IET Networks

Research Article

IET Journals
The Institution of Engineering and Technology

Towards an efficient anomaly-based intrusion detection for software-defined networks

ISSN 2047-4954
Received on 29th May 2018
Revised 19th July 2018
Accepted on 16th August 2018
E-First on 1st October 2018
doi: 10.1049/iet-net.2018.5080
www.ietdl.org

Majd Latah[1] ✉, *Levent Toker*[2]
[1]*Department of Computer Science, Ozyegin University, 34794, Cekmekoy, Istanbul, Turkey*
[2]*Department of Computer Engineering, Ege University, 35100, Bornova, Izmir, Turkey*
✉ *E-mail: majd.latah@ozu.edu.tr*

Abstract: Software-defined networking (SDN) is a new paradigm that allows developing more flexible network applications. A SDN controller, which represents a centralised controlling point, is responsible for running various network applications as well as maintaining different network services and functionalities. Choosing an efficient intrusion detection system helps in reducing the overhead of the running controller and creates a more secure network. In this study, we investigate the performance of the well-known anomaly-based intrusion detection approaches in terms of accuracy, false alarm rate, precision, recall, f1-measure, area under receiver operator characteristic curve, execution time and McNemar's test. Precisely, the authors focus on supervised machine-learning approaches where we use the following classifiers: decision trees, extreme learning machine, Naive Bayes, linear discriminant analysis, neural networks, support vector machines, random forest, K-nearest-neighbour, AdaBoost, RUSBoost, LogitBoost and BaggingTrees where we employ the well-known NSL-KDD benchmark dataset to compare the performance of each one of these classifiers.

1 Introduction

Network security is one of the most important aspects in modern communications. Recently, programmable networks have gained popularity due to their abstracted view of the network, which, in turn, provides a better understanding of the complex network operations and increases the effectiveness of the actions that should be taken in the case of any potential threat. Software-defined networking (SDN) represents an emerging centralised network architecture, in which the forwarding elements are being managed by a central unit, called an SDN controller, which has the ability to obtain traffic statistics from each forwarding element in order to take the appropriate action required for preventing any malicious behaviour or abusing of the network. At the same time, the SDN controller uses a programmable network protocol, which is OpenFlow (OF) protocol, in order to communicate and forward its decisions to OF-enabled switches [1].

In spite of the significant impact of using a centralised controller, the controller itself creates, a single point of failure, which makes the network more vulnerable compared with the conventional network architecture [2]. On the other hand, the existence of a communication between the OF-enabled switches and the controller opens the door for various attacks such denial of service (DoS) [3], host location hijacking and man in the middle attacks [4]. Therefore, in order to develop an efficient intrusion detection system (IDS) for SDNs, the system should be able to make intelligent and real-time decisions. Commonly, an IDS designed for SDNs works on the top of the controller, which forms an additional burden on the controller itself. Thus, designing a lightweight IDS is considered an advantage, since it helps in effectively detecting any potential attacks as well as performing other fundamental network operations such as routing and load balancing in a more flexible manner. Scalability is also an important factor which should be taken into consideration during the designing stage of the system [4]. There are two main groups of intrusion detection systems: signature-based IDS and anomaly-based IDS. Signature-based IDS searches for defined patterns within the analysed network traffic. On the other hand, the anomaly-based IDS is able to estimate and predict the behaviour of system. The signature-based IDS shows a good performance only for specified well-known attacks. On the contrary, the anomaly-based IDS exhibits ability to detect unseen intrusion events, which is an important advantage in order to detect zero day attacks [5].

Anomaly-based IDS can be grouped into three main categories [5]: statistical-based approaches, knowledge-based approaches, and machine learning-based approaches. In this study, we are focusing on machine learning-based approaches. Machine learning techniques can be categorised into four main categories: (i) supervised techniques, (ii) semi-supervised techniques, (iii) unsupervised techniques and (iv) reinforcement techniques. In this study, we investigate various supervised learning techniques with respect to their accuracy, false alarm rate (FAR), precision, recall, F1-measure, area under receiver operator characteristic (ROC) curve, McNemar's test and time taken to train and test each classifier.

2 Related work

Previous research efforts for providing a detailed analysis of supervised machine learning techniques used for intrusion detection are summarised in Table 1. These studies focused on training and testing different machine learning approaches using standard intrusion detection datasets. However, obtaining all these features from an SDN controller could be computationally expensive. Therefore, we have two possible choices: either using a subset of these standard datasets [6] or extracting new features based on network traces of standard datasets or statistics provided by the controller [7]. In this study, we use a subset of features extracted from the NSL-KDD dataset based on the principal components analysis (PCA) approach and we consider the following supervised machine learning approaches: decision trees (DTs), extreme learning machine (ELM), Naive Bayes (NB), linear discriminant analysis (LDA), neural networks (NNs), support vector machines (SVM), random forest (RT), K-nearest-neighbour (KNN), AdaBoost, RUSBoost, LogitBoost, and BaggingTrees. As mentioned before, for performance measurement, we use accuracy, FAR, precision, recall, F1-measure and area under ROC curve, as well as time taken to train and test each one of these classifiers. Furthermore, we use McNemar's test to statistically demonstrated that a significant increase has been achieved by using the algorithm over the other one.

Table 1 Overview of previous supervised machine learning studies for intrusion detection

Ref.	Year	Algorithms	Dataset
[8]	2005	C 4.5	KDD CUP'99
		KNN	
		multi-layer perceptron (MLP)	
		regularised discriminant analysis	
		Fisher linear discriminant	
		SVMs	
[9]	2007	DTs	KDD CUP'99
		RF	
		NB	
		Gaussian classifier	
[10]	2009	J48	NSL-KDD
		NB	
		NB-Tree	
		RF	
		RandomTree	
		MLP	
		SVM	
[11]	2010	discriminative multinomial NB classifiers	NSL-KDD
[12]	2013	PCA-based feature selection	NSL-KDD
		genetic algorithm-based detector generation, J48, NB, MLP, BF-Tree, NB-Tree, RF-Tree	
[13]	2013	feature selection by correlation-based feature selection and consistency-based filter	NSL-KDD
		ADTree, C4.5, J48graft, LAD-Tree, NB-Tree, RandomTree, RF, REPTree	
[14]	2013	J48, BayesNet, Logistic, SGD, IBK, JRip, PART, RT, RandomTree and REPTree	NSL-KDD
[15]	2015	NNs	NSL-KDD
[16]	2016	logistic regression	NSL-KDD
		Gaussian NB	
		SVM and RF	

Table 2 List of features of KDD Cup '99 dataset

F. #	Feature name.	F. #	Feature name.	F. #	Feature name.
F1	duration	F15	Su attempted	F29	Same srv rate
F2	protocol type	F16	Num root	F30	Diff srv rate
F3	service	F17	Num file creations	F31	Srv diff host rate
F4	flag	F18	Num shells	F32	Dst host count
F5	source bytes	F19	Num access files	F33	Dst host srv count
F6	destination bytes	F20	Num outbound cmds	F34	Dst host same srv rate
F7	land	F21	Is host login	F35	Dst host diff srv rate
F8	wrong fragment	F22	Is guest login	F36	Dst host same src port rate
F9	urgent	F23	Count	F37	Dst host srv diff host rate
F10	hot	F24	Srv count	F38	Dst host serror rate
F11	number failed logins	F25	Serror rate	F39	Dst host srv serror rate
F12	logged in	F26	Srv serror rate	F40	Dst host rerror rate
F13	num compromised	F27	Rerror rate	F41	Dst host srv rerror rate
F14	root shell	F28	Srv rerror rate	F42	Class label

Table 3 List of feature categories presented in the NSL-KDD dataset

Category	Features
basic features	F1, F2, F3, F4, F5, F6, F7, F8, F9, F10
content features	F11, F12, F13, F14, F15, F16, F17, F18, F19, F20, F21, F22
time-based features	F23, F24, F25, F26, F27, F28, F29, F30, F31
host-based features	F32, F33, F34, F35, F36, F37, F38, F39, F40, F41

3 Dataset

As mentioned earlier, in this study, we use the NSL-KDD dataset. The NSL-KDD is an improved version of the KDD Cup99 dataset, which suffers from huge number of redundant records [10]. Both KDD Cup99 and NSL-KDD datasets include the features shown in Table 2. It is worth mentioning that these features fall into four different categories as described in Table 3.

As shown in Table 4, the NSL-KDD includes a total of 39 attacks where each one of them is classified into one of the following four categories (DoS, R2L, U2R, and probe). Moreover, a set of these attacks is introduced only in the testing set. These new attacks are indicated in bold font.

In addition, Table 5 shows the distribution of the normal and attack records in NSL-KDD training and testing sets.

4 Feature selection

To increase the efficiency of SDN-based intrusion detection systems we need to select the best features that can be used in the SDN context. It is worth noting that the content features need to be omitted because these features are complex to extract by a network-based IDS [17]. Therefore, content features (i.e. F11 to F22) were excluded from the NSL-KDD dataset. For the remaining features, we apply principal component analysis (PCA) on the training set. PCA allow us to transform a large dataset into a new, smaller and uncorrelated one [18]. The standard approach of PCA can be summarised in the following six steps [19]:

- Find the covariance matrix of the normalised d-dimensional dataset.
- Find the eigenvectors and eigenvalues of the covariance matrix.
- Sort the eigenvalues in descending order.
- Select the k eigenvectors that correspond to the k largest eigenvalues.
- Construct the projection matrix from the k-selected eigenvectors.
- Transform the original dataset to obtain a new k-dimensional feature space.

In this study, we employ the PCA in the following steps:

- First, we extract the features with the largest coefficients from the principal components.
- Second, we select the k eigenvectors that correspond to the k largest eigenvalues.
- Third, we transform the original dataset with corresponding features using the projection matrix from the k-selected eigenvectors.
- Finally, we validate the performance of the selected features and corresponding k component by applying the DT approach on the training test.

5 Evaluation metrics

The performance of each classifier is evaluated in terms of accuracy, FAR, precision, recall, F1-measure, area under ROC curve [area under curve (AUC)], execution time and McNemar's test. A good IDS should achieve high level of accuracy, precision, recall and F1-measure with low FAR. The accuracy is calculated by

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

True positives (TPs) are the number of attack records correctly classified; true negatives (TNs) are the number of normal traffic records correctly classified; false positives (FP) are the number of normal traffic records falsely classified and false negatives (FN) is number of attack record instances falsely classified. FAR is calculated by

$$\text{False alarm rate} = \frac{\text{FP}}{\text{TN} + \text{FP}}. \tag{2}$$

We also calculate the precision, recall and F1-measure for each classifier where precision is calculated by

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{3}$$

Recall is calculated by

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{4}$$

F1-measure is calculated by

$$\text{F1} - \text{measure} = 2 \times \frac{(\text{precision} \times \text{recall})}{(\text{precision} + \text{recall})}. \tag{5}$$

In addition, we evaluate the performance of previously selected classifiers based on execution time as well as the analysis of the ROC curve where the AUC can be used to compare each classifier with another one. The higher AUC, the better IDS. One other important metric that can be used for comparing two algorithms is McNemar's test, which is a non-parametric pair-wise test, showing that a statistically significant increase has been achieved by an algorithm over the other one. When z-value of McNemar's test >1.96 (p<0.05), the conclusion is that there is a significant difference between the two algorithms. Z-score is used to show the confidence levels [20]

$$z = \frac{(|N_{12} - N_{21}|) - 1}{\sqrt{(N_{12} + N_{21})}}, \tag{6}$$

N_{12}: represents the number of times when the first algorithm success in classification and other one fails.
N_{21}: represents the number of times when the second algorithm success in classification and the first one fails.

6 Experimental results

The experiment is conducted on Intel i5 machine with 12 GB RAM. As shown in Fig. 1, we obtain the best results when selecting nine of the top features that contribute to the all PCA's components as input which need to be transformed to less dimensional space of the corresponding components.

These nine selected features are F27, F30, F5, F23, F8, F1, F2, F39, and F3. A brief description of these features is provided in Table 6.

Fig. 2 shows the level of accuracy achieved when using different number of principal components. The best results achieved with the first ten components.

Table 7 shows the results obtained for both training and testing stages. In terms of the accuracy level, the most accurate classifiers for the training stage are DT, random forest (RF), BaggingTrees, RUSBoost and AdaBoost with a slight difference between them. For the testing stage, however, we notice that the DT approach achieved the highest level of accuracy followed by AdaBoost, RUSBoost and BaggingTrees. One can observe that ensemble methods achieved a lower false positive rate compared with DT.

Table 4 List of attacks presented in NSL-KDD dataset

Attack category	Attack name
DoS	**Apache2**, Smurf, Neptune, Back, Teardrop, Pod, Land, **Mailbomb**, **Processtable**, **UDPstorm**
remote to local (R2L)	WarezClient, Guess_Password, WarezMaster, Imap, Ftp_Write, **Named**, MultiHop, Phf, Spy, **Sendmail**, **SnmpGetAttack**, **SnmpGuess**, **Worm**, **Xsnoop**, **Xlock**
user to root (U2R)	Buffer_Overflow, **Httptuneel**, Rootkit, LoadModule, Perl, **Xterm**, **Ps**, **SQLattack**
probe	Satan, **Saint**, Ipsweep, Portsweep, Nmap, **Mscan**

Table 5 Distributions of attacks and normal records in NSL-KDD dataset

KDD dataset	Total records	Normal	DoS	R2L	U2R	Probe
KDD train	125,973	67,343	45,927	995	52	11,656
		53.46%	36.46%	0.79%	0.04%	9.25%
KDD test	22,544	9711	7458	2754	200	2421
		43.07%	33.08%	12.22%	0.89%	10.74%

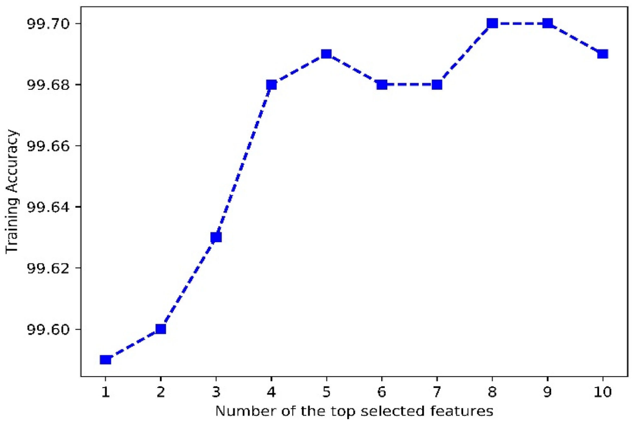


Fig. 1 *Level of accuracy obtained by using top selected features*

Table 6 List of features selected from the NSL-KDD dataset

Feature	Description
F27	percentage of connections that have REJ errors
F30	percentage of connections to different services
F5	number of data bytes from source to destination
F23	number of connections to the same host as the current connection in the past 2 s
F8	number of wrong fragments
F1	duration of the connection in seconds
F2	connection protocol (tcp, udp, icmp)
F39	percentage of connections to the current host and serror rate specified service that have an S0 error
F3	destination port mapped to service

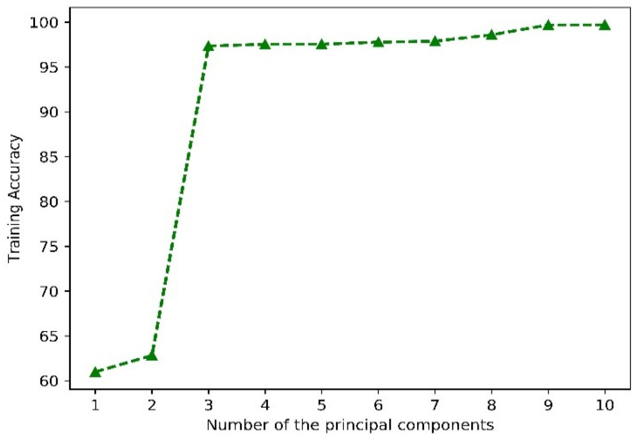


Fig. 2 *Level of accuracy obtained with different number of PCA's principle components*

In terms of FAR, it is worth mentioning that the LogitBoost approach achieved the best results. Therefore, one can conclude that ensemble methods such AdaBoost and LogitBoost can achieve a good accuracy with low false positive rate.

In Tables 7 and 8, one can observe that using PCA feature selection enhanced the accuracy level for most of the classifiers in comparison with using the basic features provided by the SDN controller (F1, F2, F5, F6, F23 and F24). In terms of area under ROC curve, as shown in Fig. 3*a*, we notice that DT and RF approaches achieved the best AUC for the training task followed by BaggingTrees, RUSBoost, AdaBoost and LogitBoost with slight difference between each other. Both NN and SVM had nearly the same AUC for the training task. NB, however, achieved the least training AUC.

For the testing task, as shown in Fig. 3*b*, one can observe that the best AUC obtained by DT followed by AdaBoost, RUSBoost, LogitBoost and BaggingTrees. KNN achieved a better AUC than SVM and RF. In the same context, we notice that the SVM also achieved a higher AUC than ELM approach. In terms of precision and F1-measure the best results were achieved by DT whereas LogitBoost achieved the best results in terms of recall (Table 9).

Table 7 Detection accuracy and FAR obtained after training and testing different supervised machine learning algorithms with ten principle components

Method	Accuracy, %		FAR, %	
	Training	Testing	Training	Testing
NB	64.16	49.12	4.54	5.74
LDA	72.37	70.32	9.98	3.76
linear SVM	91.04	81.40	9.21	5.92
NN	92.15	74.23	4.54	6.38
ELM	92.66	75.86	5.54	3.57
KNN	98.14	82.31	1.92	3.53
LogitBoost	98.95	84.85	0.94	**2.83**
AdaBoost	99.03	87.16	1.03	3.68
RUSBoost	99.19	85.57	0.96	3.59
BaggingTrees	99.33	84.03	0.81	3.51
RF	**99.70**	80.13	**0.29**	3.49
DT	**99.70**	**88.74**	0.31	3.99

The bold values indicate the best values for each metric.

Table 8 Detection accuracy and FAR obtained after training and testing different supervised machine learning algorithms based on basic features provided by the SDN controller (i.e. features number F1, F2, F5, F6, F23 and F24)

Method	Accuracy, %		False positive rate, %	
	Training	Testing	Training	Testing
NB	59.27	49.88	3.7227	5.14
LDA	87.57	69.36	3.26	2.24
SVM	90.86	71.00	6.55	10.27
NN	84.10	66.22	2.41	1.61
ELM	93.16	74.17	2.25	2.31
KNN	98.23	77.09	3.128	4.07
RF	98.09	75.96	**0**	**0**
DT	98.37	74.43	0.306	6.43
LogitBoost	99.38	79.44	0.43	2.75
BaggingTrees	99.54	79.16	0.47	3.26
AdaBoost	99.56	78.94	0.384	2.76
RUSBoost	**99.68**	**80.31**	0.29	3.48

The bold values indicate the best values for each metric.

For McNemar's test, the null hypothesis suggests that the different classifiers perform similarly whereas the alternative hypothesis claims that at least one of the classifiers performs differently. As shown in Table 10, by looking at the z-score values of McNemar's test, one can conclude that DT achieved significantly better results than the other classifiers where the alternative hypothesis was accepted with a confidence level >99.5%. KNN also performed better than RF. AdaBoost also performed better than the other algorithms except DT, with a confidence level more than 99.5%. Bagging and boosting produced better results over other conventional machine learning methods such as KNN, ELM, NN, RF, SVM and LDA.

In terms of execution time, as shown in Fig. 4*a*, we notice that the NB approach achieved the best results for the training task. We excluded the KNN from Fig. 4*a* because the KNN has no training time, where this algorithm employs a distance function in order to predict the corresponding labels [21]. From Fig. 4*b*, on the other hand, one can observe that the ELM approach achieved the best testing time. Moreover, ELM has achieved an acceptable FAR as shown from Table 7. Therefore, the ELM and its improved hierarchical approach [22] can possibly be an efficient choice for SDNs.

On the other hand, in spite of the good level of accuracy for the testing stage achieved by the KNN approach, it showed the worst

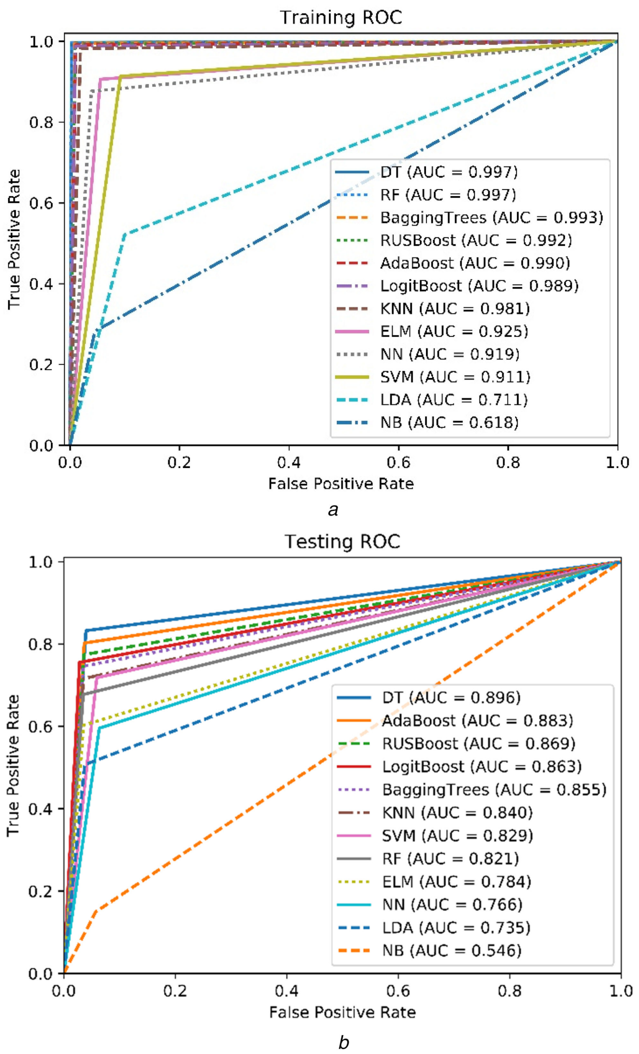


Fig. 3 *ROC curve comparison for*
(a) Training, ***(b)*** Testing different supervised machine learning based IDS

testing time, which may indicate that the KNN algorithm is not the best choice for SDNs where each controller may need to handle thousands of flows per second. A possible solution to this problem can be achieved by reducing the number of training instances by applying an appropriate sampling method. Finally, one can observe that DT has achieved the highest level of accuracy and a good testing time in comparison with the other classifiers.

7 Conclusion

We provide a comparative study of choosing an efficient anomaly-based intrusion detection method for SDNs. We focused on supervised machine learning approaches by using the following classifiers: NN, LDA, DT, RF, Linear SVM, KNN, NB, ELM, AdaBoost, RUSBoost, LogitBoost and BaggingTrees. In addition, we used the PCA method for feature selection and dimensionality reduction. Using the NSL-KDD dataset and based on our experimental studies, we conclude that the DT approach shows the best performance in terms of accuracy, precision, F1-measure, AUC and McNemar's test. Also, bagging and boosting approaches outperformed other conventional machine learning methods such as KNN, ELM, NN, RF, SVM and LDA with a confidence level >99.5%, whereas in terms of FAR and recall the best results achieved by LogitBoost. In terms of the execution time, the ELM approach achieved the best testing time.

It is worth noting that using the PCA approach was very successful in enhancing the accuracy level from 80.31 to 88.74% in comparison with the basic features provided by the SDN controller. Our future work will be focused on comparing the results obtained from this study with other machine learning approaches and exploring other flow-based features that could be used in order to achieve a higher level of accuracy with a lower FAR.

Table 9 Precision, Recall, F1-measure obtained after training and testing different supervised machine learning algorithms with ten principle components

Method	Precision, %	Recall, %	F1-measure, %
NB	14.95	77.49	25.06
LDA	50.7	94.69	66.07
linear SVM	71.81	94.13	81.47
NN	59.56	92.5	72.46
ELM	60.29	85.71	73.98
KNN	71.59	96.41	82.17
LogitBoost	75.53	**97.24**	85.03
AdaBoost	80.23	96.65	87.67
RUSBoost	77.41	96.6	85.95
BaggingTrees	74.61	96.56	84.17
RF	67.73	96.25	80.13
DT	**83.24**	96.50	**89.38**

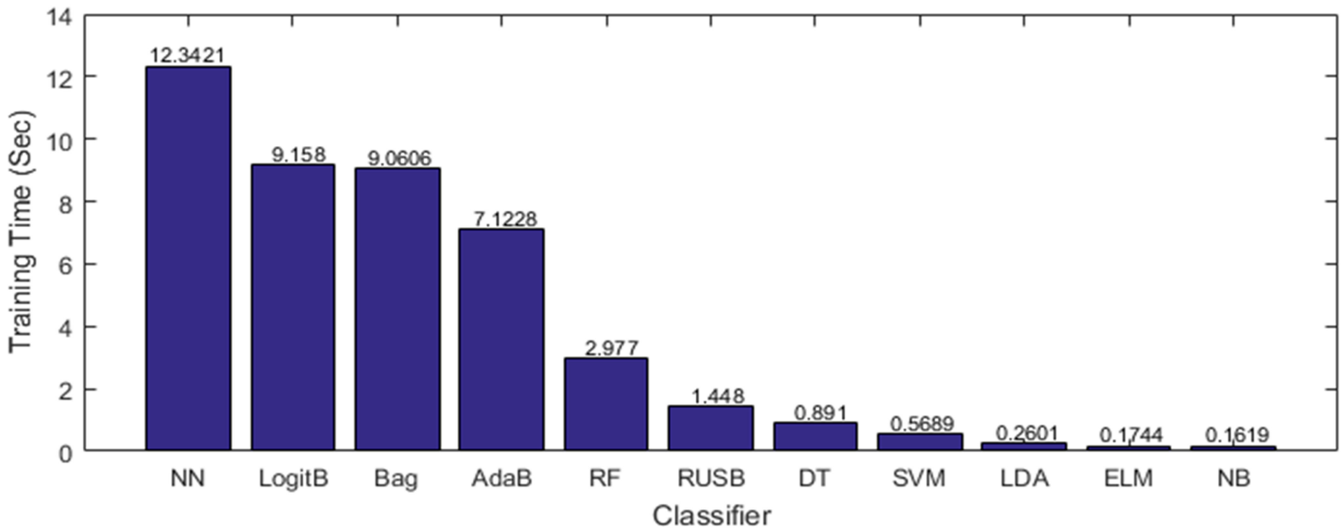
The bold values indicate the best values for each metric.

8 Acknowledgment

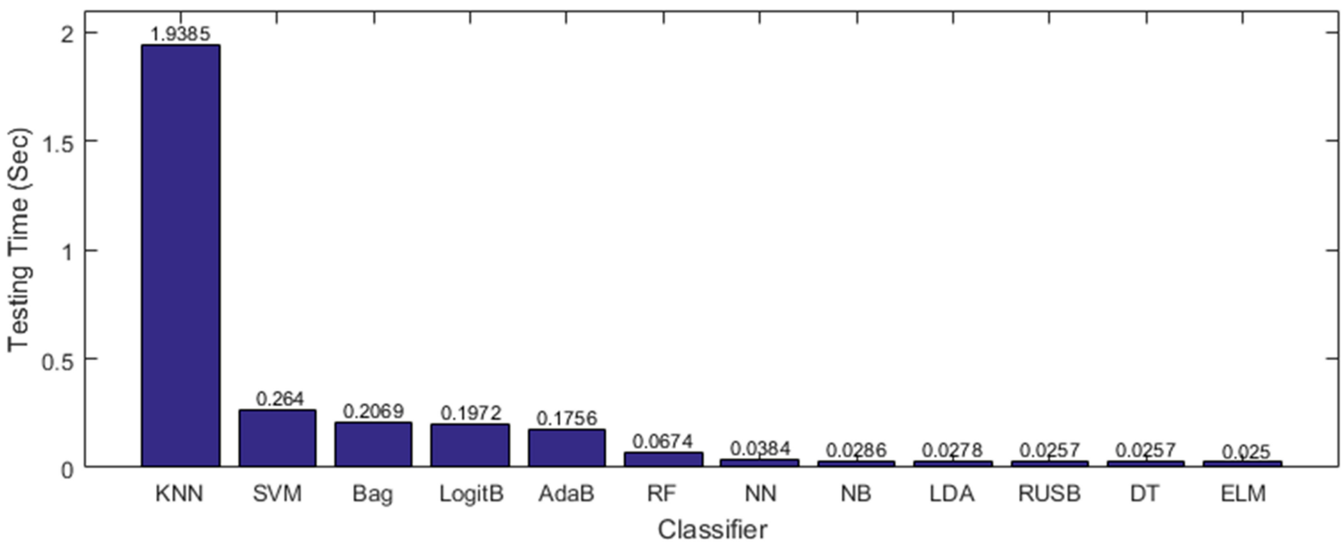
We would like to thank the reviewers for their insightful comments to improve the quality of this paper.

Table 10 *Z*-score values of McNemar's test for the supervised machine-learning algorithms used in the this study (the arrowheads ← ↑ denote which classifier performed better). The shaded ones that are larger than 1.96 indicate statistically significant differences at the confidence level of 95% ($p<0.05$)

	NB	DT	AdaBoost	RUSBoost	LogitBoost	Bagging	RF	KNN	ELM	NN	SVM	LDA
NB	—	—	—	—	—	—	—	—	—	—	—	—
DT	86.2←	—	—	—	—	—	—	—	—	—	—	—
AdaBoost	88.2←	7.1↑	—	—	—	—	—	—	—	—	—	—
RUSBoost	82.4←	19.2↑	10.3↑	—	—	—	—	—	—	—	—	—
LogitBoost	86.2←	16.5↑	18.2↑	4.3↑	—	—	—	—	—	—	—	—
Bagging	84.4←	20.7↑	22.1↑	9.2↑	6.4↑	—	—	—	—	—	—	—
RF	75.0←	40.0↑	35.7↑	30.0↑	25.2↑	24.8↑	—	—	—	—	—	—
KNN	78.1←	28.0↑	23.8↑	17.5↑	13.9↑	10.1↑	12.3←	—	—	—	—	—
ELM	71.1←	42.6↑	43.2↑	38.6↑	35.6↑	33.5↑	13.1↑	29.3↑	—	—	—	—
NN	64.4←	49.4.↑	47.6↑	43.9↑	39.0↑	37.9↑	24.1↑	30.2↑	13.7↑	—	—	—
SVM	75.5←	25.1↑	22.9↑	15.7↑	14.0↑	10.5↑	4.6↑	3.4↑	17.4↑	22.3↑	—	—
LDA	58.5←	56.7↑	54.9↑	49.7↑	50.7↑	47.1↑	33.8↑	41.1↑	26.6↑	12.4↑	36.0↑	—



a



b

Fig. 4 *Execution time for*
(a) Training, ***(b)*** Testing different supervised machine learning method

9 References

[1] Ha, T., Kim, S., An, N, *et al.*: 'Suspicious traffic sampling for intrusion detection in software-defined networks', *Comput. Netw.*, 2016, **109**, pp. 172–182

[2] AlErouda, A., Alsmadib, I.: 'Identifying cyber-attacks on software defined networks: An inference-based intrusion detection approach', *J. Netw. Comput. Appl.*, 2017, **80**, pp. 152–164

[3] Cui, Y., Yan, L., Li, S., *et al.*: 'SD-Anti-DDoS: fast and efficient DDoS defense in software-defined networks', *J. Netw. Comput. Appl.*, 2016, **68**, pp. 65–79

[4] Hong, S., Xu, L., Wang, H., *et al.*: 'Poisoning network visibility in software-defined networks: new attacks and countermeasures'. Proc. NDSS. 22nd Annual Network and Distributed System Security Symp., California, USA, February 2015, pp. 1–15

[5] Garcıa-Teodoroa, P., Dıaz-Verdejoa, J., Macia-Fernandeza, G., *et al.*: 'Anomaly-based network intrusion detection: techniques, systems and challenges', *Comput. Secur.*, 2009, **28**, pp. 18–28

[6] Tang, T.A., Mhamdi, L., McLernon, D., *et al.*: 'Deep learning approach for network intrusion detection in software defined networking'. Proc. Int. Conf. on Wireless Networks and Mobile Communications, Fez, Morocco, October 2016, pp. 258–263

[7] Braga, R., Mota, E., Passito, A.: 'Lightweight DDoS flooding attack detection using NOX/OpenFlow'. Proc. IEEE 35th Conf. on Local Computer Networks, Denver, USA, October 2010, pp. 408–415

[8] Laskov, P., Düssel, P., Schäfer, C., *et al.*: 'Learning intrusion detection: supervised or unsupervised?'. Proc. 13th Int. Conf. on Image Analysis and Processing–ICIAP, Cagliari, Italy, September 2005, pp. 50–57

[9] Gharibian, F., Ghorbani, A.A.: 'Comparative study of supervised machine learning techniques for intrusion detection'. Proc. IEEE Fifth Annual Conf. on Communication Networks and Services Research, New Brunswick, Canada, May 2007, pp. 350–358

[10] Tavallaee, M., Bagheri, E., Lu, W., *et al.*: 'A detailed analysis of the KDD CUP 99 data set'. Proc. IEEE Symp. on Computational Intelligence for Security and Defense Applications, Ontario, Canada, July 2009, pp. 1–6

[11] Panda, M., Abraham, A., Patra, M.R.: 'Discriminative multinomial naive Bayes for network intrusion detection'. Proc. IEEE Sixth Int. Conf. on Information Assurance and Security, Atlanta, Canada, August 2010, pp. 5–10

[12] Aziz, A.S.A., Hassanien, A.E., Hanaf, S.E.O., *et al.*: 'Multi-layer hybrid machine learning techniques for anomalies detection and classification approach'. Proc. IEEE 13th Int. Conf. on Hybrid Intelligent Systems, Gammarth, Tunisia, December 2013, pp. 215–220

[13] Thaseen, S., Kumar, C.A.: 'An analysis of supervised tree based classifiers for intrusion detection system'. Proc. IEEE Int. Conf. on Pattern Recognition, Informatics and Mobile Engineering, Salem, India, February 2013, pp. 294–299

[14] Chauhan, H., Kumar, V., Pundir, S., *et al.*: 'A comparative study of classification techniques for intrusion detection'. Proc. IEEE Int. Symp. on Computational and Business Intelligence, New Delhi, India, August 2013, pp. 40–43

[15] Ingre, B., Yadav, A.: 'Performance analysis of NSL-KDD dataset using ANN'. Proc. IEEE Int. Conf. on Signal Processing and Communication Engineering Systems, Guntur, India, January 2015, pp. 92–96

[16] Belavagi, M.C., Muniyal, B.: 'Performance evaluation of supervised machine learning algorithms for intrusion detection'. Proc. Twelfth Int. Multi-Conf. on Information Processing, Bangalore, India, December 2016, pp. 117–123

[17] Staudemeyer, R., Omlin, C.W.: 'Feature set reduction for automatic network intrusion detection with machine learning algorithms'. Proc. southern African Telecommunication Networks and Applications Conf. (SATNAC), Swaziland, Swaziland, August 2009

[18] Taylor, J., King, R.D., Altmann, T., *et al.*: 'Application of metabolomics to plant genotype discrimination using statistics and machine learning', *Bioinformatics*, 2002, **18**, pp. S241–S248

[19] Ikram, S.T., Cherukuri, A.K.: 'Improving accuracy of intrusion detection model using PCA and optimized SVM', *J. Comput. Inf. Technol.*, 2016, **24**, pp. 133–148

[20] Bostanci, B., Bostanci, E.: 'An evaluation of classification algorithms using McNemar's test'. Proc. Seventh Int. Conf. on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012), Gwalior, India, December 2012, pp. 15–26

[21] Wettschereck, D., Aha, D.W., Mohri, T.: 'A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms', *Artif. Intell. Rev.*, 1997, **11**, (1–5), pp. 273–314

[22] Tang, J., Deng, C., Huang, G.B.: 'Extreme learning machine for multilayer perceptron', *IEEE Trans Neural Netw. Learn.*, 2016, **27**, (4), pp. 809–821