Hierarchical machine learning for IoT anomaly detection in SDN

Perekebode Amangele
University of Essex
p.amangele@essex.ac.uk

Martin J. Reed
University of Essex
mjreed@essex.ac.uk

Mays Al-Naday
University of Essex
mfhaln@essex.ac.uk

Nikolaos Thomos
University of Essex
nthomas@essex.ac.uk

Mateusz Nowak
IITIS PAN, Poland
mateusz@iitis.pl

Abstract— The Internet of Things is a fast emerging technology, however, there have been a significant number of security challenges that have hindered its adoption. This work explores the use of machine learning methods for anomaly detection in network traffic of an IoT network that is connected through a Software Defined Network (SDN). The use of SDN allows a hierarchical approach to machine learning with the aim of reducing the packet level processing of anomaly detection at the edge through applying additional, centralized, machine learning in the SDN controller. For the sake of evaluation, we compare several supervised classification algorithms using a publicly available dataset. The results support a decision-tree based approach and show that the proposed solution promises a considerable reduction in the per-packet processing at the network edge compared to a single stage classifier.

Keywords - network security, SDN, anomaly detection, IoT

I. INTRODUCTION

The Internet of Things (IoT) is a fast-growing network of physical devices that will soon encompass billions of devices. However, there have been significant problems with security in IoT devices due to a combination of: the constrained and heterogeneous nature of the end devices themselves; the fact that end-devices are often not visible and lack standard management systems for firmware/software upgrades; and, connecting to the Internet opens them up to attacks both from and to the Internet. As the IoT devices themselves are often highly constrained, it is not straightforward to operate security services such as intrusion detection/prevention systems on the devices themselves. Therefore, network-based solutions to IoT security are required. Such a network solution is the architecture proposed by the SerIoT project [1], which uses SDN [2] as the network infrastructure to both interconnect the IoT devices and provide the IoT security solution. Our system uses a two-stage classifier to reduce the packet processing effort in an edge classifier by utilising an SDN architecture with an additional centralised anomaly detection layer. Although we address the anomaly detection within the SerIoT SDN architecture, the proposed method is generic and, thus, applicable to other SDN based networks. The work uses modelling of synthetic attack traffic to demonstrate the benefit of the proposed mechanism. In Section II we describe background literature before presenting the proposed hierarchical machine learning solution in Section III. Section IV outlines the classifier design and evaluation while Section V concludes and suggests further work.

Perekebode Amangele is sponsored by Petroleum Technology Development Fund of Nigeria. The work was carried out within the project SerIoT, which has received funding from the European Union's Horizon 2020 Research and Innovation programme under grant agreement No 780139

II. BACKGROUND

A. SDN Overview

SDN is a next generation networking concept which offers greater flexibility and control compared to traditional networks. SDN provides logically centralised control over the network and separates the control and data planes, by abstracting the lower-level functionality allowing network management and control to be directly programmable. This is achieved by extracting the network control logic (control plane) from the underlying switches and routers that perform the actual task of traffic forwarding (data plane). With this separation, network nodes become simple, efficient, forwarding devices and the control function is implemented in a centralised controller [2]. The centralised nature of SDN allows the SDN controller to deploy network-wide policies for both routing and security in a flexible and agile manner. The centralised controller can instruct forwarding devices to allow or block traffic (as well as deciding the route). In this work, we will also use SDN to redirect potentially malicious traffic.

Notably, despite its ability to enhance network security, the SDN architecture introduces new security vulnerabilities into the network: the centralised controller becomes a central point of failure and thus a prime target for attacks [3]. However, due to constraints of space here we will not consider these aspects in this paper.

B. Machine Learning Based SDN Security

In traditional networks, machine learning algorithms have been widely used to detect malicious traffic and classify network attacks [4]. These methods have demonstrated significant potential in the classification of network traffic and are widely used for classification and prediction problems [5]. The major advantage of using machine learning algorithms in SDN is their ability to effect network-wide security rules compared to a more local policy implementation in traditional networks [4].

S. Nanda et al. [5] proposed the use of machine learning algorithms to detect potentially harmful connections and likely attack destinations. This solution utilises four widely used machine learning algorithms to define security rules in the SDN Controller to block potential attack traffic. L. Barki *et al.* utilise a support vector machine classifier and a neural network classifier to detect harmful DDoS traffic towards the SDN controller [6]. The solution in [6] is implemented using an emulation environment and demonstrates the effectiveness of the solution on different network topologies. S. Gangadhar et al. [4] extends the application of Machine Learning to improve traffic tolerance in SDN by extending the functionality of the SDN controller to integrate a resilient

framework, ReSDN. The ReSDN deploys machine learning to detect DDoS traffic in a real-time system; achieving high levels of traffic tolerance.

While the solutions above have demonstrated the effectiveness of SDN as a basis for a security solution, in this paper we explore how SDN can be used to improve the efficiency of a machine learning approach by distributing the learning process. In particular, the SerIoT project exploits the SDN architecture to increase the security of IoT devices and of the network itself [1],[7]. Detection of anomalies is crucial for the proper operation of SerIoT network security mechanisms thus the proposal described in this paper is a natural fit into the framework of that project.

III. PROPOSED 2-STAGE HIERARCHICAL MACHINE LEARNING BASED SDN SECURITY SOLUTION

This paper proposes a novel, 2-stage hierarchical machine learning process, integrated into an SDN architecture for network traffic anomaly detection and mitigation. We adopt SDN in our framework because of the flexibility it offers, i.e. it has the capacity to dynamically address security requirements of networks, and the ability of a central controller to have a global view of the network. The proposed SDN solution integrates a 2-stage machine learning instance as shown in Fig. 1. The first instance of machine learning, i.e., Classifier 1, works on summarized network flow traffic features captured using techniques such as IPFIX [8]. This classifier is implemented in the SDN Controller and serves as a central classifier that works on gross flow level information. This is a reasonable approach as it is inefficient to send every packet of the flow to a central classifier. Although, by using gross flow level information this could result in a relatively poorer false positive performance (some good packets may be classified as bad), the aim of this classifier is to have low false negatives (i.e., to identify all bad packets). Thus, the central Classifier 1 is used to identify potentially harmful network traffic which is fed into a second machine learning stage, Classifier 2 as shown in Fig. 1, which works on a per-packet basis at the network edge.

The key reason for employing the above two-stage approach is to improve the efficiency of anomaly detection compared to, less efficient, alternative strategies: edge anomaly detection and centralised anomaly detection. In traditional edge anomaly detection, traffic classification is carried out at the edge, however, this involves processing every packet in a classifier. This somewhat negates the use of efficient switches in the network, as, if every packet is passed through an anomaly detector it is effectively already process switched and there is no need for the actual network hardware switch. This means switches would need to be replaced by expensive (both in hardware capability and cost) CPU intensive devices. Additionally, an edge device cannot make use of wider network knowledge. Alternatively, in the case of centralised anomaly detection, if the processing is carried out only in the central SDN controller (or some other central monitoring position), it cannot make use of finer packet level information as it is not scalable to send all the traffic to a single network monitoring device.

The two-stage classifier proposed in this paper overcomes the problems described previously, by combining both the edge and centralised classifiers, as shown in Fig. 1. The SDN architecture is ideally placed to provide this hierarchical approach as the SDN controller already provides a central computation resource that is designed to receive monitoring information. Additionally, the SDN switch can highly efficiently divert the, potentially, malicious traffic to Classifier 2. Consequently, Classifier 1 identifies which flows are potentially bad and only packets of such flows are diverted to Classifier 2 using the SDN switch.

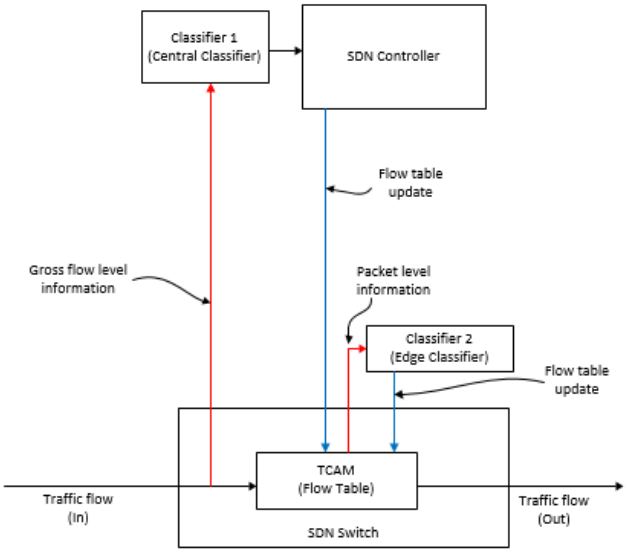


Fig 1 Hierarchical Machine Learning Architecture for SDN Security (note only one switch of the many is shown)

IV. CLASSIFIER DESIGN AND EVALUATION

A. Dataset for model selection and evaluation

The classifier selection was carried out using the CICIDS2017 dataset which is specifically for network security and intrusion detection testing [9]. The CICIDS2017 dataset is comprised of seven attack categories: denial of service (DoS) (Slowloris, Hulk, Golden Eye, Heartbleed, Slowhttptest), distributed DoS (DDOS), botnet traffic (BOT) that does not include DoS traffic, Patator (SSH-patator, FTP-patator), Infiltration, Portscan (PSCAN) and Web Attacks (SQL injection, XSS, password brute force) [9]. While all of these attacks were used in the preparation of this work, the results report from a smaller subset to save space. The dataset consists of both the packets captured and flow level information for a mix of benign and attack traffic. The flow level information used in this paper for Classifier 1 consists of the aggregated information available from the unencrypted fields of the packets, i.e. the IP and TCP/UDP headers, but not the contents, which for most data is encrypted and, thus, not available for interpretation. The information from the headers of the packet are summarised into 80 fields including the static field content for each flow (i.e. addresses, ports) and the coarse level statistics (e.g. min, mean, max etc) from header fields such as packet length and TCP window size, to name only two.

TABLE I. MOST IMPORTANT 6 FEATURES FOR DDOS

	Feature	Importance
1	Fwd Packet Length Max	0.7907
2	Destination Port	0.4464
3	Init_Win_bytes_forward	0.3176
4	Total Length of Fwd Packets	0.047
5	Subflow Fwd Bytes	0.0421
6	Init_Win_bytes_backward	0.0367

B. *Classifier and feature selection*

The quality metrics used in assessing the machine learning algorithms include the standard accuracy, precision, recall and F1 metrics which operate on true/false positive (TP/FP) and true/false negative (TN/FN) occurrences as defined below:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN} \quad (1)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2)$$

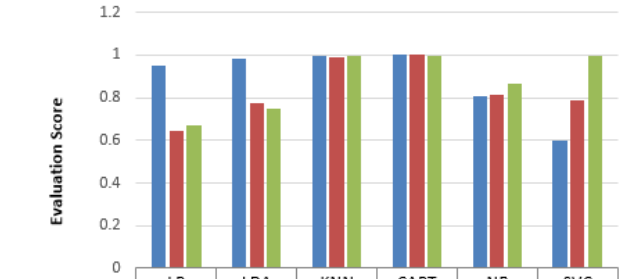
$$\text{Precision} = \frac{TP}{TP + FP} \quad (3)$$

$$F_1 = 2\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

The recall metric is of particular importance in the design of Classifier 1 because of the need for a very low false negative but it is less concerned about the false positive rate as this is dealt with by Classifier 2.

To ensure an efficient classification model, feature selection was performed to remove unnecessary features. The feature importance was determined using mean decrease accuracy [10] with an example output for the 6 most important features of the DDoS category of attack shown in Table I. The features for the other attack types are not shown here due to limitations of space, but it should be noted that the selected features and trained model were different for each attack type (as is to be expected).

Algorithm selection was performed over a number of different metrics including both machine learning quality metrics and prediction time. The latter is of concern as the aim is to run the algorithm on real-time network traffic, whereas the model fit time is less important as the fit can be performed offline. Performance was compared across six algorithms from the Scikit-learn Python package [11]: Linear Regression (LR), Linear Discriminant Analysis (LDA), k-Nearest Neighbour (KNN), Classification and Regression Tree (CART), Naive Bayes (NB) and Support Vector Classification (SVC). Fig. 2 shows the accuracy score of the different algorithms when considering different feature sets i.e. using all 80 features or only the first 6 or 3 most important respectively. Initial results indicate CART and KNN both return very high evaluation scores of approximately 99%



	LR	LDA	KNN	CART	NB	SVC
All Features	0.950584	0.980523	0.997297	0.999862	0.808161	0.59787
1st 6 Features	0.645956	0.773116	0.992114	0.999424	0.812032	0.785001
1st 3 Features	0.66725	0.745968	0.997397	0.997392	0.867197	0.99738

Fig. 2. Performance Evaluation of the six machine learning algorithms for DDoS using accuracy scoring.

TABLE II PREDICTION TIMES FOR DDOS USING DIFFERENT FEATURE SELECTION RESULTS

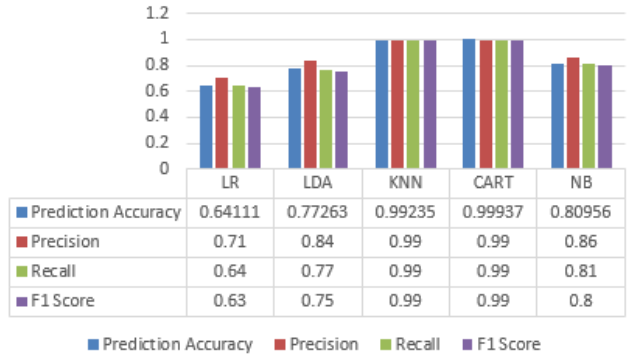
Algorithm	Train Time	Predict Time for DDoS (s)		
		All Feat	6 Feat	3 Feat
LR	0.89	0.008985	0.001996	0.000998
LDA	0.11	0.007979	0.000998	0.000998
KNN	2.71	5.971206	5.626392	11.625983
CART	0.12	0.012965	0.002023	0.002023
NB	0.04	0.094778	0.006981	0.003963

across all feature set combinations. Other attack types had differing results but approximately the same trends in terms of ranking of performance. The SVC had very slow fit and high prediction times so it was rejected after this phase as it had little advantage in terms of performance, and thus is not included in later results.

The second phase of algorithm selection investigated the prediction time with results shown in Table II, which also shows the training times. From the first phase of selection, KNN and CART were identified as the most promising algorithms; whereas, the predict time shows CART has significantly better runtime performance for similar quality metric compared to KNN. Finally taking all performance metrics into consideration, CART with the 6 highest features was selected as the chosen model. Again, while other attacks showed differing performances CART with the 6 most important features performed well across all attack types. It is interesting to note that CART performed well with differing number of features but showed a significant improvement to predict time without sacrificing quality when moving to 6 features whereas other algorithms (excepting KNN) showed significant performance degradation with fewer features. Another significant advantage of CART is that it provides straightforward interpretation of the classifier through the decision tree, which allows the parameters to be directly viewed; although, an example is not shown here due to space restrictions. Fig. 3 presents further results which include a range of model implementation metrics such as accuracy, precision, recall and F1 score. From these results we can see that CART performs well across them all.

Further detailed analysis of the results, not easily shown in the overall aggregated results, opens up the selection process discussion a little. The DDoS validation dataset had a total of 45,143 traffic flows with 19,679 benign and 25,464

DDoS. The CART Classifier failed to classify 18 DDoS traffic flows (false negative) while also returning a false positive of 10. The 1st stage Classifier of the proposed hierarchical machine learning architecture is required to have a very low false negative. The LDA algorithm had a lower false negative result for this category with only 2 false negative flows and thus could have been considered for the first stage Classifier. However, LDA had 10,262 false



	LR	LDA	KNN	CART	NB
Prediction Accuracy	0.64111	0.77263	0.99235	0.99937	0.80956
Precision	0.71	0.84	0.99	0.99	0.86
Recall	0.64	0.77	0.99	0.99	0.81
F1 Score	0.63	0.75	0.99	0.99	0.8

Fig. 3. Algorithm comparison for DDoS

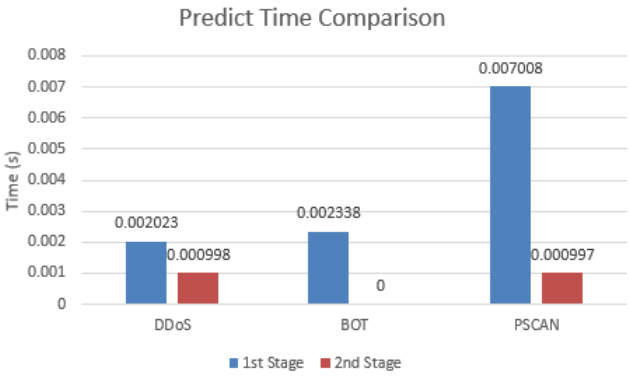


Fig. 4. Comparison between 1st stage and 2nd stage processing times for three attack types, where the 1st stage result shows the effort required if using an edge-only classifier without the hierarchical approach proposed. Note that for BOT the 2nd stage result is too small to measure

positives, which could have overloaded Classifier 2. On balance, using CART, a very small number of DDoS packets would get through, but this would be acceptable taking into account the whole algorithm performance and attack type into consideration. Thus, it is important to consider the wider performance issues when selecting an algorithm, not just the raw statistics.

The final analysis looks at the performance consideration of the hierarchical approach when considering the second stage classifier. Results for this comparison are shown in Fig. 4 for three different attack types: DDoS, BOT and PSCAN. These results are using the selected CART algorithm for Classifier 2 with the 6 most important features and compare the effort if only implementing the single stage classifier at the edge vs implementing the two stage classifier. The results show the reduction on effort in the second stage classifier is significant, in particular for the BOT and PSCAN traffic (with BOT nearly zero). This is important as it shows the benefit of moving from an edge-based classifier alone to the proposed two-stage classifier can lead to a significant reduction in packet level traffic needed to be processed in Classifier 2, thus leading to a more scalable approach for practical deployment.

V. CONCLUSION

A novel hierarchical machine learning architecture for SDN security has been proposed in this paper. The architecture consists of two classifier stages with the first classifier implemented in the SDN controller and the second implemented at the edge in a processing device co-located with the SDN switch. A range of suitable machine learning algorithms were evaluated, suggesting that a classification and regression tree model is the most suitable algorithm from those investigated. The results show that using the proposed hierarchical approach there is a significant reduction in the number of packets that have to be processed in the classifier associated with the SDN switches. Future work will investigate the second stage classifier in more detail.

REFERENCES

[1] E. Gelenbe, J. Domanska, T. Czchorski, A. Drosou, and D. Tzovaras, "Security for Internet of Things: The SerIoT Project," in Proc. of Int. Symp. on Networks, Computers and Communications (ISNCC), Rome, Italy, June 2018.

[2] D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[3] I.Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, "Security in Software Defined Networks: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2317–2346, Apr. 2015.

[4] S. Gangadhar and J. P. Sterbenz, "Machine learning aided traffic tolerance to improve resilience for software defined networks," *Proc. of Int. Workshop on Resilient Networks Design and Modeling, RNDM 2017*, Sep. 2017.

[5] S. Nanda, F. Zafari, C. Decusatis, E. Wedaa, and B. Yang, "Predicting network attack patterns in SDN using machine learning approach," *Proc. of IEEE Conf. on Network Function Virtualization and Software Defined Networks, NFV-SDN 2016*, pp. 167–172, Nov. 2017.

[6] L. Barki, A. Shidling, N. Meti, D. G. Narayan, and M. M. Mulla, "Detection of distributed denial of service attacks in software defined networks," *Proc. of Conf. on Advances in Computing, Communications and Informatics, ICACCI 2016*, pp. 2576–2581, Sep. 2016.

[7] J. Domanska, M. Nowak, S. Nowak, and T. Czachorski, "European cybersecurity research and the seriot project," in *Computer and Information Sciences*, T. Czachó rski, E. Gelenbe, K. Grochla, and R. Lent, Eds. Cham: Springer International Publishing, 2018, pp. 166–173.

[8] B. Trammell and E. Boschi, "An introduction to IP flow information export (IPFIX)," *IEEE Communications Magazine*, vol. 49, no. 4, pp. 89–95, Apr. 2011.

[9] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in *Proc. of Int. Conf. on Information Systems Security and Privacy (ICISSP)*, Funchal, Madeira, Portugal, Jan. 2018, pp. 108–116.

[10] H. Han, X. Guo, and H. Yu, "Variable selection using Mean Decrease Accuracy and Mean Decrease Gini based on Random Forest," *Proc. of the IEEE Int. Conf. on Software Engineering and Service Sciences, ICSESS*, pp. 219–224, Nov 2017.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, Jan. 2011.