

Evaluation of Machine Learning Classifiers for Zero-Day Intrusion Detection – An Analysis on CIC-AWS-2018 dataset

Qianru Zhou and Dimitrios Pezaros

School of Computing Science, University of Glasgow, U.K.

Abstract. Detecting **Zero-Day** intrusions has been the goal of Cybersecurity, especially intrusion detection for a long time. Machine learning is believed to be the promising methodology to solve that problem, numerous models have been proposed but a practical solution is still yet to come, mainly due to the limitation caused by the out-of-date open datasets available. In this paper, we take a deep inspection of the flow-based statistical data generated by CICFlowMeter, with six most popular machine learning classification models for **Zero-Day** attacks detection. The training dataset CIC-AWS-2018 Dataset contains fourteen types of intrusions, while the testing datasets contains eight different types of attacks. The six classification models are evaluated and cross validated on CIC-AWS-2018 Dataset for their accuracy in terms of precision, recall, F1 value, and time overhead. Testing dataset, including eight novel (or **Zero-Day**) real-life attacks and benign traffic flows collected in real research production network are used to test the performance of the chosen decision tree classifier. Promising results are received with the accuracy as high as 100% and reasonable time overhead. We argue that with the statistical data collected from CICFlowMeter, simple machine learning models such as the decision tree classification could be able to take charge in detecting Zero-Day attacks.

Keywords: Intrusion detection, Zero-Day attacks, CIC-AWS-2018 dataset, cybersecurity, machine learning, decision tree classifier

1 Introduction

With the novel cyber attacks keep emerging, and the rapid extension of new communication protocols, which encrypts not only the user payload data but also scrambles the packet header information such as IP address and Port number [1], traditional intrusion detection methodologies which relies on finding and matching the patterns of packets headers information (especially IP address and Port number) are gradually losing their effectiveness. Thus adopting machine learning technologies to detect intrusions are more and more believed to be the future solution for intrusion detection [2–8]. As a technology of Artificial Intelligence, machine learning is well known by its capability to grasp hidden patterns from massive datasets and provide accurate prediction.

The performance of a machine learning algorithm is largely depends on the dataset it is trained on [2]. The majority of current machine learning based intrusion detection approaches are trained with DARPA 98/99¹, KDD CUP 99² datasets. However, the use of these datasets has become a serious issue and an increasing number of researchers recommend against their use [2, 8–12]. The CSE-CIC-IDS 2018 dataset³ collected on one of Amazon’s AWS LAN network (thus also known as the CIC-AWS-2018 Dataset) by the Canadian Institute of Cybersecurity (CIC) has gaining attention [13]. Besides the straightforward TCP/IP level traffic information such as IP address and port number, CIC-AWS datasets provides statistical traffic information based on flow, calculated by the network flow generator and analyser developed by CIC – CICFlowMeter.

There are six different intrusion scenarios in the dataset, Brute-force, Botnet, DoS, DDoS, Web attacks, and infiltration of the network from inside, with a total of 14 types of intrusions, namely, *Botnet attack*, *FTP-BruteForce*, *SSH-BruteForce*, *BruteForce-Web*, *BruteForce-XSS*, *SQL Injection*, *DDoS-HOIC attack*, *DDoS-LOIC-UDP attack*, *DDoS-LOIC-HTTP attacks*, *Infiltration*, *DoS-Hulk attack*, *DoS-SlowHTTPTest attack*, *DoS-GoldenEye attack*, and *DoS-Slowloris attack*. All the data are fully labeled, describing the statistical features of the traffic, e.g., flow duration, number of packets, number of times a certain flag was set in packets, total bytes used for the header in an upward flow, etc. The raw record of network traffic and event logs are also provided in CIC-AWS-2018 Dataset , although they are not discussed in this paper.

Although the intrusion detection studies using CIC-AWS-2018 Dataset has not yet been much reported, there are plenty research work has been done on an earlier version of CIC-AWS-2018 Dataset , CICIDS2017 [13–15]. Radford et.al [14] has applied unsupervised learning on CICIDS2017. The creator of CICIDS2017 dataset, Sharafaldin et.al from the CIC institute use RandomForestRegression to choose the top four features that can best describe each attacks, and apply machine learning based intrusion detection for them [13].

The paper is organised as follows. Section 2 provides detailed introduction of the datasets used in this paper, including the CIC-AWS-2018 Dataset which is the training dataset, and the **Zero-Day** intrusions collected online and benign data collected in real research production environment. The procedure of data laundry adopted in this paper is also presented. Section 3 describes the machine learning methodology adopted in this paper. The evaluation experiments and the results are analysed in detail in Section 4, including the cross validation process and results, and the testing results. Finally, Section 5 summaries the paper and provides our vision for future work.

¹ <https://www.ll.mit.edu/r-d/datasets>

² <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>

³ <https://registry.opendata.aws/cse-cic-ids2018/>

2 Dataset

2.1 Training Dataset: CIC-AWS-2018 Dataset

The features in CIC-AWS dataset are described in Table 1. There are 80 features in the dataset, providing statistical information of the flows from both uplink and downlink. Comparing to the straightforward TCP/IP traffic header informations provided by the previous datasets, like DARPA and KDD, it is widely believed that the statistics based on flow could provide more useful information for intrusion detection [16].

In order to get a general idea of the dataset, we plot a bar chart of each feature against the label. As redundant features will increase computation expense, induce chaos and reduce accuracy, we delete the features that does not show any difference between benign traffic and malicious one.

2.2 Test Dataset

We use real life traffic data as the source of test dataset. The test datasets are collected from mainly two different sources, the benign data and intrusion data.

Benign test data Benign data are collected from our real-life online surfing traffic collected from a typical research product network, it generated during the following daily online activities: emailing, searching (mainly on Google), reading news, watching video (through Netflix and Youtube), downloading paper from Google Scholar.

The data are collected for a week on our office desktop in a research daily routine environment, and then converted into flow-based statistical dataset consisting of 12,681 MB.

Intrusion test data To evaluate the ability of the machine learning models in detecting an attack that it has not seen before (or in other word, Zero-Day attacks), we collect novel real-life attacks traffic data containing eight new attack types with no repetitive with the training CIC-AWS-2018 Dataset . The attack types in the test data are listed in Table 2. This dataset is collected from most recent real life attacks or abnormal traffic that humans failed to detect and prevent, most of them are still active till nowadays, such as **ransom malware**, **DDoS Bot? a Darkness**, **Google doc macadocs**, and **Bitcoin Miner**(this is more like abnormal traffic rather than intrusions to many people).

2.3 Data Laundry

The following steps are adopted to laundry the CIC-AWS-2018 Dataset .

- Delete noisy features;
- Format data into standard datatype;

- To reduce the size of the datasets, reduce the unnecessary accuracy of the float numbers by dropping digits after the decimal point;
- Replace noisy, machine unprocessable chars by underline _ ;
- Replace “*Infinity*” and “*NaN*” value with suitable numbers.

After the laundry, the total size of training dataset has dropped from 6,886 MB to 4 MB, without losing valuable information.

Table 1. Features Used in CIC-AWS Dataset

Feature	Description	Type
Label	indicate whether the traffic is malicious or not, e.g., benign, SQL-Injection, etc.	string
Dst Port	Destination port number	integer
Protocol	Protocol	integer
TimeStamp	Time Stamp of the flow	string
Flow Duration	Flow duration	integer
Tot Fwd/Bwd Pkts	Total packets in forward/backward directions	integer
TotLen Fwd/Bwd Pkts	Total size of packets in forward/backward directions	integer
Fwd/Bwd Pkt Len Max/Min/Mean/Std	Maxi/Mini/Average/Std. Dev. size of package in forward/backward directions	integer
Flow Byts/s & Flow Pkts/s	Flow byte rate, i.e., number of packets per seconds	float64
Flow IAT Mean/Std/ Max/Min	Average/Std. Deviation/Maxi/Mini time between two flows	float64
Fwd/Bwd IAT Tot/Mean/ Std/-Max/Min	Total/Average/Std. Deviation/Maxi/Mini time between two packets in forward/backward directions	float64
Fwd/Bwd PSH/URG Flags	Number of times the PSH/URG flag was set in packets in forward/backward direction	integer
Fwd/Bwd Header Len	Total bytes used for header in forward/backward direction	integer
Fwd/Bwd Pkts/s	Number of forward/backward packets per second	float64
Pkt Len Min/Max/Mean/Std	Maxi/Mini/Average/Std. Dev. length of a flow	integer
Pkt Len Var	Mini inter-arrival time of packet	float64
FIN/SYN/RST/PUSH/ACK/URG/CWE/ECE Flag Cnt	Number of packets with FIN/SYN/RST/PUSH/ACK-/URG/CWE/ECE	integer
Down/Up Ratio	Download/upload ratio	integer
Pkt Size Avg	Average size of packets in forward/backward direction	float64
Fwd/Bwd Seg Size/Byts/b/Blk Rate Avg	Average number of bulk rate/bytes bulk rate/packets bulk rate in forward/backward directions	float64
Subflow Fwd/Bwd Pkts/Byts	The average number of bytes/packets in a sub flow in forward/backward direction	integer
Init Fwd/Bwd Win Byts	Number of bytes sent in initial window in forward/backward directions	integer
Fwd Act Data Pkts	Number of packets with at least 1 byte of TCP data payload in forward	integer
Fwd Seg Size Min	Minimum segment size observed in forward	integer
Active Mean/Std/Max/Min	Maxi/Mini/Average/Std. Dev. a flow was active before becoming idle	float64
Idle Mean/Std/Max/Min	Maxi/Mini/Average/Std. Dev. a flow was idle before becoming active	float64

Table 2. Attack Types included in Test Dataset

Attack Type	Description
Bitcoin Miner	Traffic generated during Bitcoin mining, maybe not a typical attack, but is treated as traffic blocker in production network.
Drowor worm	A virus in Windows operation system that infects portable executable (PE) files, such as those with EXE, DLL, and SYS files. It stops security processes from running, and overwrites some of their code, which means that you may have to reinstall affected security programs ⁴ .
Nuclear ransomware	A new version of file-encrypting virus that actively spreads in Hungary, Italy, and Iran. Crypto-malware uses a combination of RSA and AES encryption and appends <code>.[black.world@tuta.io].nuclear</code> extension. Criminals provide a ransom note in <code>HELP.hta</code> file and ask to contact <code>black.world@tuta.io</code> for more information.
False content injection	Some network operators inject false content into users' network traffic, the injected packets have identical IP address, port number, and TCP sequence numbers, but different payload.
Ponmocup trojan	A trojan in Windows operation system, which tries to download other malware from the Internet.
DDoS Bot'a Darkness	Containing four types of DDoS attacks, namely HTTP flood, ICMP flood, ping, SYN flood and UDP flood. Still under active development by Russian malware coders.
Google doc macadocs	A new variant of the Macadocs malware to be using Google docs as a proxy server and not connecting to a command and control (C&C) server directly.
ZeroAccess	A Trojan horse computer malware that affects Microsoft Windows operating systems. It is used to download other malware on an infected machine from a botnet while remaining hidden using rootkit techniques.

3 Machine Learning Classifiers

In our problem model, the task is that given a set of statistic information of a flow, identify whether this flow is benign traffic or intrusion, based learning on a set of already labelled data containing both benign and intrusion traffic, that makes our problem a *supervised classification* problem. To find the best classification model that suit into our problem, we run a exhaustive test of the six most commonly used machine learning classification models on the training dataset, comparing their performance using the criterias of *precision*, *recall*, *F1 score*, and *time expense*.

The supervised machine learning classification models tested are listed below.

- Random forest classifier
- Gaussian naive bayes classifier
- Decision tree classifier
- Multi-layer Perceptron (MLP) classifier
- K-nearest neighbours classifier
- Quadratic discriminant analysis classifier

The performance of each classification model will be analysed in detail during the cross validation process in Section 4.2.

4 Evaluation

This section presents the evaluation experiments setup and the evaluation results of intrusion detections. The evaluation experiments includes: 1) cross validation of the training CIC-AWS-2018 Dataset on each of the attack types, and 2) the prediction using testing data on the model chosen.

4.1 Environment

The software tool used in the evaluation experiments is *sklearn*⁵, *numpy*⁶, and *pandas*⁷. All the evaluation experiments are carried on a Dell server working at 3.4 GHz on 8 cores AMD64 CPU, with 16 GB memory and 1 TB hardware disk.

4.2 Cross Validation Results

To test the fitness of each machine learning classification models in prediction, we run a cross validation on each of the fourteen types of attacks in the training CIC-AWS-2018 Dataset , and the results are shown in detail in Table 3, 4 , 5, 6, 7, 8, 9, 10 , 11, 12, and 13.

⁵ <https://scikit-learn.org/>

⁶ <https://www.numpy.org/>

⁷ <https://pandas.pydata.org/>

Table 3. Machine Learning Results of Botnet Attack on Different Classification Models.

		Random Forest	Naive Bayes	Decision Tree	Neural Network (MLP)	Quadratic Discriminante	KNeighbors
True-Positive		98%	52%	100%	100%	100%	100%
False-Positive		0%	0%	0%	0%	0%	0%
recall	Benigh	1.00	0.65	1.0	1.0	1.0	1.0
	Bot	1.00	1.0	1.0	1.0	1.0	1.0
f1-score	Benigh	1.00	0.79	1.0	1.0	1.0	1.0
	Bot	0.99	0.68	1.0	1.0	1.0	1.0

Table 4. Machine Learning Results of FTP-BruteForce Attack on Different Classification Models.

		Random Forest	Naive Bayes	Decision Tree	Neural Network (MLP)	Quadratic Discriminante	KNeighbors
True-Positive		100%	100%	100%	100%	100%	-
False-Positive		0%	0%	0%	0%	0%	-
recall	Benigh	1.0	1.0	1.0	1.0	1.0	-
	FTP-BruteForce	1.0	1.0	1.0	1.0	1.0	-
f1-score	Benigh	1.0	1.0	1.0	1.0	1.0	-
	FTP-BruteForce	1.0	1.0	1.0	1.0	1.0	-

The intrusion detection performance of each classification model is illustrated in terms of true-positive rate (Fig 1), false-positive rate (Fig 2) and time expense (Fig 3). The *true-positive rate* is the rate of the intrusion detected against all the intrusions happened, while *false-positive rate* is the rate of normal traffic been mistaken by the model as intrusions against all the normal traffics.

In Fig 1, x-axis denotes the true-positive rate, while the y-axis lists all types of intrusions recorded in the training dataset. The results of different machine learning classification models are denoted by different colours. As shown in Fig 1, different intrusions types illustrated different patterns in the traffic, some can be easily detected by all the classification models tested, such as DoS-SlowHTTPTest, DoS-Hulk, DDoS-LOIC-UDP, DDoS-HOIC, FTP-BruteForce. In other word, the traffic generated by these intrusions illustrated more distinct characters or patterns comparing to normal traffic. Some intrusions, however, shown more subtle differences. For example, for the intrusions SQL Injection, and BruteForce-Web, only one classification model **decision tree classifier** can provide a *true-positive rate* higher than 96%. The most difficult to detect intrusion is the **Infiltration**, where almost all the classification models can have a *true-positive rate* less than 30%, except **decision tree classifier**, which gets almost 90%. To summarise, there is only one classification model perform well in all the intrusion types, the **decision tree classifier**, which is denoted by the pale green bars in Fig 1. Actually, as shown in Tables 3 – 13, the accuracy of **decision tree classification** is straight 100%, except for **Infiltration**, which still performs better than the other models.

The *false-positive rate* is illustrated in Fig 2. The x-axis is the *false-positive rate* ranging from 0.00% to 20.00%, and the y-axis lists all types of intrusions in the training set. The *false-positive rate* is indicator of great significance in intrusion detection system, even more important than *true-positive rate*, for in real

Table 5. Machine Learning Results of SSH-BruteForce Attack on Different Classification Models

		Random Forest	Naive Bayes	Decision Tree	Neural Network (MLP)	Quadratic Discriminante	KNeighbors
	True-Positive	100%	100%	100%	100%	100%	–
	False-Positive	0%	0%	0%	0%	0%	–
recall	Benigh	1.0	1.0	1.0	1.0	1.0	–
	SSH-BruteForce	1.0	1.0	1.0	1.0	1.0	–
f1-score	Benigh	1.0	1.0	1.0	1.0	1.0	–
	SSH-BruteForce	1.0	1.0	1.0	1.0	1.0	–

Table 6. Machine Learning Results of BruteForce-Web Attack on Different Classification Models.

		Random Forest	Naive Bayes	Decision Tree	Neural Network (MLP)	Quadratic Discriminante	KNeighbors
	True-Positive	0%	0%	96%	0%	100%	96%
	False-Positive	0%	0%	0%	0%	0%	0%
recall	Benigh	0.21	0.21	1.0	1.0	1.0	1.0
	BruteForce-Web	0.0	0.0	0.46	0.0	0.66	0.90
f1-score	Benigh	0.35	0.35	1.0	1.0	1.0	1.0
	BruteForce-Web	0.0	0.0	0.62	0.0	0.8	0.93

life, if an intrusion detection system generate too many false-positives, the alarms will not be taken seriously or even shut down by human users, which would cause greater danger than low *true-positive rate*. As shown in Fig 2, for most of the intrusions (more specifically, any intrusions other than **Infiltration**), *false-positive rate* generated by all the common machine learning classification models are as low as 0%, except for **Infiltration**, which experiences between 10.00% ~ 17.00%.

The overhead in terms of time expense is illustrated in Fig 3 to demonstrate the efficiency of each machine learning classification models. The model that performs best in accuracy, **decision tree classification**, also cause less time than the peer classification models, consuming less than 20 seconds on all attack types.

As analysed above, in the cross validation experiment, the **decision tree classification** fitted best among the six common classifier models, consuming less time. Thus, we choose it as the training model for the intrusion detection of testing data, as discussed in the following section.

4.3 Test Results

Our goal is to detect intrusions with the accuracy in terms of *false-positive rate* and *true-positive rate* as high as possible, especially when the intrusions has not been seen before (in other words **Zero-Day** attacks), we normalised the intrusion types in train CIC-AWS-2018 Dataset and test datasets into one unique type “Evil”. Thus there are only two types of traffic in the datasets, “Benign” and “Evil”. The goal is to detect “Evil” traffic, especially **Zero-Day** attacks, with the highest possible accuracy, no matter what type of intrusion it is. The ability of identifying the exact types of detected intrusions is also highly desirable, but

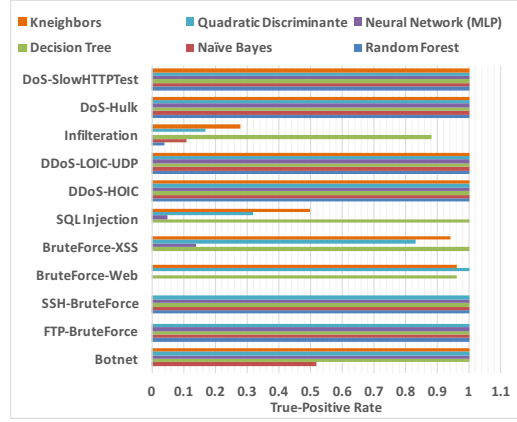


Fig. 1. The True-Positive rate for different classification models on different attack types.

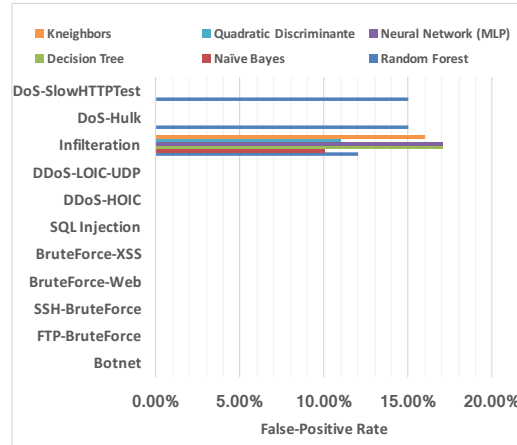


Fig. 2. The False-Positive rate for different classification models on different attack types.

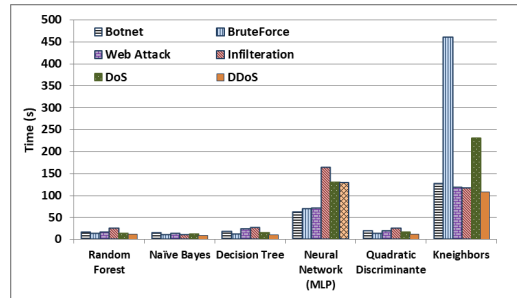


Fig. 3. The machine learning time consumption for different attack types.

Table 7. Machine Learning Results of BruteForce-XSS Attack on Different Classification Models.

		Random Forest	Naive Bayes	Decision Tree	Neural Network (MLP)	Quadratic Discriminante	KNeighbors
	True-Positive	0%	0%	100%	14%	83%	94%
	False-Positive	0%	0%	0%	0%	0%	0%
recall	Benigh	0.21	0.21	1.0	1.0	1.0	1.0
	BruteForce-XSS	0.0	1.0	0.50	0.31	0.94	0.94
f1-score	Benigh	0.35	0.35	1.0	1.0	1.0	1.0
	BruteForce-XSS	0.0	0.0	0.67	0.19	0.88	0.94

Table 8. Machine Learning Results of SQL Injection Attack on Different Classification Models.

		Random Forest	Naive Bayes	Decision Tree	Neural Network (MLP)	Quadratic Discriminante	KNeighbors
	True-Positive	0%	0%	100%	5%	32%	50%
	False-Positive	0%	0%	0%	0%	0%	0%
recall	Benigh	0.21	0.21	1.0	1.0	1.0	1.0
	SQL Injection	0.0	0.43	0.71	0.29	1.0	0.43
f1-score	Benigh	0.35	0.35	1.0	1.0	1.0	1.0
	SQL Injection	0.0	0.0	0.83	0.09	0.48	0.46

we focus on detecting intrusions only in this paper and will leave it as future work, thus will not discuss it here.

We evaluate the test datasets on the chosen **decision tree classification** model in three steps: 1) test on “Benign” test data only; 2) test on “Evil” test data only; 3) test on the combined and shuffled benign and intrusion test dataset, with different max depth of the decision tree.

The test results of steps 1) and 2) are shown in Table 14. As shown in the bold numbers, the model is able to detect the tested “Evil” and “Benign” data with 100% accuracy. For there is no “Benign” data in the “Evil data only” test, and no “Evil” data in the “Benign data only” test, the obvious “0.00%” precision of corresponding data are got.

At step 3), we merge and shuffle the intrusion test data with the **Zero-Day** intrusions (comparing to training data) listed in Table 2 into the benign dataset. The *depth* of a decision tree is the length of the longest path from root to a leaf, it determined the size of the tree and affect the performance of the tree models. To find the best size of the decision tree, we fit the test datasets into a couple of decision tree classifiers with the maximum depth of 2, 3, 4, 5, and 6. The detection efficiency is evaluated in terms of *false-positive rate* and *true-positive rate* in Fig 4, and the time expenses of different tree models are also given in Fig 5. In Fig 4, the x-axis denotes the *maximum depth* of the different **decision tree classifiers**, and the y-axis denotes the percentage rates. The *true-positive rate* is denoted with orange solid bars in every model, while the *false-positive rate* is denoted with striped black bars. As shown in Fig 4, the *true-positive rate* achieves the best at 96% when the *maximum depth* is 5, and slightly deteriorates to 92% and 90% when the *maximum depth* reduces to 3 and 4. The *false-positive rate* gets its lowest at 5% when the *maximum depth* are 3 and 4, and deteriorates to about 10% when the *maximum depth* gets either lower or higher to 2 or 5. In summary, the intrusion detector performs best with the

Table 9. Machine Learning Results of DDoS-HOIC Attack on Different Classification Models.

		Random Forest	Naive Bayes	Decision Tree	Neural Network (MLP)	Quadratic Discriminante	KNeighbors
True-Positive		100%	100%	100%	100%	100%	100%
False-Positive		0%	0%	0%	0%	0%	0%
recall	Benigh	1.0	1.0	1.0	1.0	1.0	1.0
	DDoS-HOIC	1.0	1.0	1.0	1.0	1.0	1.0
f1-score	Benigh	1.0	1.0	1.0	1.0	1.0	1.0
	DDoS-HOIC	1.0	1.0	1.0	1.0	1.0	1.0

Table 10. Machine Learning Results of DDOS-LOIC-UDP Attack on Different Classification Models.

		Random Forest	Naive Bayes	Decision Tree	Neural Network (MLP)	Quadratic Discriminante	KNeighbors
True-Positive		100%	100%	100%	100%	100%	100%
False-Positive		0%	0%	0%	0%	0%	0%
recall	Benigh	1.0	1.0	1.0	1.0	1.0	1.0
	DDoS-LOIC-UDP	1.0	0.66	1.0	0.93	0.66	0.69
f1-score	Benigh	1.0	1.0	1.0	1.0	1.0	1.0
	DDoS-LOIC-UDP	1.0	0.80	1.0	0.97	0.80	0.81

tree depth at 3, 4, and 5, either increase or decrease the maximum depth will deteriorate the performance. Further experiments will be carried out to find the better model to fit the flow-based statistical data for intrusion detection, but will be discussed somewhere else.

The time expenses of the decision tree classifiers with different maximum depth are shown in Fig 5, in terms of real (striped blue bars), sys (doted orange bars), and user (solid green bars) time in seconds. As obviously shown in the trending line of real time (which is the totally consumed time), the time expense grows exponentially with the maximum depth. This is predictable for the size of the decision tree classifier is $2^{d+1} - 1$, where d is the depth of the tree, and the calculation time expense is positively related to the size of the decision tree.

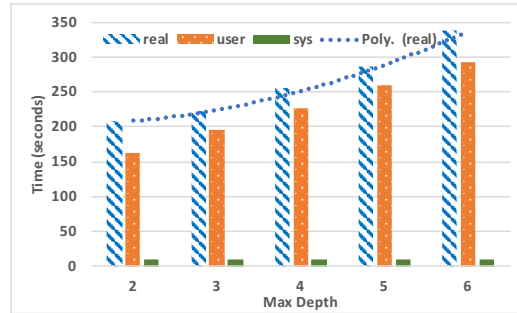
**Fig. 4.** The true-positive and false-positive rate of the test process with decision tree classifier under different max depth.

Table 11. Machine Learning Results of Infiltration Attack on Different Classification Models.

		Random Forest	Naive Bayes	Decision Tree	Neural Network (MLP)	Quadratic Discriminante	KNeighbors
True-Positive		4%	11%	88%	0%	17%	28%
False-Positive		12%	10%	17%	17%	11%	16%
recall	Benigh	0.96	1.0	1.0	1.0	1.0	0.92
	Infiltration	0.01	0.01	0.01	0.0	0.01	0.15
f1-score	Benigh	0.92	1.0	0.91	0.91	1.0	0.88
	Infiltration	0.02	0.02	0.02	0.0	0.02	0.19

Table 12. Machine Learning Results of DOS-Hulk Attack on Different Classification Models.

		Random Forest	Naive Bayes	Decision Tree	Neural Network (MLP)	Quadratic Discriminante	KNeighbors
True-Positive		100%	100%	100%	100%	100%	100%
False-Positive		15%	0%	0%	0%	0%	0%
recall	Benigh	1.0	1.0	1.0	1.0	1.0	1.0
	DOS-Hulk	1.0	1.0	1.0	0.83	0.66	1.0
f1-score	Benigh	1.0	1.0	1.0	0.92	1.0	1.0
	DOS-Hulk	1.0	1.0	1.0	0.91	1.0	1.0

**Fig. 5.** The time expenses of the test process with decision tree classifier under different max depth, in terms of real, sys, and user time.**Table 13.** Machine Learning Results of DoS-SlowHTTPTest Attack on Different Classification Models.

		Random Forest	Naive Bayes	Decision Tree	Neural Network (MLP)	Quadratic Discriminante	KNeighbors
True-Positive		100%	100%	100%	100%	100%	100%
False-Positive		15%	0%	0%	0%	0%	0%
recall	Benigh	1.0	1.0	1.0	1.0	1.0	1.0
	DoS-SlowHTTPTest	1.0	1.0	1.0	1.0	0.66	1.0
f1-score	Benigh	1.0	1.0	1.0	0.92	1.0	1.0
	DoS-SlowHTTPTest	1.0	1.0	1.0	1.0	1.0	1.0

Table 14. Classification Results of the Benign and Intrusion Test Datasets on Decision Tree Classification Model.

			Results
Evil data only	precision	Benign	0.00%
		Evil	100%
	recall	Benign	0.00
		Evil	0.40
	f1-score	Benign	0.00
		Evil	0.57
Benign data only	precision	Benign	100%
		Evil	0.00%
	recall	Benign	1.00
		Evil	0.00
	f1-score	Benign	1.00
		Evil	0.00

5 Conclusions

In this paper, we take an intensive analysis on intrusion detection using the flow-based statistical data generated from network traffic packets with CICFlowMeter, using machine learning classification models. Six common machine learning classifications models are tested on the datasets generated from real-life attacks and production networks. CIC-AWS-2018 Dataset which is collected by Amazon cluster networks, containing benign traffic and fourteen different types of intrusions are used as training dataset, eight different types of intrusions traffic data collected online and benign traffic data collected from our research production network are used as testing dataset. Cross validations over the training dataset are carried out on the six common machine learning classification models, one model, decision tree classification, with the best performance on general adaptability, precision, and time consumption, is chosen to carry out the testing experiment. The testing results demonstrate acceptable performance for intrusion detection, with 100% accuracy in detecting on **Zero-Day** intrusion or benign data alone, and 96% accuracy in scrambled **Zero-Day** intrusion and benign data, and false-positive rate of 5% at lowest.

Much is left to do in the future, for example, finding better models to fit the statistical flow data, and testing on more novel types of intrusions in real time.

Bibliography

- [1] R. Hamilton, J. Iyengar, I. Swett, A. Wilk, et al., Quic: A udp-based secure and reliable transport for http/2, IETF, draft-tsvwg-quic-protocol-02.
- [2] R. Sommer, V. Paxson, Outside the closed world: On using machine learning for network intrusion detection, in: 2010 IEEE symposium on security and privacy, IEEE, 2010, pp. 305–316.
- [3] L. Dhanabal, S. Shantharajah, A study on nsl-kdd dataset for intrusion detection system based on classification algorithms, *International Journal of Advanced Research in Computer and Communication Engineering* 4 (6) (2015) 446–452.
- [4] A. L. Buczak, E. Guven, A survey of data mining and machine learning methods for cyber security intrusion detection, *IEEE Communications Surveys & Tutorials* 18 (2) (2016) 1153–1176.
- [5] A. S. A. Aziz, E. Sanaa, A. E. Hassanien, Comparison of classification techniques applied for network intrusion detection and classification, *Journal of Applied Logic* 24 (2017) 109–118.
- [6] J. Kim, J. Kim, H. L. T. Thu, H. Kim, Long short term memory recurrent neural network classifier for intrusion detection, in: 2016 International Conference on Platform Technology and Service (PlatCon), IEEE, 2016, pp. 1–5.
- [7] M. Ahmed, A. N. Mahmood, J. Hu, A survey of network anomaly detection techniques, *Journal of Network and Computer Applications* 60 (2016) 19–31.
- [8] S. Aljawarneh, M. Aldwairi, M. B. Yassein, Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model, *Journal of Computational Science* 25 (2018) 152–160.
- [9] J. Undercofer, et al., Intrusion detection: Modeling system state to detect and classify aberrant behavior.
- [10] A. Gharib, I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani, An evaluation framework for intrusion detection dataset, in: 2016 International Conference on Information Science and Security (ICISS), IEEE, 2016, pp. 1–6.
- [11] P. Aggarwal, S. K. Sharma, Analysis of kdd dataset attributes-class wise for intrusion detection, *Procedia Computer Science* 57 (2015) 842–851.
- [12] I. Sharafaldin, A. Gharib, A. H. Lashkari, A. A. Ghorbani, Towards a reliable intrusion detection benchmark dataset, *Software Networking* 2018 (1) (2018) 177–200.
- [13] I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani, Toward generating a new intrusion detection dataset and intrusion traffic characterization., in: ICISSP, 2018, pp. 108–116.
- [14] B. J. Radford, B. D. Richardson, Sequence aggregation rules for anomaly detection in computer network traffic, *arXiv preprint arXiv:1805.03735*.

- [15] I. Ullah, Q. H. Mahmoud, A two-level hybrid model for anomalous activity detection in iot networks, in: 2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC), IEEE, 2019, pp. 1–6.
- [16] Z. Zhang, J. Li, C. Manikopoulos, J. Jorgenson, J. Ucles, Hide: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification, in: Proc. IEEE Workshop on Information Assurance and Security, 2001, pp. 85–90.