Automatic DDoS Attack Detection on SDNs

Preprint

Sabrina Corsetti,[1] Avi Purkayastha,[2] Adarsh Hasandka,[2] and Michael Samon[3]

1 University of Michigan
2 National Renewable Energy Laboratory
3 University of Colorado

Presented at the 2021 International Conference on Security and Management (SAM'21)
Las Vegas, Nevada
July 26 – 29, 2021

NREL is a national laboratory of the U.S. Department of Energy
Office of Energy Efficiency & Renewable Energy
Operated by the Alliance for Sustainable Energy, LLC

This report is available at no cost from the National Renewable Energy Laboratory (NREL) at www.nrel.gov/publications.

Contract No. DE-AC36-08GO28308

Conference Paper
NREL/CP-2C00-81041
September 2022

Automatic DDoS Attack Detection on SDNs

Preprint

Sabrina Corsetti,[1] Avi Purkayastha,[2] Adarsh Hasandka,[2] and Michael Samon[3]

1 University of Michigan
2 National Renewable Energy Laboratory
3 University of Colorado

Suggested Citation
Corsetti, Sabrina, Avi Purkayastha, Adarsh Hasandka, and Michael Samon. 2022. *Automatic DDoS Attack Detection on SDNs: Preprint*. Golden, CO: National Renewable Energy Laboratory. NREL/CP-2C00-81041. https://www.nrel.gov/docs/fy22osti/81041.pdf.

NREL is a national laboratory of the U.S. Department of Energy
Office of Energy Efficiency & Renewable Energy
Operated by the Alliance for Sustainable Energy, LLC

This report is available at no cost from the National Renewable Energy Laboratory (NREL) at www.nrel.gov/publications.

Contract No. DE-AC36-08GO28308

Conference Paper
NREL/CP-2C00-81041
September 2022

National Renewable Energy Laboratory
15013 Denver West Parkway
Golden, CO 80401
303-275-3000 • www.nrel.gov

NOTICE

This work was authored in part by the National Renewable Energy Laboratory, operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Science Undergraduate Laboratory Internships Program (SULI). The views expressed herein do not necessarily represent the views of the DOE or the U.S. Government. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes.

Automatic DDoS Attack Detection on SDNs

Sabrina Corsetti[1]†*, Avi Purkayastha[2†‡], Adarsh Hasandka[2], and Michael Samon[3]

[1] University of Michigan, Ann Arbor, MI 48109
sabcorse@umich.edu
[2] National Renewable Energy Laboratory, Golden, CO 80401
{avi.purkayastha,adarsh.hasandka}@nrel.gov
[3] University of Colorado, Boulder CO 80309
michael.samon@colorado.edu

Abstract. Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks pose a serious threat to computing networks - especially to critical systems within the U.S. electrical grid. As attack mechanisms have increased in complexity and variety, more sophisticated detection mechanisms have become necessary to ensure network security. This paper explores the use of artificial intelligence to automate the process of detection and mitigation of DoS and DDoS attacks within the framework of Software-Defined Networking (SDN), to a high degree. Machine learning algorithms are trained to recognize DoS and DDoS attacks and are deployed in real-time to mitigate malicious network traffic. The results show a well-tuned gradient-boosted decision tree detecting DoS and DDoS attacks, as well as initial successful mitigation of attacks within an SDN framework.

Keywords: Denial of Service, Cyber Detection, Machine Learning

1 Introduction

Software-Defined Networking (SDN) is a novel technology that decouples network controls and forwarding. This structure allows for flexible, automated network construction, and it facilitates network management. These benefits, coupled with low operational costs, make SDN architecture a highly desirable tool for modern networking.

As SDNs have become more prevalent, protecting them from common network threats, like DoS and DDoS attacks, has become increasingly important,

*†This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Science Undergraduate Laboratory Internships Program (SULI).

†Corresponding Author

‡A portion of this research was performed using computational resources sponsored by the Department of Energy's Office of Energy Efficiency and Renewable Energy and located at the National Renewable Energy Laboratory.

as addressed in [8, 15]. The target of this study is the detection of these prevalent network attacks within an SDN framework using Machine Learning (ML) classifiers, as well as the implementation of initial mitigation steps.

DoS and DDoS attacks make machine or network resources unavailable to legitimate users by using streams of traffic to consume network resources such as bandwidth and memory. These attacks can be executed via a variety of protocols, including UDP, TCP, and HTTP, among others as listed in [5]. It is well known that these attacks have increased many-fold in recent years. This has been researched broadly, including by [20], while using ML classifiers as a detection mechanism for such attacks has been shown in studies such as [14]. While many studies have achieved success using certain classifiers such as Random Forests (RF), this study aims to show that there is room for accuracy improvements through the use of enhanced classification algorithms, such as gradient-boosted decision trees, in disarming DoS/DDoS attacks. In addition, using a rich, high-variance dataset would also add robustness to these algorithms.

This study was designed as an extension of these previous works in an effort to further explore ML approaches in the classification and obstruction of assorted DoS/DDoS attacks on SDN architecture. After initially implementing the RF classification and performance testing scripts outlined in [15], several additions were made: a randomized RF hyperparameter scan, repeated stratified k-fold cross-validation, and the addition of new classification variables. This study also trained and tested the involved modules on the dataset used in [14] in an attempt to better reflect real-world performance.

Following the optimization of the RF model and the subsequent optimization of a comparative gradient-boosted decision tree classifier, a traffic-generating script was used to create both attack and non-attack traffic within a simulated SDN. This study found that although random forests and gradient-boosted classifiers performed comparably, well-tuned gradient-boosted algorithms outperformed the best RF classifiers. This study also laid the groundwork for a real-time DoS/DDoS attack mitigation mechanism on a simulated SDN and provided a baseline for future investigations.

In the following section, we first review some of the basic background material which forms the basis for this study, as well as a broad literature search on related studies. Then in the following section, we look more extensively at the datasets used and the classifiers considered, as well as the implementation of the SDN architecture. We lay out the definitions of the metrics for the data analysis in the next section and conclude with all the different and varied results which we set out to explore, as well as initial attempts at mitigation. Finally, we lay out our future research plans.

2 Background and Related Work

2.1 Software-defined networking

In a traditional network, switches independently determine where to send packets. In other words, traditional switches assume a dual controlling/forwarding

role. Software Defined Network (SDN) architecture offers a fundamentally different approach to network topology, in that it separates network controls and forwarding, as shown in Fig. 1. In an SDN, switches act only as forwarding devices and receive packet-handling guidance from a controller: a software application on a remote server, as explained in [8]. As with traditional networks, SDN topology can vary widely, with the number and arrangement of controllers, switches, and hosts in a network chosen to meet the needs of its users. Due to the centralization of SDN controls, SDNs are capable of dynamic load-balancing and resource scaling. This makes them ideal for high-bandwidth applications. The programmable nature of SDNs also allows for dynamic management and adaptable security.

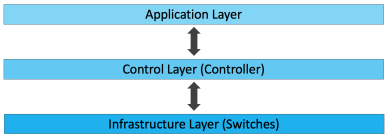

Fig. 1: *Basic SDN structure. Applications interact with and manage the controller, which provides forwarding guidelines to switches.*

As SDNs have grown in popularity, securing them from attacks has become a high priority. While SDN control centralization can facilitate attacks intended to overwhelm network resources, this same centralization can be advantageous for the streamlined detection and mitigation of attacks. Specifically, the decoupled nature of SDN control and forwarding planes has created the possibility for new methods of attack detection and obstruction – namely, those that take advantage of a controller's "bird's-eye" view of a whole network. By probing all traffic for attacks using machine learning and deploying mitigation software when necessary, a single controller can act as both a surveillance and a defense system for an entire SDN.

2.2 DoS and DDoS attacks

Easy to execute and difficult to mitigate, DoS and DDoS attacks are a common menace to modern networks. Although these attacks come in many forms, they share a goal: the consumption of network resources. By forcing a network to allocate resources to the processing of attack traffic, DoS/DDoS attacks inhibit network usage for legitimate users. DDoS attacks are especially effective, as they initiate DoS attacks from multiple hosts at once.

These attacks, as explained in [5], can be divided into several classes: volumetric, protocol, and application layer. Volumetric attacks consume bandwidth; protocol attacks consume server or intermediate equipment resources; application layer attacks crash web servers with seemingly innocuous connection requests. Among these, some of the most common are UDP floods (volumetric), TCP SYN floods (protocol), and HTTP floods (application).

2.3 Machine learning classifiers

Given the capability of DoS and DDoS attack mechanisms to evolve rapidly, classical detection software can struggle to identify new attacks. In an attempt to create detection mechanisms that are robust to changes in attack methods,

researchers have increasingly turned to machine learning as a resource for determining the underlying patterns in attack traffic – [14].

Classical methods of attack detection struggle to identify new attacks due to their reliance on rigid, predefined statistical expectations for threatening network patterns. Machine learning algorithms, on the other hand, are advantageous due to their ability to make inferences about previously unencountered data points based on patterns in familiar datasets.

Since network attack detection is a classification problem involving the separation of attack from benign network traffic, machine learning classifiers such as random forests (RFs) and gradient-boosted decision trees were of particular interest for this study. As shown in Fig. 2, RFs and gradient-boosted classifiers both rely on decision trees to make classifications. While RFs randomly generate trees in parallel, gradient-boosted classifier trees are built in series so that each tree can learn from previous mistakes, as explained in [9, 12].

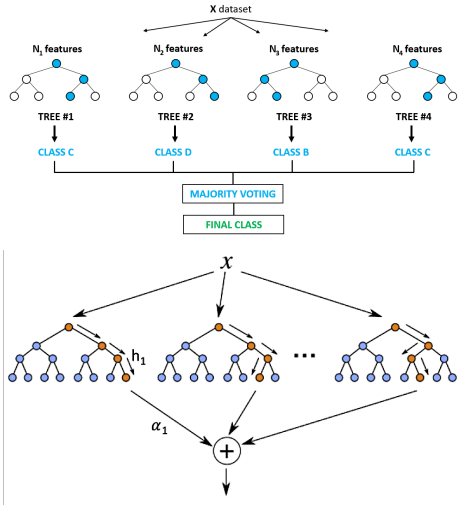

Fig. 2: Top: *Random forest schematics exemplifying majority voting.* Bottom: *Gradient-boosted ensemble schematics.*

Each of these algorithms relies on implementation in two stages: training and testing. During training, network flow data is input into the algorithm, and it learns to recognize what makes a flow an attack. During testing, the performance of the algorithm is tested on a separate dataset. New data is input into the algorithm, and it predicts whether or not each flow is an attack. The algorithm's predictions for each flow are then compared to the true classifications, and the resulting accuracy is used to gauge how well the algorithm can classify previously unencountered data.

2.4 Related investigations

The vulnerability of network structures to DoS and DDoS attacks has been previously investigated in a variety of contexts in [20], [14], [15] and [17] among others. In 2018, [15] introduced the usage of RFs for the detection of DDoS attacks within an SDN framework. The study explored the detection accuracy of RFs in comparison to a dynamic statistical method and a support vector machine (SVM) machine learning implementation.

Although [15] concluded that RFs outperformed the other methods in question, achieving more than 97% detection accuracy, further room was left for algorithm and dataset improvement. In particular, the RF used in the study was trained using minimal tuning and a lack of constructed variables (variables

created out of operations on other variables), although these variables have the potential to significantly strengthen classifier performance – [18].

In addition, the dataset used in the experiment was a relatively low-variance UDP flood dataset (Fig. 3), supplemented by independently-generated data. As indicated by [14], the use of such a dataset could lead to a predictor that performs well during training and testing, yet weaker when confronted with higher variance real-world attacks. In the place of similar low-variance datasets, sets containing realistic benign and attack traffic across a variety of protocols are preferable for achieving comparable detection accuracy (more than 96%, in the case of [14]) within a more broadly applicable framework.

	A	B	C	D	E	F	G	H
1	packet_time	srcIP	destIP	srcPort	destPort	U	bytes	packets
2	0.015675	1.1.139.109	1.1.236.8	9998	2	U	1001	1
3	0.015674	1.1.139.167	1.1.236.8	9997	3	U	1001	1
4	0.01574	1.1.139.92	1.1.236.8	9996	4	U	1001	1
5	0.015824	1.1.139.210	1.1.236.8	9995	5	U	1001	1
6	0.015945	1.1.139.71	1.1.236.8	9994	6	U	1001	1
7	0.016028	1.1.139.142	1.1.236.8	9993	7	U	1001	1
8	0.01612	1.1.139.131	1.1.236.8	9992	8	U	1001	1
9	0.0162	1.1.139.84	1.1.236.8	9991	9	U	1001	1
10	0.016283	1.1.139.80	1.1.236.8	9990	10	U	1001	1
11	0.016373	1.1.139.214	1.1.236.8	9989	11	U	1001	1
12	0.016457	1.1.139.80	1.1.236.8	9988	12	U	1001	1
13	0.01654	1.1.139.61	1.1.236.8	9987	13	U	1001	1
14	0.016621	1.1.139.91	1.1.236.8	9986	14	U	1001	1

Fig. 3: *A snapshot of the dataset used in [15], demonstrating its low duration, byte count, and packet count variance.*

3 Materials and Methods

3.1 Dataset preparation

Two datasets were prepared for this study. While our initial investigations began with UDP floods, as in [15], the richness and complexity of TCP packet data coupled with their prevalence in NREL DDoS attack data led us to make TCP floods a primary focus of this study. Thus, our first dataset consisted of exclusively TCP SYN flood traces and "clean", or legitimate, TCP traces. The second dataset was generated by the Canadian Institute for Cybersecurity (CIC). It consisted of a variety of DoS and DDoS attacks against a multi-protocol clean traffic background. To prevent discrepancies in variance across separate dataset regions, the order of each set's entries was randomly shuffled. Each dataset was divided into 80% for training, 10% for validation, and 10% for testing, advised by the Pareto principle – [16].

TCP dataset The first dataset was acquired by the National Renewable Energy Lab. The data consisted of 604,589 TCP SYN flood netflow traces and 8,162,905 legitimate TCP netflow traces. This dataset was significantly "imbalanced", as it consisted of almost entirely legitimate traces. Typically, imbalanced datasets are preferable for classification scenarios with unequal class frequencies, as they contain information about relative appearance probabilities. However, for the classification problem of attack versus benign traffic in which both high accuracy and recall are prioritized, a balanced dataset can help the algorithm better recognize the low-frequency attack class, thus raising its recall for threatening traffic. For comparative purposes we generated a relatively balanced subset

consisting of all 604,589 attack traces and 1,028,362 legitimate traces. For classification, attack traces were labelled with a 1, and legitimate traces were labelled with a 0, as shown in [3].

While numerous netflow classification variables were available, such as source and destination IP addresses and ports, only a select few were relevant to classification: duration (ms), payload (bytes), and packet count. From these values, two other variables were constructed: payload per packet and duration per packet.

CIC dataset The second dataset was adapted from the CIC's IDS2017 dataset, used in [14]. This dataset has been widely used for classification studies due to its wide assortment of attack classes and its proprietary realistic background simulation. The dataset offers a selection of more than 80 classification from variables per bidirectional flow, flow duration and payload to low-level characteristics like the maximum packet payload of a given flow, as shown in [19].

	A	B	C	D	E	F
1	duration	bytes	packets	results	s_per_packet	bytes_per_packet
2	5	2901	16	0	0.3125	181.3125
3	0	54	1	1	0	54
4	2248	2391	11	0	204.3636364	217.3636364
5	0	60	1	1	0	60
6	1023	148	2	0	511.5	74
7	6	934	8	0	0.75	116.75
8	1	432	5	0	0.2	86.4
9	57	13322	200	0	0.285	66.61
10	10	2435	14	0	0.714285714	173.9285714
11	4	1908	8	0	0.5	238.5
12	0	58	1	1	0	58
13	48	13057	196	0	0.244897959	66.61734694
14	3	1907	8	0	0.375	238.375

Fig. 4: *A snapshot of the TCP data sample.*

The first step in adapting the IDS2017 dataset for this study was isolating DoS and DDoS traces from other attacks. This was accomplished by using data exclusively from the Wednesday Working Hours and the Friday Afternoon DDoS files. Within these files, "benign" and "attack" trace labels were converted to 0 and 1 respectively, for classification. Then, to ensure the compatibility of the dataset with netflow data capture, each bidirectional flow was separated into two unidirectional flows. 0-packet "backward" flows were eliminated from the dataset.

	A	B	C	D	E	F
1	duration	bytes	bytes_per_packet	results	packets_per_second	packets
2	5.063925	383	42.55555556	0	1.777277507	9
3	0.024066	452	226	0	83.10479515	2
4	0.573531	26	8.666666667	1	5.230754746	3
5	4.13186801	46	46	0	0.242021284	1
6	0.062457	204	102	0	32.02203116	2
7	0.33681	218	109	0	5.938065972	2
8	0.000962	0	0	1	6237.006237	6
9	0.000199	170	85	0	10050.25126	2
10	0.025155	234	117	0	79.50705625	2
11	1.00E-06	0	0	1	2000000	2
12	0.155199	55319	1229.311111	0	289.9503218	45
13	0.434965	4195	466.1111111	0	20.69131999	9
14	71.7019415	11601	2900.25	1	0.055786495	4
15	0.050753	0	0	0	39.40653754	2

Fig. 5: *A snapshot of the wider DoS/DDoS dataset.*

Following this preparation, the classification variable list was trimmed and constructed for netflow compatibility, as shown in Fig. 5. The variables kept from the original set were the number of packets, the payload in bytes, the rate of packets per second, and the rate of bytes per packet. From the packet count and packet/second rate, a "duration" variable was constructed. Any infinite duration values (resulting from division by 0) were cleared from the dataset. The end result was a relatively balanced dataset with 1,005,995 legitimate background traces and 640,417 attack traces.

3.2 Proposed classifiers

For this study, two different decision tree-based ML classifications were used. Random Forests (RFs) are an ensemble learning method known for providing high-quality classification results with minimal user-end tuning – [9]. Although RFs perform well with default settings, this study explored hyperparameter tuning as a method for boosting their performance.

Gradient boosting (GB) takes a more refined approach to decision tree-based classification. GB constructs decision trees iteratively, with each tree constructed such that it "learns" from previous trees' mistakes. GB typically performs better than RFs, but only after more intensive hyperparameter tuning. A popular Python implementation of gradient-boosted decision trees is the XGBoost (XGB) package, used in this study – [12]. A systematic approach was followed in order to ensure the optimal hyperparameter tuning of the XGB classifiers used in this study – [1].

3.3 SDN implementation

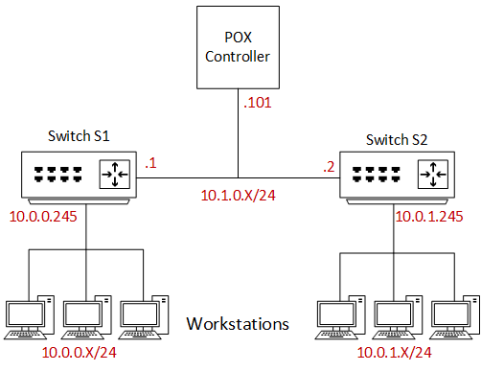

Fig. 6: *A network diagram of the implemented network within CEEP*

The SDN used in this experiment was simulated within NREL's CEEP system infrastructure – [4]. The network was simulated using Minimega, a virtual machine launching and management tool – [7]. The simulated network consisted of one controller, two switches, and 20 hosts (10 per switch), as shown in Fig. 6. The controller and hosts operated under Operating System (OS) Ubuntu 18.04.4 LTS (Bionic Beaver), and the switches operated under Cumulus Linux 4.1.1 OS.

Although Cumulus Linux is not a traditional SDN switch OS, its adaptability in terms of creation of multi-layered hierarchical network design enables the implementation of SDN solutions, as explained in [2]. It enabled the key SDN features required for this study: traffic monitoring, switch-to-controller and host-to-host data transmission, and attack detection and mitigation. Fig. 7 shows the the visualization of the network with both benign and attack traces, the details of which are discussed in the results section.

Several scripts were used to facilitate the detection and mitigation of DoS and DDoS attacks, all available in the project git repository – [3]. First, an "fprobe" command was executed on each switch to initiate real-time traffic capture. This data was passed to the controller, where it was captured as netflow data. Finally, this data was parsed and tested for attacks in real-time, using a Python script

on the controller. In the event of an attack, this script used iptables to remotely disable traffic from an offending IP address.

In order to gauge the detection and mitigation capabilities of the real-time machine learning testing script, DDoS attacks were simulated using the Low Orbit Ion Cannon application – [6]. The script sent randomly generated TCP and UDP flood attack packets to a specific port on a victim machine, using IP address spoofing – a process in which a computer masks its own IP address with another to make an attack appear as though it came from a trusted source.

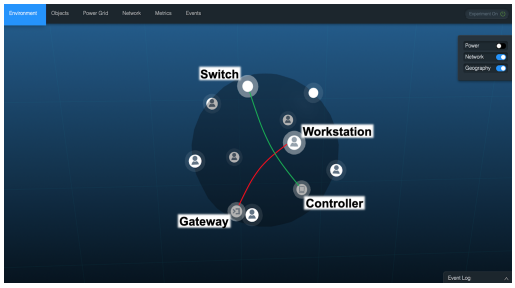

Fig. 7: Snapshot of the SDN solutions implemented for this study. *Red line: Attack transmitted from a spoofed "gateway" IP address to a victim workstation.* Green line: Benign switch-to-controller data transmission.

4 Data Analysis

4.1 Machine learning performance metrics

Accuracy, Precision, and Recall The primary figures used to evaluate a learning machine's performance are true and false positives and negatives, where a "positive" in this study is an attack flow, and a "negative" is a non-attack flow. A true positive (TP) denotes a positive accurately predicted as a positive; a true negative (TN) denotes a negative accurately predicted as a negative; a false positive (FP) denotes a negative inaccurately predicted as a positive; a false negative (FN) denotes a positive inaccurately predicted as a negative.

These figures can be used to compute several metrics, including accuracy, precision, and recall. Accuracy gives an overview of prediction correctness, precision gives the proportion of true positives to reported positives, and recall gives the proportion of accurately reported positives [10].

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

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

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

The final model implemented in this study was refined using these metrics, with confusion matrices providing TP, FP, TN, and FN counts. While maximal accuracy was given high priority, high recall was an equally significant metric. This is because, due to the high cost of not identifying an attack, the primary goal of the model was to identify as many attacks as possible at the cost of minimal false positives.

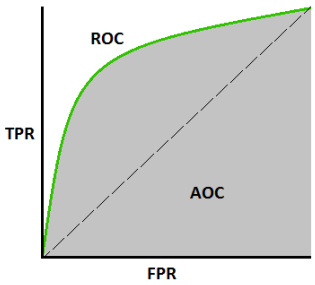

Fig. 8: *Sample ROC – [13].*

ROC AUC Another metric used to evaluate binary classifiers is the Area Under the Curve (AUC) for the Receiver Operating Characteristic (ROC) Curve, where a higher ROC AUC indicates a stronger ability to distinguish between classes. ROC curves plot True Positive Rate (TPR) vs. False Positive Rate (FPR), so an ideal ROC AUC is 1, as shown in Fig. 8 and explained in [13]. This indicates that the classifier has a 100% chance of distinguishing between positive and negative classes. Equations 4 and 5 provide the TPR and FPR formulas.

$$TPR = \frac{TP}{TP + FN} \quad (4) \qquad FPR = \frac{FP}{TN + FP} \quad (5)$$

5 Results and Discussion

5.1 TCP results

The first round of tests was performed on an abbreviated, 100,000-point, randomly chosen subset of the balanced NREL TCP dataset (Table 1). Three RFs were trained on this dataset, each on a unique set of classification variables. As a baseline for comparison, the first set included the three relevant variables used to identify netflow traces: duration (ms), packet count, and payload (bytes). To probe for differences in non-linear relationships between benign and attack traffic, we added the seconds/packet variable to the second set. For the same reason, we added both the seconds/packet and the bytes/packet variables to the third set. The final five-variable combination was found to be optimal, indicating that the nonlinear relationships between packet count, duration, and payload differed significantly between benign and attack traffic.

Table 1: *Variable Set Performance Comparison. Variable Key: P = Payload, D = Duration, PC = Packet Count, D/P = Duration/Packet, P/P = Payload/Packet*

Algorithm	Variables	Accuracy	Precision	Recall	ROC AUC
Random Forest	P, D, PC	78.98 ± 0.31	93.49 ± 0.63	46.56 ± 0.76	0.80
Random Forest	P, D, PC, D/P	79.08 ± 0.29	93.19 ± 0.59	47.04 ± 0.74	0.80
Random Forest	**P, D, PC, D/P, P/P**	**79.70 ± 0.28**	**94.96 ± 0.32**	**47.79 ± 0.79**	**0.81**

Once the classification variable list was optimized, a comparison was made between classification on the full imbalanced and balanced NREL TCP datasets

(Table 2). Although classification accuracy drastically increased on the imbalanced dataset, this was a byproduct of decreased attack prevalence. Virtually every trace was labelled as legitimate, rendering the classifier ineffective. Thus, the balanced dataset was chosen as the final dataset for RF optimization.

Table 2: *Balanced vs. Imbalanced Data Performance Comparison.*

Algorithm	Balance	Accuracy	Precision	Recall	ROC AUC
Random Forest	Imbalanced	93.67 ± 0.02	82.08 ± 0.85	10.75 ± 0.31	0.81
Random Forest	**Balanced**	**81.90 ± 0.10**	**94.98 ± 0.15**	**53.98 ± 0.24**	**0.84**

With an optimal RF classifier pinned down, a comparison was made against the XGBoost classifier, trained on the five-variable set (Table 3). Across all metrics, the XGBoost classifier outperformed the best random forest.

Table 3: *RF vs. XGB Performance Comparison.*

Algorithm	Accuracy	Precision	Recall	ROC AUC
Random Forest	81.90 ± 0.10	94.98 ± 0.15	53.98 ± 0.24	0.84
XGBoost	**83.43**	**95.25**	**58.19**	**0.85**

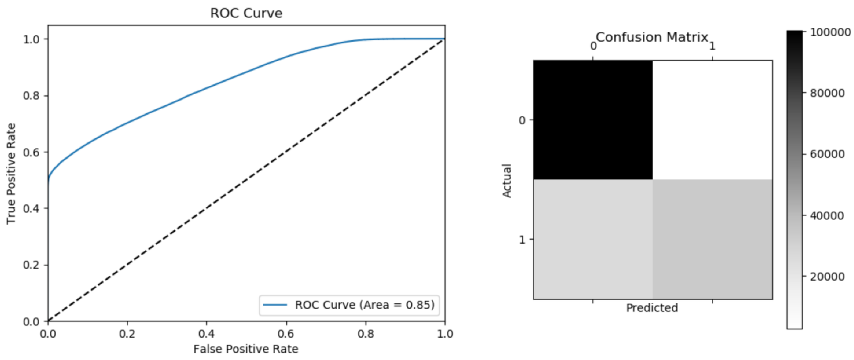

Fig. 9: *Testing set performance metrics for XGBoost classifier. Left: ROC curve. Right: confusion matrix (0: non-attack, 1: attack).*

5.2 Generalized DoS/DDoS results

Once the TCP-exclusive tests were completed, the optimization rounds were reproduced for the CIC IDS2017 dataset (Table 4). This time, the first round of RF tests was conducted to compare classification based on two variable sets. As before, the first was the duration, packet count, and payload set. The second paired these variables with the packets/second and bytes/packet variables. The difference in performance this time was virtually nonexistent between the classi-

fiers, indicating that the initial three variables differed significantly between the dataset's attack and non-attack traffic.

Table 4: *Variable Set Performance Comparison. Variable Key: P = Payload, D = Duration, PC = Packet Count, D/P = Duration/Packet, P/P = Payload/Packet*

Algorithm	**Variables**	**Accuracy**	**Precision**	**Recall**	**ROC AUC**
Random Forest	P, D, PC	97.15 ± 0.04	95.60 ± 0.07	97.14 ± 0.08	0.99
Random Forest	P, D, PC, D/P, P/P	97.15 ± 0.04	95.57 ± 0.07	97.17 ± 0.08	0.99

With the RF optimization complete, an XGBoost classifier was again trained on the five-variable dataset. The five-variable dataset was selected in order to capitalize on all available netflow data. A balanced/imbalanced dataset comparison was not made for this dataset due to its innately balanced nature. This time, the difference between the XGBoost and RF classifiers was less pronounced than between the TCP-only classifiers. However, a non-negligible increase in recall was observed. As time and resource efficiency are comparable for prediction-making by trained RF and XGBoost algorithms, this increase in recall made the XGBoost classifier the better choice for real-time attack detection. In general, the CIC DoS/DDoS classifiers performed significantly better than the TCP-exclusive classifiers, likely due to the exemplary quality of the CIC IDS2017 dataset.

Table 5: *RF vs. XGB Performance Comparison.*

Algorithm	**Accuracy**	**Precision**	**Recall**	**ROC AUC**
Random Forest	97.15 ± 0.04	95.57 ± 0.07	97.17 ± 0.08	0.99
XGBoost	**97.66**	**95.33**	**98.81**	**0.99**

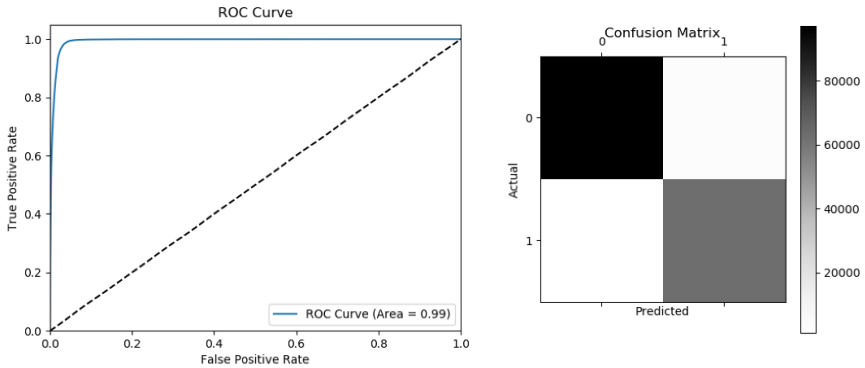

Fig. 10: *Testing set performance metrics for XGBoost classifier. Left: ROC curve. Right: confusion matrix (0: non-attack, 1: attack).*

5.3 Experimental Result Visualization

Through the CEE Platform used to deploy the simulated SDN, it was possible to visualize benign traffic passing through the network, as well as attack alerts signalled by the machine learning algorithm, in real-time.

On the SDN, attacks were generated using the LOIC application. A serialized version of the trained XGB machine learning algorithm was loaded onto the controller and deployed to detect them. As the model was precompiled, predictions were rapid, occurring in under a second. However, performance on any specific machine depends on computational resources.

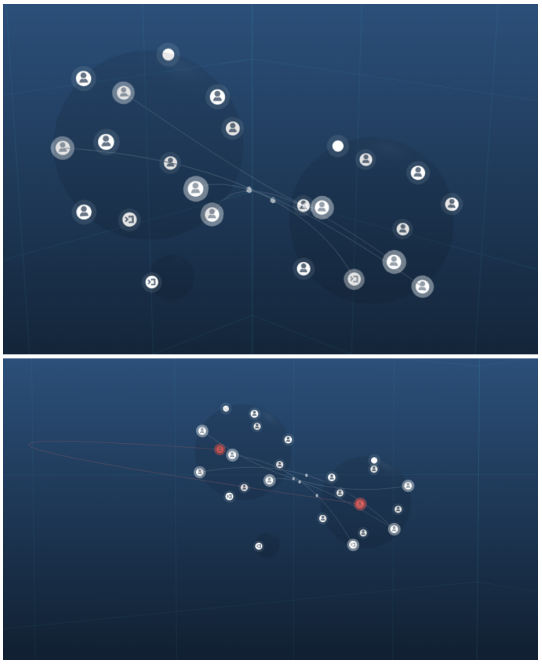

Fig. 11: Top: *Visualization with normal traffic flow. Bottom: Visualization of alert (in red) simultaneously with normal traffic*

Upon the detection of an attack by the algorithm, the following preliminary mitigation steps were taken: when malicious traces were identified by the algorithm, its containing script would produce output with information about the attack's source. Using this information, a signature-based rule would be created for Suricata [11], an open-source Intrusion Detection System (IDS), deployed in the platform back-end.

This IDS instance was connected to the common tap interface and could scan all communications in the platform in order to identify the network flow instance representing the malicious packets. The identified flow was then handled by the alerts module to visualize the malicious flow in real-time overlaid over benign background traffic.

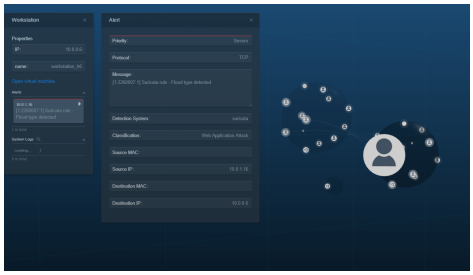

Fig. 12: *Detailed logs of alert events available in platform*

In case a real-time visualization was missed, a detailed log of each event was available on the platform's GUI, as shown in Fig. 12. The missed

event could be compared to similar events in order to verify correct operation of the malicious flow identification. These alerts were also simultaneously logged in the experiment's historical database, as shown in Fig. 13. Thus, all traffic information could be analyzed post-experiment by exporting this dataset.

This experiment served to prove that the rule developed on the identified malicious flow was capable of correctly identifying the attack. This leads to the conclusion that containerization and deployment of the algorithm in the platform backend, similar to the deployment of Suricata used for the visualization of alerts, would allow for tighter integration with the CEE Platform in general. In addition, as the XGB attack detection algorithm was agnostic to network structure, in that it could scan traffic streams on a controller in any SDN for attacks, these solutions could be implemented for an SDN of any topology.

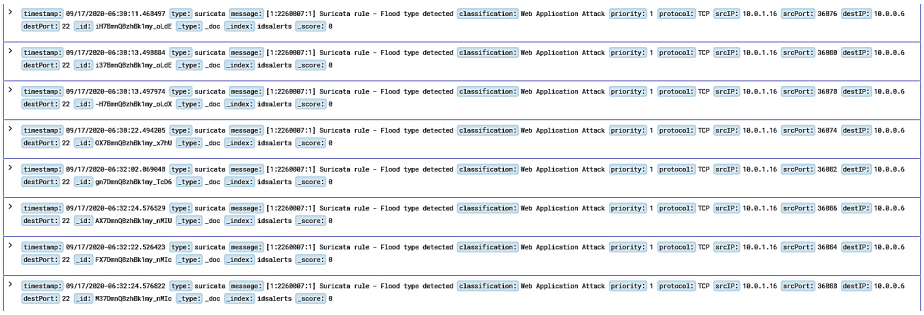

Fig. 13: *Some alert logs available in historical database (elasticsearch)*

6 Conclusions and Future Research

This study capitalized on SDN control centralization to develop single machine learning algorithms capable of detecting DoS and DDoS attacks targeting any host, or combination of hosts, in an SDN. Optimal machine learning classifiers were determined for DoS and DDoS attacks in both a targeted (TCP-specific) and a general (multi-protocol) case, with general case algorithm accuracy $> 97\%$, rivaling that of competing single-protocol algorithms.

The study also demonstrated the favorable effects of using modern enhanced classification-based decision tree methods, such as gradient-boosted decision trees. While random forests require little tuning to produce high-accuracy classifications, the additional tuning opportunities available for gradient-boosted decision trees provided an avenue for increased detection performance in this study, providing a $> 1\%$ increase in recall in the multi-protocol case.

Finally, this study laid the groundwork for DoS and DDoS attack mitigation within an SDN framework and provided guidance for future research. Specifically, the creation of a single gradient-boosted decision tree algorithm capable of detecting DoS and DDoS attacks across a variety of protocols has provided a foundation upon which attacks from additional protocols can be detected.

Likewise, the implementation of a scalable simulated SDN environment with preliminary mitigation measures has provided a foundation for the construction of more complex network topologies and more robust mitigation measures.

References

1. Analytics Vidhya: *Complete Guide to Parameter Tuning in XGBoost with codes in Python*, https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python
2. Cumulus, *Linux networking: it's not just SDN*
3. Github: *mlddos*, https://github.nrel.gov/apurkaya/mlddos
4. Gitlab: *Single-node-ceep*, ceep-services.nrel.gov/gitlab/ceep-systems/single-node-ceep
5. Imperva, *DDoS Attacks*, www.imperva.com/learn/application-security/ddos-attacks/
6. LOIC: *Low Orbit Ion Cannon*, en.wikipedia.org/wiki/Low_Orbit_Ion_Cannon
7. Minimega: *What is Minimega?*, https://minimega.org
8. Open Networking Foundation, *SDN Overview*, www.opennetworking.org/sdn-definition
9. Scikit-learn *3.2.4.3.1.sklearn.ensemble.RandomForestClassifier*, scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.htm
10. Towards Data Science: *What's the deal with Accuracy, Precision, and Recall?*, https://towardsdatascience.com/whats-the-deal-with-accuracy-precision-recall-and-f1-f5d8b4db1021
11. What is Suricata?, https://suricata.readthedocs.io/en/suricata-5.0.3/index.html
12. XGBoost, *XGBoost Documentation*, https://xgboost.readthedocs.io/en/latest/
13. Medium - towards data science, *Understanding AUC - ROC Curve*, https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5
14. Bindra, N., Sood, M.: Detecting ddos attacks using machine learning techniques and contemporary intrusion detection dataset. Automatic Control and Computer Sciences 53(5), 419–428 (Sep 2019), https://doi.org/10.3103/S0146411619050043
15. Dennis, M.J.R.: Machine-Learning and Statistical Methods for DDoS Attack Detection and Defense System in Software Defined Networks. Master's thesis, Ryerson University Digital Repository (2018)
16. Gen, M., Cheng, R.: Genetic Algorithms and Engineering Optimization. Wiley, Hoboken, NJ (1999)
17. Hajj, S., El Sibai, R., Bou Abdo, J., Demerjian, J., Makhoul, A., Guyeux, C.: Anomaly-based intrusion detection systems: The requirements, methods, measurements, and datasets. Transactions on Emerging Telecommunications Technologies 32(4), e4240 (2021), https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.4240
18. Kato, K., Klyuev, V.: An intelligent ddos attack detection system using packet analysis and support vector machine. International Journal of Intelligent Computing Research (IJICR) 5(3) (2014)
19. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization (2018)
20. Zekri, M., Kafhali, S.E., Aboutabit, N., Saadi, Y.: Ddos attack detection using machine learning techniques in cloud computing environments. In: 2017 3rd International Conference of Cloud Computing Technologies and Applications (CloudTech). pp. 1–7 (2017)