information contained in the payload. Packets are provided as byte sequences, so, before feeding them to the classifier, we must encode them in an interpretable format. Because the packet length is highly variable and we want the model to learn the important features of each packet, we encode each byte as a floating-point value from 0 to 1, by dividing the 8-bit unsigned integer representation of each byte by 255.

3.4 Model architectures

Autoencoders are a class of unsupervised neural networks which are composed of two main components: an encoder and a decoder. The encoder reduces the dimensionality of the incoming data and produces a compressed representation (encoding). The decoder performs the opposite operation by reconstructing the original data with little or no error starting from the encoding. This way, the autoencoder learns to automatically extract relevant features from the input and how to use them to reconstruct the original input.

Key idea. Traditional approaches make use of binary classifiers trained with labeled benign and malicious packets; however, their main drawback is the low accuracy in the classification of new attacks, thus requiring continuous retraining of the network to keep it up to date. Our approach aims at overcoming this limitation by *training the autoencoder with only benign packets*. This way, the autoencoder will have low reconstruction error on benign packets and high error on malicious (anomalous) ones. Thus, by analyzing the reconstruction loss of the packets, it is possible to detect malicious ones (including zero-day attacks) without the need for retraining. Specifically, if X is the input sequence of length N and Y is the reconstructed one, the reconstruction loss is defined as:

$$loss(X, Y) = \sum_{i=1}^{N} |x_i - y_i|, \quad (2)$$

where x_i and y_i are the i-th elements of the X and Y sequences, respectively. Since in the considered scenario packets are unencrypted, we fed the network the whole Ethernet frame (X).

In the following, we first explore two common autoencoder architectures: 1D Convolutional Neural Network Autoencoder (1D CNN AE), and Long Short Term Memory Autoencoder (LSTM AE). We selected these architectures because both of them have shown great performance when dealing with sequences in other fields. We will then introduce an enhanced architecture that exploits parallelization to further improve detection capability.

1D CNN AE. 1D CNNs are widely used when dealing with sequences and time series, due to their advantages over conventional (2D) CNNs. In this kind of network, the filter slides along a single dimension to produce an output. Compared to traditional fully connected autoencoders, a 1D CNN reduces the number of parameters and thus the memory footprint and computational complexity of the network while improving its accuracy. In this work, we used this type of convolution networks to create an autoencoder.

LSTM AE. LSTM networks are a class of recurrent neural networks (RNN) specifically designed to deal with sequences. LSTM cells can be organized into an encoder-decoder architecture to support the reconstruction of variable-length sequences. Unlike convolutional networks, in this architecture, the encoder reads the input sequence step by step. After reading the whole sequence, the hidden state of the encoder represents the internal learned representation of the entire sequence. The hidden state is a fixed-length vector, which is then used as the initial hidden state of the decoder to reconstruct the original sequence.

Observation. While training the LSTM AE, we observed that choosing the proper hidden state size is crucial to allow the network to learn how to properly extract features from the input sequence. A larger hidden state helps the network when dealing with long sequences because it can better remember long-term dependencies between distant elements in the sequence. In contrast, a smaller hidden state can better handle short-term dependencies. Thus, if we can combine states of different sizes, we should be able to cover both long-term and short-term dependencies. Based on this insight, we introduce a novel architecture, called *Multi-State Memory AE (MSM AE)*, which leverages the advantage of having feature vectors

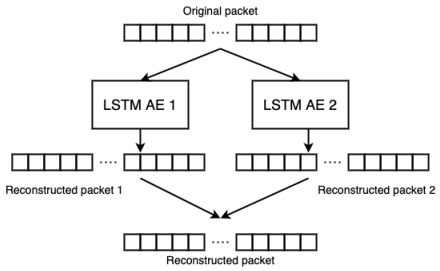

Figure 1: Architecture of the MSM AE packet flow

of different lengths to further improve the detection performance.

MSM AE. This new network combines the advantages of both a big and a small hidden state using multiple parallel LSTMs with different hidden sizes. Its design was inspired by the Inception [11] network architecture, which introduces parallel convolutional filters acting on the same input. Similar to the original Inception network, MSM AE contains multiple LSTM AE that receive as input the same sequence, which is then encoded and decoded independently by each autoencoder. The resulting reconstructions are then combined using an element-wise average operation to determine the final reconstructed sequence. In this work, we considered two parallel LSTM AEs: although adding more would probably further improve the detection accuracy thanks to a more granular analysis of the packets, this would also increase the computational complexity. Since the proposed solution aims to be deployed on resource-constrained edge devices, we decided to keep the smallest architecture which achieved a reasonable detection performance. Figure 1 illustrates the complete structure of the proposed MSM AE. To the best of our knowledge, this is the first approach to integrate parallel LSTM networks in the same network architecture, and it represents a major contribution of this work.

3.5 Training and hyperparameters selection

As explained earlier, our approach trains the autoencoders with only benign packets to enhance the generalization capability against novel malicious attacks. To do so, we divide the available benign packets into three groups: training set, validation set, and test set. For our experiments, we used the same sets for all the proposed architectures (1D CNN, LSTM AE, and MSM AE) to allow for a consistent performance evaluation of their performances.

During the training phase of each autoencoder, a hyperparameter exploration (kernel size, number of layers, hidden state size, ...) is performed to optimize the reconstruction loss of the network on the benign packets. Due to space limitations, we omit the details on the selection of hyperparameters.

After the training phase, we analyze the reconstruction loss distribution of each of the three proposed architectures for the benign packets and a small training subset of malicious packets of a specific category. Thanks to this analysis, it is possible to evaluate how well each network can distinguish between malicious and benign packets. Our approach detects an anomalous packet based on a tunable threshold value, selected based on this analysis. Generally, a small threshold value will make the detection more sensitive to malicious packets, but it can also misclassify some sporadic benign packets. On the contrary, a higher threshold value will reduce the misclassification of benign packets, but at the same time it will reduce the sensitivity of the detection of malicious packets. In a real-world deployment, the threshold should be tuned based on both the analysis of the reconstruction loss distribution and the specific application requirements. In this work, after training each network, we selected a detection threshold based on the reconstruction loss observed on a subset of benign and malicious (small portion of backdoor samples) packets.

4 IMPLEMENTATION AND EVALUATION

Prototype. To evaluate our approach, we built an implementation of our detection method based on the three architectures. For the training phase, we implemented the three networks using the PyTorch machine learning framework, and we performed the training using an Nvidia DGX server. To evaluate the performance in a real-world setting, we implemented the models in C++ for testing on an NVIDIA Jetson AGX Orin Developer Kit, using the libtorch library. We initialized the networks with the weights obtained from the training phase. Overall, our C++ implementation has approximately 200 LoCs, and our Python implementation has about 500 LoCs.

Evaluation. Our evaluation focuses on the two main objectives of our IDS design: **(i)** high detection accuracy of malicious packets, including novel unseen attacks; and **(ii)** predictable inference time that is compatible with practical deployments on edge devices. In the following, we report our experimental evaluation, which aims at evaluating two metrics: **(i)** the detection accuracy of the three proposed network architectures, and **(ii)** the inference timing analysis of the most accurate architecture on a real platform.

4.1 Detection accuracy

Each of the trained networks was tested with the associated threshold obtained at the end of the tuning phase. Tests were performed using all malicious packets of each attack type to analyze the generalization capability of each architecture against new attack types.

To evaluate the performance of each network, we used the labels of the samples to measure conventional performance metrics such as the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). Here, TP represents the number of malicious packets that are correctly detected, and FP represents the number of normal packets that are classified as malicious. TN (FN) can be defined similarly, but for normal (malicious) packets classified as normal packets.

Moreover, since the different classes of malicious packets and the benign packets are unbalanced, we also computed the True Positive Rate (TPR) and the False Positive Rate (FPR), as follows:

$$TPR = \frac{TP}{TP + FN}, \quad FPR = \frac{FP}{TN + FP}. \tag{3}$$

Table 1 reports the performance of the three proposed architectures in terms of TP (and TPR) and FP (and FPR), over all the malicious packet classes. (The dataset contains 5,919,987 malicious packets and 686,368 normal packets.)

The results in Table 1 demonstrate that the 1D CNN architecture is not suitable for detecting unknown malicious packets, since it is not capable of generalizing what was learned during training. It achieved low error on benign packet reconstruction but high error

Architecture	TP	TPR	FP	FPR
1D CNN	2,074,430	35.04%	126,895	18.49%
LSTM	5,633,294	95.16%	121,277	17.67%
MSM AE	5,909,716	99.83%	1,233	0.18%

Table 1: Comparison between normal and malicious packet classification for the three proposed architectures.

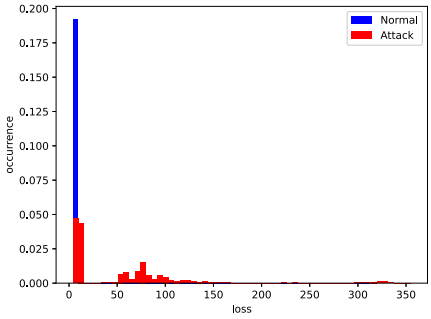

Figure 2: Normalized reconstruction loss distribution of benign and malicious training packets for the 1D CNN

	Single LSTM AE			MSM AE		
Attack	Norm.	Mal.	Prec.	Norm.	Mal.	Prec.
Backdoor	125	14823	99,16%	244	14704	98,37%
DDoS HTTP	596	136889	99,56%	449	137036	99,67%
DDoS ICMP	69155	1679458	96,04%	78	1748536	99,99%
DDoS TCP	30	1212061	99,99%	41	1212050	99,99%
DDoS UDP	195196	1734243	89,90%	78	1929361	99,99%
MITM	14	723	98,10%	205	532	72,18%
OS Fing.	79	626	88,79%	97	608	86,24%
Password	6704	625631	98,93%	1133	631202	99,82%
Port Scan	9628	4369	31,21%	6207	7790	55,65%
Ransom	159	6459	97,60%	406	6212	93,86%
SQL Inject.	371	30365	98,79%	23	30713	99,93%
Uploading	1851	20735	91,80%	439	22147	98,06%
Vuln. Scan	2247	157249	98,59%	221	159275	99,86%
XSS	538	9191	94,47%	650	9079	93,32%
TOTAL	286693	5633294	95,16%	10271	5909716	99,83%

Table 2: Detection performances of the LSTM AE and MSM AE architectures on different attacks classes

on the malicious ones. This problem is even more prominent if we look at the reconstruction loss distribution of benign and malicious packets used for tuning the threshold of that network (shown in Figure 2). Most of such malicious packets are reconstructed with a loss similar to one of the benign ones, making it difficult to properly select a suitable threshold to distinguish the two classes.

In contrast, the LSTM architecture achieved a good generalization capability, as it learned the pattern of the benign packets much better and was able to better distinguish such packets from the malicious ones. In fact, after tuning the threshold, this network was able to greatly increase the number of detected malicious packets. However, LSTM still has a relatively high FPR. This is mainly due to the similarity of some kinds of malicious packets to benign ones. From Table 2, we can see that when dealing with UDP DDOS, Port Scanning and OS Fingerprinting attacks, this network can still reconstruct the corresponding packets well. This is because such packets are quite short and very similar to the benign ones.

	CPU		GPU	
	30 W	EDP	30 W	EDP
Max	599 ms	375 ms	242 ms	78 ms
Median	317 ms	195 ms	125 ms	39 ms
Mean	314 ms	194 ms	135 ms	52 ms

Table 3: Inference time results of the LSTM architecture with different platform configurations

	CPU		GPU	
	30 W	EDP	30 W	EDP
Max	769 ms	490 ms	322 ms	121 ms
Median	401 ms	255 ms	170 ms	63 ms
Mean	402 ms	256 ms	172 ms	67 ms

Table 4: Inference time results of the MSM AE architecture with different platform configurations

The MSM AE, thanks to the multiple hidden sizes, manages to solve the limitations of the traditional LSTM AE by providing an improved detection performance also on these kinds of attacks, while greatly reducing the false positives. The results shown in Table 1 demonstrate how this network reached a detection accuracy close to 100%, with a very small number of misclassified normal packets. We selected the threshold to both minimize the FPR and FNR. Selecting a more stringent threshold could yield further improvements in the detection capabilities of such model and thus improve attack detection, but at the same time also increase the FPR. In a real environment, the designer should select the most suitable threshold value depending on the specific application.

4.2 Inference time analysis

After assessing the accuracy performance of the proposed architectures, we also assessed the inference time of the two LSTM models. We did not perform this evaluation on the 1D CNN architecture since its detection accuracy was unsuitable for our task.

To properly measure the inference time, we configured the test program to run with a high priority to minimize the interference potentially introduced by other services running on the same platform. Each measure was repeated 1 million times, considering different packet lengths taken from the dataset. Then, we evaluated such measures using three aggregated metrics: maximum inference time, median inference time (50th percentile), and mean inference time. Inference times were measured both with and without GPU acceleration, and with two different levels of power of the platform: 30 W and EDP (full power) [4].

Table 3 shows the inference time of the simple LSTM network trained before. The results clearly show that by increasing the power (and thus the core frequency), we can almost halve the inference time. Accelerating the network using the available GPU reduced the inference time by up to 80–90%. However, it is worth noting that even in the low-power configuration and without GPU acceleration, the proposed architecture can achieve an inference time comparable to the latency typically experienced by network packets.

The results reported in Table 4 show that the MSM AE architecture requires higher inference times. However, since the two LSTMs in the architecture work in parallel, such time can be reduced by means of proper code parallelization. Results of such profiling campaign determine the maximum inference time required to analyze a

Attack	Backdoor	MITM	OS Fing.	Pass. attack	Port Scanning	Ransomware	SQL Injection	Uploading	Vuln. Scan	XSS	TOTAL
Prec.	0,23%	66,21%	23,45%	0,22%	0,23%	0,23%	0,23%	0,23%	0,23%	0,23%	0,24%

Table 5: Detection performances of the previous work on attack classes not used for training

network packet. Thus, we are able to guarantee the maximum delay introduced by the proposed system into a real-world scenario.

4.3 Comparing with existing work

Finally, we compared the performance achieved by the proposed solutions with the one proposed by the authors of the EDGE-IIOTSET dataset. In their work, Mohamed et al. [6] trained both a binary and a multiclass classifier, using as input a set of features manually extracted from the packet flow.

Due to the nature of the solution proposed in this paper, we only compared the performance with the binary classifier. Based on the information provided in [6], we recreated the same model used by the authors and we trained it using the same hyperparameters. However, to compare the generalization capability of the two approaches to new unseen attacks, we changed the training dataset using only the set of benign samples and a subset of attack classes for the training phase of their network. After training, their network achieved excellent performance (99% accuracy) in detecting the attacks it was trained on, but it was almost never able to detect the unseen ones (close to 0% accuracy), classifying them as normal packets. Table 5 reports the TPR achieved by the reference architecture on attack flows not used for training. Since the reference work uses per-flow instead of per-packet detection, it is not possible to numerically compare the two detections, so we compare the achieved rates.[1]

The experimental results presented above confirm that an unsupervised approach like those proposed in this work is much more robust than a supervised one, since it is able to correctly identify new types of attacks as well as known attacks.

4.4 Discussion

Overall, the LSTM-based architectures achieved better accuracy with respect to the 1D CNN architectures in terms of distinguishing between normal and malicious packets. In particular, the proposed MSM AE architecture exhibited the strongest detection performance. Using multiple LSTM cells in parallel with different hidden state sizes allows us to extract more features from the input sequence while preserving a small number of parameters. As a result, the architecture is able to better learn the features of the benign packets, minimizing the reconstruction error and enabling better detection of malicious packets, including unseen ones.

When tested on known malicious attacks, the proposed LSTM-based detection solutions provided an accuracy comparable to that of existing work. However, when dealing with novel attacks, our solutions were able to maintain the same accuracy as with known attacks, whereas the existing approaches exhibited poor detection performance. These results confirm the importance of unsupervised learning for IDSs, given that it does not require continuous retraining to detect novel threats, unless the network traffic pattern changes significantly.

Finally, the inference time of the architectures was evaluated on a real platform. Inference time analysis is crucial to correctly deploy IDSs in a safety-critical practical setting, but was largely ignored in prior IDS research. The experiments showed promising results towards the application of the proposed techniques on resource-constrained devices. We note that these timing results were achieved without network quantization or other optimization techniques, which can bring substantial performance improvements. We plan to investigate such optimizations in future work.

[1]The work by Mohamed et al. [6] performs per-flow analysis, while our proposed solutions use a per-packet approach. Therefore, it is not meaningful to compare their inference times.

5 CONCLUSIONS

We have presented a novel dual autoencoder architecture for real-time intrusion detection based on anomalous packet identification for networked edge devices. Our experimental study demonstrated that unsupervised learning can greatly boost the performance of an IDS, allowing it to reliably detect not only trained but also novel attacks without the need to retrain the network. Although the classification capability of the considered traditional autoencoders was promising, the novel MSM AE architecture proposed in this work was shown to substantially improve performance, minimizing the false positives while achieving very low false negatives. Finally, the inference time analysis performed on a real edge device setting serves as a foundation for achieving predictable inference time, while also demonstrating the applicability of the proposed architecture for real deployments on resource-constrained devices.

ACKNOWLEDGMENTS

This work was supported in part by Huawei and the Italian Ministry of University and Research (MIUR), under the SPHERE project funded within the PRIN-2017 framework (grant no. 20172NNB4T_001), and NSF grants CNS-1750158, CNS-1955670, CNS-2111688 and CNS-1703936.

REFERENCES

[1] Md. Shahanur Alam, B. Rasitha Fernando, Yassine Jaoudi, Chris Yakopcic, Raqibul Hasan, Tarek M. Taha, and Guru Subramanyam. 2019. Memristor Based Autoencoder for Unsupervised Real-Time Network Intrusion and Anomaly Detection. In *Proceedings of the ICONS*.
[2] Brook Andreas, Jayaweera Dilruksha, and Eric McCandless. 2020. Flow-Based and Packet-Based Intrusion Detection Using BLSTM. In *SMU Data Science Review*.
[3] Francesco Carrera, Vincenzo Dentamaro, Stefano Galantucci, Andrea Iannacone, Donato Impedovo, and Giuseppe Pirlo. 2022. Combining Unsupervised Approaches for Near Real-Time Network Traffic Anomaly Detection. *Applied Sciences* 12, 3 (2022).
[4] NVIDIA Corporation. 2022. NVIDIA Jetson Orin - Tuning Power.
[5] Juliette Dromard, Gilles Roudière, and Philippe Owezarski. 2017. Online and Scalable Unsupervised Network Anomaly Detection Method. *IEEE Transactions on Network and Service Management* 14, 1 (2017), 34–47.
[6] Mohamed Amine Ferrag, Othmane Friha, Djallel Hamouda, Leandros Maglaras, and Helge Janicke. 2022. Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications for Centralized and Federated Learning. *IEEE Access* 10 (2022), 40281–40306.
[7] Georgios Kathareios, Andreea Anghel, Akos Mate, Rolf Clauberg, and Mitch Gusat. 2017. Catch It If You Can: Real-Time Network Anomaly Detection with Low False Alarm Rates. In *ICMLA*.
[8] Ghulam Mohi-ud din. 2018. NSL-KDD. https://doi.org/10.21227/425a-3e55

[9] Nour Moustafa and Jill Slay. 2015. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *MilCIS*.
[10] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. 2018. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In *International Conference on Information Systems Security and Privacy*.
[11] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2014. Going Deeper with Convolutions. https://doi.org/10.48550/ARXIV.1409.4842
[12] Tram Truong-Huu, Nidhya Dheenadhayalan, Partha Pratim Kundu, Vasudha Ramnath, Jingyi Liao, Sin G. Teo, and Sai Praveen Kadiyala. 2020. An Empirical Study on Unsupervised Network Anomaly Detection Using Generative Adversarial Networks. In *SPAI*.