Intrusion and Vulnerability Detection in Software Defined Networks

Sotiris Chatzimiltis*, Mohammad Shojafar*, Mahdi Boloursaz Mashhadi*, and Rahim Tafazolli*
* 5GIC & 6GIC, Institute for Communication Systems (ICS), University of Surrey, Guildford, UK
sc02449@surrey.ac.uk, m.shojafar@surrey.ac.uk, m.boloursazmashhadi@surrey.ac.uk, r.tafazolli@surrey.ac.uk

***Abstract*—Software Defined Networks (SDNs) have revolutionised the way modern networks are managed and orchestrated. This sophisticated infrastructure can provide numerous benefits but at the same time introduce several security challenges. A centralised controller holds the responsibility of managing the network traffic, thus making it an attractive target to attackers. Intrusion detection systems (IDS) play a crucial role in identifying and addressing security threats within the SDN. By utilising machine learning algorithms an anomaly based detection system was established to identify deviation in network behaviour. Five machine learning algorithms were employed to train the SDN-IDS, and ultimately, the most appropriate one was chosen. The SDN-IDS demonstrated an exceptional overall performance, particularly when using the XGBoost Classifier trained with a reduced feature train dataset, reaching 99.9% accuracy as well as 99.9 F1-score. Furthermore, it exhibited near-perfect performance in identifying the most types of attacks within the traffic data.**

***Index Terms*—Software Defined Network, Imbalanced Data, Intrusion Detection System, Machine Learning.**

I. Introduction

Software Defined Networks (SDNs) are a fundamental technology driving the evolution of modern networks, allowing more efficient and agile management of network resources while allowing the development of advance networking applications. SDNs enable dynamic and centralized control of network resources. By decoupling the network's control plane from the data plane, network administrators can easily manage the entire network infrastructure through software applications rather than configuring individual devices (e.g. routers, switches) one by one [1]

Transitioning from traditional networks to SDNs introduces plenty of advantages such as programmability and automation. Because of the structural design of the SDN, security concerns and potential threats arise within each of its components including the controller, northbound API and southbound API. The controller essentially is the core of an SDN. Gaining access to the controller will enable the intruder to manipulate the network. Further attacks can include denial of service (DoS), paralysing the network, and man in the middle attacks that disrupt the integrity of the communications [2].

To mitigate these security threats, SDNs should employ best practices such as authentication and authorisation mechanisms, encryption of communications, traffic monitoring and intrusion and vulnerability detection. Developing an intrusion detection system (IDS) can be achieved using different techniques or strategies such as signature-based and anomaly-based. Signature-based intrusion detection systems operate by comparing a list of known threats with the packets traversing through the network. On the other hand anomaly-based IDS utilises machine learning techniques to create models that can distinguish between normal and malicious behaviours. In pursuit of this project's objectives, different machine learning techniques are employed to create an anomaly-based intrusion detection system [3].

Furthermore, network traffic datasets used for IDS are usually mostly comprised of benign data traffic while having only a relatively small fraction of malicious data that indicate security threats or attacks. This data distribution creates an imbalanced dataset, where the number of instances in each class is significantly different. Imbalanced data usually lead to a biased model towards the majority class. To address this problem several techniques can be employed such as resampling the data or the use of appropriate ML techniques such as ensemble methods. From our perspective to tackle this problem, resampling techniques such as SMOTE [4] and Tomek's link [5] were utilised in order to alleviate data imbalances between classes, as well as ensemble methods such as XGBoost and Bagging [6] that helped to improve the overall model performance.

Contributions of the paper. The main contribution of the paper is as follows:

1) We propose and implement an IDS for SDNs to defend itself from data-traffic attacks.
2) We propose different techniques to handle data class imbalances.
3) Employing ML methodologies for the SDN-based IDS.
4) Assessing the effectiveness of the proposed intrusion detection system through an extensive analysis of performance metrics using a novel dataset.

The following sections are organised as follows. Section II discusses the methodology used for this project. Section III displays a performance analysis for each machine learning technique used and discusses the findings as well. Finally, Section IV provides a summary of the project, presents key conclusions, and suggests potential avenues for future work.

II. Methodology

This section describes the steps of our proposed intrusion detection designed to address the issue presented by [7] in [8] . Furthermore, a small analysis of each every machine learning technique used is appended.

TABLE I: Train and Test dataset flow type breakdown

A/A	Traffic Flow Type	Training Set	Cleaned Training Set	Test Set	Cleaned Test Set
0	**Benign**	1432050	1337046	411203	410865
1	**BOT**	1238	1228	355	354
2	**DDoS**	80656	80653	23160	23160
3	**DoS GoldenEye**	6484	6480	1861	1861
4	**DoS Hulk**	145575	109334	41801	41626
5	**DoS Slowhttptest**	3464	3302	944	994
6	**DoS slowloris**	3651	3411	1048	1048
7	**FTP-Patator**	5000	3819	1436	1436
8	**Hearbleed**	6	6	2	2
9	**Infiltration**	22	22	6	6
10	**Portscan**	100125	66659	28751	28728
11	**SSH-Patator**	3714	2086	1067	1067
12	**Web Attack Brute Force**	949	935	272	272
13	**Web Attack Sql Injection**	12	12	4	4
14	**Web Attack XSS**	410	410	117	117

A. Data & Data Pre-processing

For this project training and test datasets were provided by ULAK. The initial training set was constituted of 1,783,356 records and the initial test set of 512,077 records. Each record had 78 features to begin with and a label column describing the category of the data flow.

Starting with the data cleaning phase, 1,811 training records and 537 test records were firstly removed since the had either NaN or Infinite values. In addition 166,142 training records were removed as well because they were duplicates. The training set after data cleaning had 1,615,403 and the new test set 511,540 records. Moreover all records (training and test sets) with feature values of -1 (minus one) were replaced with the mean value of the respective training feature column. Finally eight (8) features were dropped since they only contributed zero values, thus now each sample had 70 features to be represented with. Table I shows the different data flow types and gives a breakdown of how those labels were distributed through the initial train and test sets and through the new cleaned sets (baseline datasets).

The second pre-processing step was label encoding, and it was used to transform the categorical values of the label column into numerical ones. Furthermore feature normalisation was applied to the features. The standard scaler was used to normalise the data by subtracting the mean of the respective column and dividing by the standard deviation. It has to be mentioned that feature normalisation was only used just before the training of each machine learning model.

The third data pre-processing stage was to handle the data imbalance between the different types of network traffic. Since the training dataset was composed of almost 90% of Benign class samples a model could easily overfit during training. There were two main methods of handling data imbalances at this stage. The first scheme was oversampling the minority classes, whereas the second way was undersampling the majority classes.

Synthetic minority over-sampling technique (SMOTE) [4] was used to over-sample the minority classes. SMOTE tries to equalise the class distribution in the dataset by creating artificial instances for the minority classes. The number of instances to be created are indicated by the user.

For majority-class undersampling two methods were used. The first method utilises tomek-links [5] and Nearest Neighbours (NN) model, to reduce the size of the initial dataset while trying not to degrade the performance of the model. Samples are removed if (1) is satisfied, meaning that if two samples (x, y) of different classes are the nearest neighbours then the sample of the majority class $(x \| y)$ is removed. The second method is called Repeated Edited Nearest Neighbours (RENN) [9] that again uses the NN model and repeatedly applies the edited nearest neighbour rule. The Edited Nearest Neighbour computes the nearest neighbours of every sample that belongs to a class to be under-sampled. If the majority or all the neighbours of a sample belong to the same class then the sample remains in the set else is discarded. All of the methods used were implemented by [10]. For final experiments the tomek's technique was used.

$$[d(x,y) < d(x,z) \ \ \& \ \ d(x,y) < d(y,z)] \forall z \in Samples \quad (1)$$

Explanation: x and y are two instances from different classes. If there is no data point z such that $d(x,z) < d(x,y)$ and $d(y,z) < d(x,y)$, where $d(a,b)$ represents the distance between data points a and b, then x and y is a tomek's link.

Feature selection was the final step of the data pre-processing stage. As per [11], feature selection algorithms help discard irrelevant features that negatively affect the performance of the model. This reduction in feature dimensionality, also helps in reducing the overall time taken to train an ML model. As mentioned above eight features were removed on a preliminary stage since they had only zero values. The new set of features were then evaluated based on their importance using a Random Forest classification algorithm in a recursive manner until a small enough subset of features remained that would still performed good enough. A 5-fold stratified cross validation dataset was created in order to determine the importance of the features. The importance of each feature over every fold was averaged. The average importance of each feature was added, in descending order, until 90% of the total importance was accumulated. All the features that didn't contribute were then discarded.

TABLE II: Performance Metrics Description

Metric	Equation	Explanation
Accuracy	$(TP+TN)/(TP+FP+TN+FN)$	Percentage of correctly classified instances
Precision	$(TP/(TP+FP))$	Percentage of positive class predictions, that are indeed positive
Recall	$(TP/(TP+FN))$	Measures how well the model can correctly identify instances of positive class
F1-score	$(2*Precision*Recall)/(Precision+Recall)$	A single score the balance Precision and Recall together
False Positive Rate	$(FP/(FP+TN))$	Probability of a false alarm

B. Baseline Machine Learning Models

Three baseline models were chosen to be used among with the some more complex models. Decision Trees (DT), Random Forest (RF) and K-Nearest Neighbours (K-NN) were selected since they have an extensive use in the topic of IDS, are easy to implement and support multi-class classification. Every ML model used is described below.

Decision Trees (DT) are models that follow a sequential structure and combine a series of simple tests in a logical manner in order to finally classify an instance into a specific category. The algorithm aims to create general patterns (decision rules) inferred from the data features that will help with the classification of the data. Finally decision trees are beneficial in terms of their comprehensive interpretations, meaning that the outcome can be traced-back to observe the decisions made [12].

A Random Forest (RF) is defined as an ensemble of base tree classifiers. Using majority voting the final classification is made [13]. The theory behind the promising performance of RF is that multiple uncorrelated models, created using different data sub-samples, can outperform an individual model.

K-Nearest Neighbours (K-NN) is the final baseline classification technique. K-NN works based on a simple mechanism. A new data point is assigned the class that is most common among its neighbouring data points in the feature space [14].

C. Ensemble Learning Classifiers

There exist several ways to enhance the predictive performance of an individual ML model using ensemble learning. Two prominent ways are Bagging and boosting classification algorithms. Bagging classifiers, like RF, create and train base classifiers without inter-dependency using different data subsets. On the other hand, boosting classifiers work in a sequential manner by trying to learn from the errors of previous iterations [6]. For this project a standard bagging classifier was used having DTs as base classifiers. Finally XGBoost classifier was used from the boosting classifiers.

D. Performance Metrics

Table II shows a description of all the metrics used to evaluate the performance of an ML model. When dealing with imbalanced data, the accuracy measurement cannot reflect the true performance of an ML technique. A simple classifier can predict all samples to be in the majority class (Benign Data), thus achieving a high accuracy model, but fail to classify instances belonging to minority classes [15] leading to a useless overall model. Metrics such as precision, recall and F1-score were used to better capture the ability of the classifier.

III. Results & Discussion

This section presents the results carried out for every machine learning technique used for this project. A total of four variants of the initial training dataset were used for training and validation. Two datasets contained all 70 features and two of them had reduced features (34). Furthermore two of the datasets contained resampled instances as well in order to alleviate data imbalance. Stratified 5-fold cross validation was used to ensure the class distribution in every fold was maintained. The average performance of the 5-fold cross validation is presented. Implementation source code can be found at [16].

A. Performance Evaluation with 5-Fold Cross Validation

As previously stated, a total of five machine learning models were employed, and the results of their average 5-Fold cross-validation performance are presented below. Macro and weighted averages were calculated to assess the model's overall performance, with and without taking into account the class distribution across instances. The macro average indicates the overall performance across all labels, offering insight into the general quality of predictions for each label. On the other hand, the weighted average provides a performance measure that considers the class distribution, revealing how well the model performs even in cases where our case were the majority of instances are benign. Figure 1 illustrates the macro average 5-Fold performance of each ML model trained with the baseline dataset, while the corresponding weighted performance metrics are presented in Table III."

TABLE III: SDN-IDS Weighted Average performance evaluation for 5-Fold Cross-validation using the baseline dataset.

Model	Precision	Recall	F1-Score	Accuracy
DT	0.9990	0.9990	0.9990	0.9973
RF	0.9988	0.9996	0.9991	0.9977
K-NN	0.9981	0.9986	0.9985	0.9958
Bagging	0.9991	0.9991	0.9991	0.9982
Boosting	0.9992	0.9995	0.9993	0.9985

Additionally, Table IV presents the weighted average performance achieved when models were trained with the reduced baseline dataset, while Table V displays the results for each model trained using resampled datasets, whether reduced or not. Finally, in Figure 2, you can observe the normalised confusion matrix for the top-performing model.

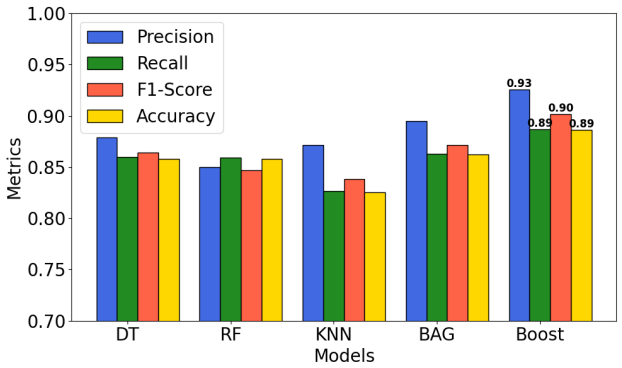

Fig. 1: SDN-IDS Macro Average performance evaluation for 5-Fold Cross-validation using the baseline dataset.

TABLE IV: SDN-IDS Weighted Average performance evaluation for 5-Fold Cross-validation using the reduced baseline dataset.

Model	Precision	Recall	F1-Score	Accuracy
DT	0.9991	0.9991	0.9991	0.9978
RF	0.9991	0.9992	0.9991	0.9976
K-NN	0.9982	0.9987	0.9985	0.9969
Bagging	0.9991	0.9991	0.9991	0.9984
Boosting	0.9992	0.9995	0.9993	0.9989

TABLE V: SDN-IDS Weighted Average performance evaluation for 5-Fold Cross-validation using the resampled baseline and reduced resampled baseline datasets.

Model	Precision	Recall	F1-Score	Accuracy
DT	0.9983	0.9983	0.9987	0.9983
RF	0.9838	0.9815	0.9815	0.9813
DT Reduced	0.9896	0.9898	0.9899	0.9895
RF Reduced	0.9650	0.9626	0.9619	0.9623
Bagging Reduced	0.9959	0.9957	0.9960	0.9955

Tables III and IV reveal that the models exhibit nearly identical performance, with only minor variations in terms of accuracy and F1-scores. The top-performing models were the ensemble models, such as Bagging and Boosting, as they leverage multiple classifiers to derive their final predictions. Additionally, an examination of Table V indicates that the DT and Boosting models perform reasonably well. However they do not contribute to a significant performance enhancement even though the data have been resampled to eliminate class imbalance.

TABLE VI: SDN-IDS Weighted Average performance evaluation of the Test Set when models were trained using different datasets.

Model/Dataset	Precision	Recall	F1-Score	Acc
DT Resampled	0.9988	0.9980	0.9986	0.9968
RF Resampled	0.9989	0.9996	0.9995	0.9984
DT Resampled Reduced	0.9988	0.9980	0.9986	0.9968
RF Resampled Reduced	0.9989	0.9996	0.9995	0.9986
Bagging Resampled Reduced	0.9988	0.9980	0.9986	0.9972
DT Baseline Reduced	0.9989	0.9973	0.9981	0.9966
RF Baseline Reduced	0.9991	0.9996	0.9996	0.9988
KNN Baseline Reduced	0.9945	0.9876	0.9948	0.9915
Bagging Baseline Reduced	0.9990	0.9996	0.9990	0.9985
XGBoost Baseline Reduced	0.9990	0.9996	0.9996	0.9990
DT Baseline	0.9989	0.9974	0.9981	0.9964
RF Baseline	0.9990	0.9995	0.9990	0.9982
KNN Baseline	0.9968	0.9962	0.9966	0.9932
Bagging Baseline	0.9990	0.9996	0.9995	0.9983
XGBoost Baseline	0.9990	0.9996	0.9996	0.9986

B. Performance Evaluation with Test Set

Taking into account the performance observed during the 5-Fold cross-validation, we proceeded to train the ML models using the entire training dataset and then evaluate them using the provided test dataset. Table VI presents the performance metrics' weighted averages of the ML models while Figure 2 the macro averages of the best performing models and dataset combinations. It can be observed that different combination result in better macro performance where different combination results in better weighted performance. XGBoost model trained with the reduced baseline dataset achieved an accuracy of 99.9% and an F1 score of 99.9 as well. On the other hand the DT model trained with the reduced feature resampled dataset had an accuracy of 87.77% and an F1 score of 87.20. The normalised confusion matrices for XGBoost and DT models are located in appendix as Figure 3 and Figure 4. Finally Table VII shows the false positive rates (FPR) of those two models for every data traffic type.

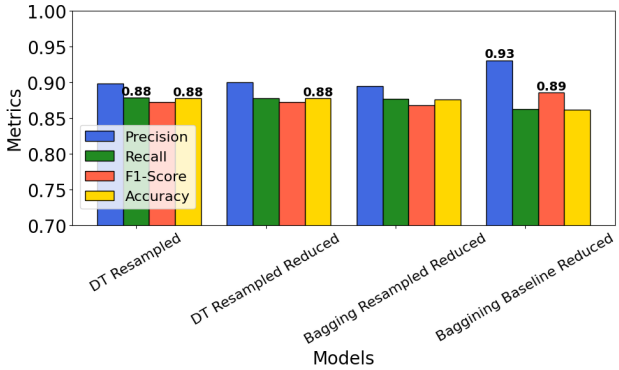

Fig. 2: Best SDN-IDS Macro Average performances of Test Set.

TABLE VII: False Positive Rate of XGBoost and DT models

Traffic Type	XGBoost	DT
Benign	0.001	0.009
Bot	0.00008	0.0003
DDoS	0.000008	0.00001
DoS Golden Eye	0.000007	0.00005
DoS Hulk	0.0001	0.0001
DoS Slowhttptest	0.00002	0.00006
DoS Slowloris	0.000009	0.00002
FTP Patator	0.000001	0.000001
Heartbleed	0	0
Infiltration	0	0.000004
ProtScan	0.0003	0.0004
SSH Patator	0.000005	0.000004
Web Attack Brute Force	0.0001	0.00001
Web Attack Sql Injection	0	0
Web Attack XSS	0.00008	0.0001

The confusion matrices reveal that the XGBoost model faces challenges when classifying the three web attacks and the Infiltration attack. On the other hand, the DT model exhibits a slight improvement in classifying these attacks but at the cost of a slight decrease in normal traffic classification performance. In both models, the False Positive Rates (FPRs) are minimal due to the large size of the test set, making the false positives negligible when compared to the true negatives.

IV. Future Work & Conclusions

This paper introduces an SDN-IDS that employs anomaly-based intrusion detection techniques with ML models. The system preprocesses the data by cleaning, encoding, and resampling it. Following this, a feature selection algorithm is applied to reduce data dimensionality, and ultimately, data normalisation is performed before feeding the data into the ML models. Also a new resampled dataset was generated. The highest level of performance was achieved when training the XGBoost model with a feature-reduced dataset, resulting in an accuracy and F1-score of 99.9%. Additionally, the Decision Tree (DT) model demonstrated strong overall performance when trained with the reduced resampled dataset, notably enhancing the classification of classes that XGBoost struggled with.

As part of our future work, we intend to explore various strategies to address the data imbalance challenge present in network traffic datasets. One prospective approach involves implementing a hierarchical classification scheme. Initially, this scheme will start with a binary classification, distinguishing between normal and attack traffic. Subsequently, it will dive deeper into finer classifications, distinguishing between different attack types, such as DoS, malware, and Web Attacks. Finally, the classification process will end up with the identification of the ultimate data traffic class. Additionally, another approach will involve a multi-step binary classification strategy. It will commence by selecting an equal portion of normal data (majority class) and attack data. In each subsequent step, the goal will be to classify classes with approximately the same number of instances in the training dataset, thereby mitigating the risk of over-fitting. Finally an ensemble can be created using the predictions made from the XGBoost model, that has higher accuracy in predicting normal data, and the predictions made from the DT model that has higher accuracy in minority classes. This fusion is expected to further enhance the overall performance of the final model.

References

[1] G. P. Tank, A. Dixit, A. Vellanki, and D. Annapurna, "Software-Defined Networking the new norm for networks," 2012. [Online]. Available: https://api.semanticscholar.org/CorpusID:53052362

[2] Y. Maleh, Y. Qasmaoui, K. E. Gholami, Y. Sadqi, and S. Mounir, "A comprehensive survey on SDN security: threats, mitigations, and future directions," *Journal of Reliable Intelligent Environments*, vol. 9, no. 2, pp. 201–239, Feb. 2022. [Online]. Available: https://doi.org/10.1007/s40860-022-00171-8

[3] N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, "Survey on SDN based network intrusion detection system using machine learning approaches," *Peer-to-Peer Networking and Applications*, vol. 12, no. 2, pp. 493–501, Jan. 2018. [Online]. Available: https://doi.org/10.1007/s12083-017-0630-0

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, jun 2002. [Online]. Available: https://doi.org/10.1613%2Fjair.953

[5] I. Tomek, "Two modifications of CNN," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-6, no. 11, pp. 769–772, 1976.

[6] S. González, S. García, J. Del Ser, L. Rokach, and F. Herrera, "A practical tutorial on bagging and boosting based ensembles for machine learning: Algorithms, software tools, performance study, practical perspectives and opportunities," *Information Fusion*, vol. 64, pp. 205–237, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253520303195

[7] ITU AI For Good Global Summit. [Online]. Available: https://challenge.aiforgood.itu.int

[8] Machine Learning for SDN security: improving intrusion and vulnerability detection. [Online]. Available: https://www.youtube.com/watch?v=zgne_H0Ki7M

[9] "An experiment with the edited nearest-neighbor rule," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-6, no. 6, pp. 448–452, 1976.

[10] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017. [Online]. Available: http://jmlr.org/papers/v18/16-365.html

[11] N. Bindra and M. Sood, "Evaluating the impact of feature selection methods on the performance of the machine learning models in detecting DDOS attacks," *Romanian Journal of Information Science and Technology*, vol. 23, no. 3, p. 250–261, 2020.

[12] S. Kotsiantis, "Decision Trees.: a recent overview," *Artificial Intelligence Review*, vol. 39, no. 4, p. 261–283, 2013. [Online]. Available: https://doi.org/10.1007/s10462-011-9272-4

[13] G. Biau and E. Scornet, "A random forest guided tour," *TEST*, vol. 25, no. 2, pp. 197–227, Apr. 2016. [Online]. Available: https://doi.org/10.1007/s11749-016-0481-7

[14] K. Taunk, S. De, S. Verma, and A. Swetapadma, "A brief review of nearest neighbor algorithm for learning and classification," in *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, 2019, pp. 1255–1260.

[15] Y. Tang, Y.-Q. Zhang, N. V. Chawla, and S. Krasser, "SVMs modeling for highly imbalanced classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 1, pp. 281–288, 2009.

[16] S. Chatzimiltis, M. Shojafar, M. Mahdi Boloursaz, and R. Tafazolli, "SDN-IDS source code," https://github.com/sotirischatzimiltis/SDN_IDS, 2023.

APPENDIX

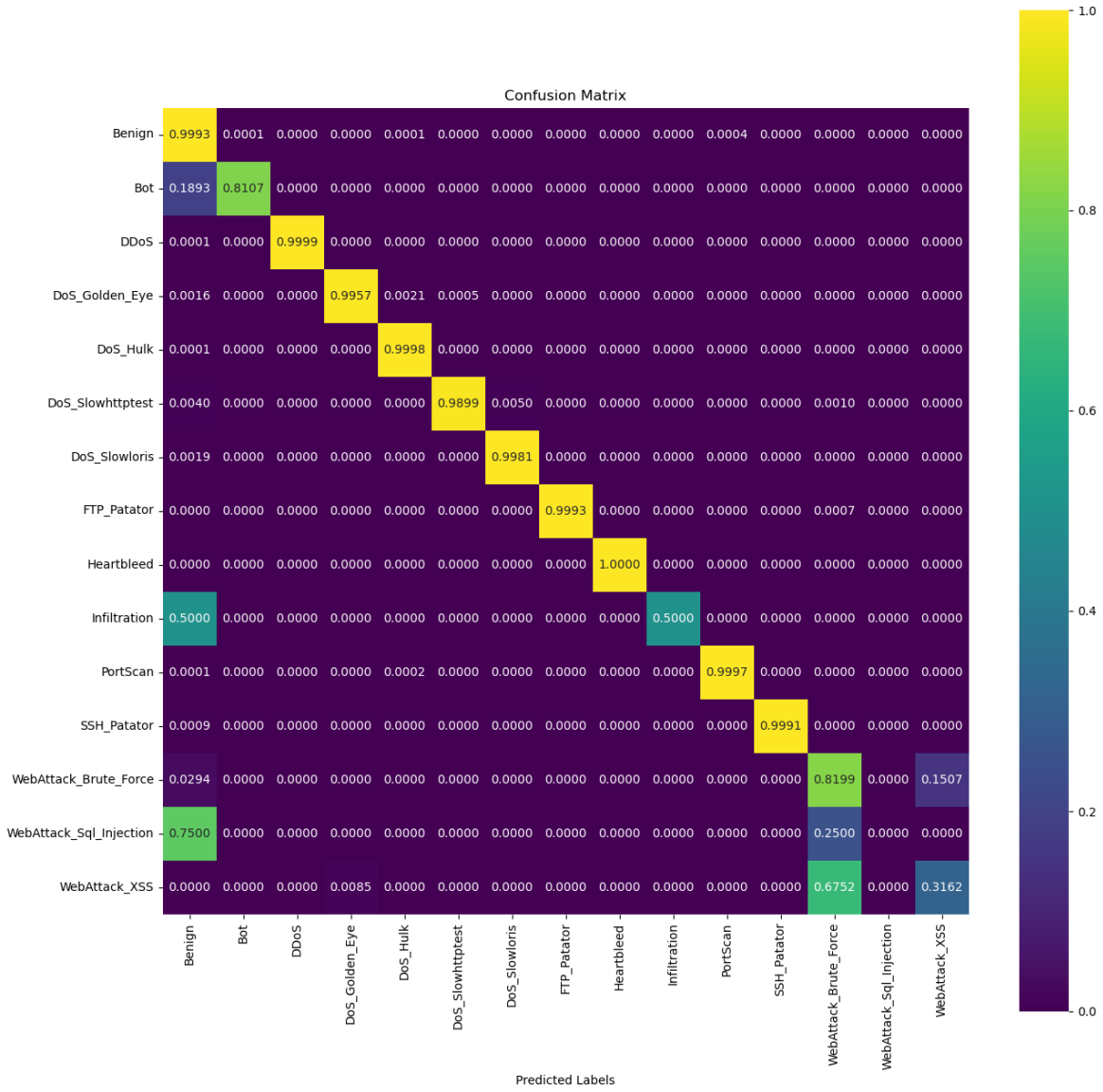

Fig. 3: XGBoost Test Set Normalised Confusion Matrix when model was trained with the reduced baseline dataset

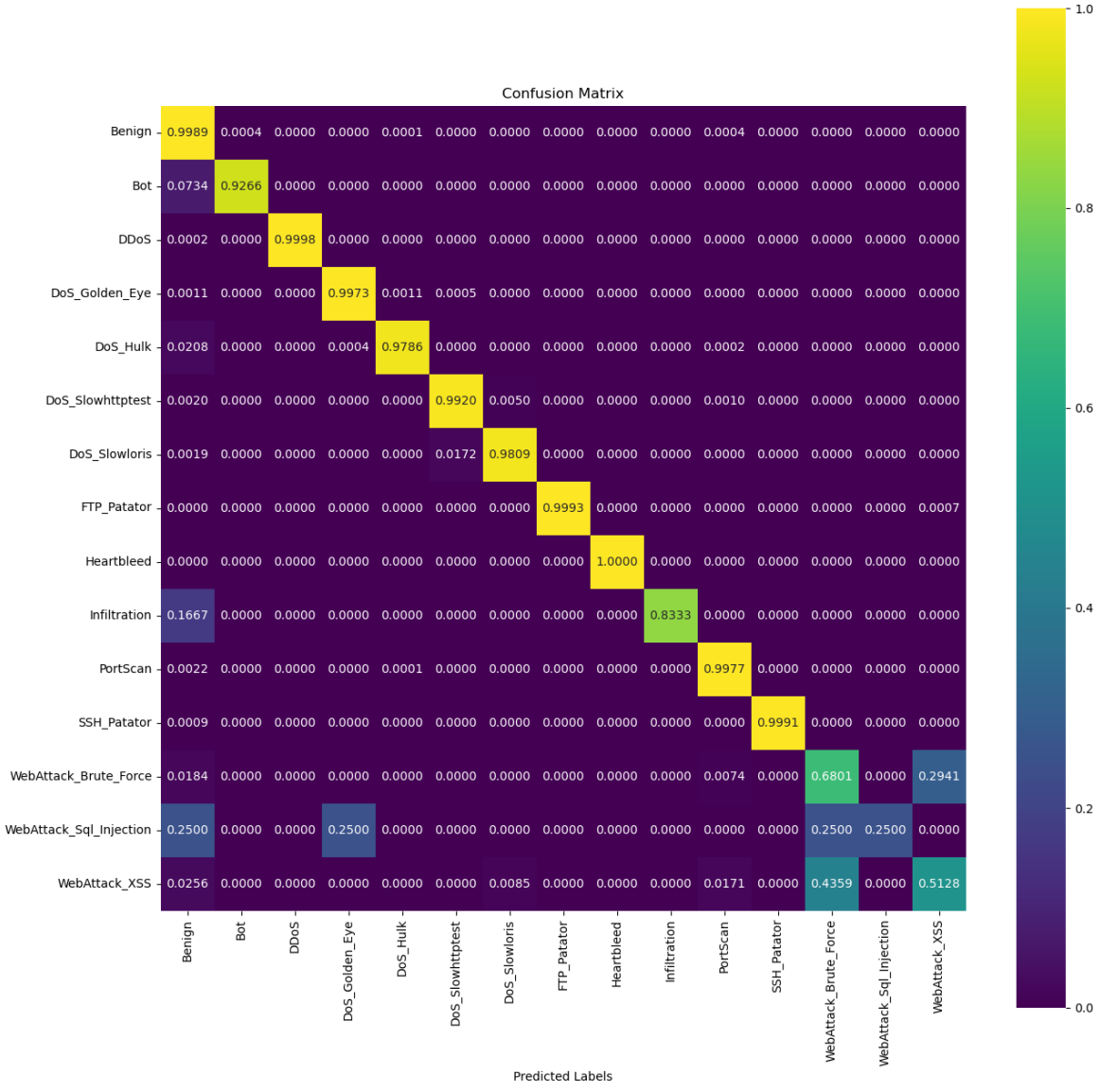

Fig. 4: DT Test Set Normalised Confusion Matrix when model was trained with the reduced resampled dataset