J.Natn.Sci.Foundation Sri Lanka 2018 46 (4): 601 - 612
DOI: http://dx.doi.org/10.4038/jnsfsr.v46i4.8560

RESEARCH ARTICLE

Two-layer malicious network flow detection system with sparse linear model based feature selection

Ferhat Ozgur Catak*
TUBITAK-BILGEM Cyber Security Institute, Kocaeli, Turkey.

Revised: 08 March 2018; Accepted: 22 June 2018

Abstract: The amount of malicious network traffic in enterprise systems has increased due to the spreading of botnets, fuzzers, shellcodes or exploits, which threatens everyday operation of enterprises. Building classification models from this malicious traffic is an important issue. Classification models can help to discover new types of attacks based on previously built predictive models. The most prominent attacks on accessibility in the CIA Triad are distributed denial-of-service attacks. By using denial-of-service attacks targeted at the availability of CIA triad, it is intended to block access to services for legitimate users who need to be connected to the service. Just like the Mirai cyber-attack, major service providers such as Twitter and Reddit can become inaccessible by simply attacking the DNS servers. Hence, distributed denial-of-service, a rather old type of attack, is still valid today. This paper describes two-stage filtering based network traffic identification based on network flow patterns. The paper also shows that the predictive performance of the malicious traffic classification model increases with the filtering of network flow. *L1*-norm based sparse linear models were used for feature selection to find an optimal feature set and determine the effect of different features. Simulation results validate the effectiveness of the proposed classification scheme.

Keywords: Cyber security, feature selection, malicious network flow, sparse linear models.

INTRODUCTION

Service providers or enterprises in the Internet are regularly targeted by different kinds of cyber-attacks, including propagation of botnet traffic, distributed denial of service (DDoS), backdoors, exploits, shellcode and fuzzers (Ben-Asher & Gonzalez, 2015). The malicious network traffic within a service provider may negatively impact the network performance regarding enormous traffic and may block service to the legitimate users (Jiang *et al*., 2014; Cappers & Wijk, 2015). Various network security peripherals such as firewalls or intrusion detection systems can shelter the network against some forms of attacks. The security mechanisms are generally signature based. Hence, they are not adaptive with the malicious traffic fluctuations or new types of network packet patterns.

It is crucial to be able to detect malicious network flows fast, precisely and in real time (Han *et al*., 2016). In DDoS attacks, either a compromised machine or a cluster of compromised machines together send an enormous amount of network traffic to the server to exhaust the resources. Fuzzing is another technique that is an automated black-box software testing technique, which involves finding implementation errors using injection of invalid, random or unexpected data to a software.

Such abnormal changes could be detected using descriptive statistical methods. Feinstein *et al.* (2003) used Chi-square statistics to identify network flow anomalies and time window-based entropy changes were proposed in network flow to detect malicious traffic.

* ozgur.catak@tubitak.gov.tr; https://orcid.org/0000-0002-2434-9966

Descriptive statistics-based detection methods rely on prior known data. A noticeable difficulty is that network flow anomalies are a timely changing target (Bhuyan *et al.*, 2015). It is difficult to accurately characterise the set of malicious network traffic. A new type of malicious traffic will continue to appear over time. As a result, malicious network classification models must avoid the over-fit to any predefined set of malicious traffic classes (Srihari & Anitha, 2014).

Over the years, malicious network traffic classification modelling has been a major research area in the industry and many schemes have been proposed by various researchers. Previous research on this topic can be observed in the signature-based deep packet inspection (DPI) methods for malicious signature, flow analysis, anomaly-based detection, DNS-based detection and data mining techniques (Silva *et al.*, 2013; Jang-Jaccard & Nepal, 2014; Khattak *et al.*, 2014; Bekerman *et al.*, 2015).

Fernandes *et al.* (2015) have proposed a profile-based anomaly detection system using digital signature of network segment using flow analysis (DSNF). DSNF were generated *via* principal component analysis. The approach has low computational complexity and shows high applicability for automatic anomaly identification. Another method is ant colony optimisation-based network anomaly detection (Fernandes *et al*., 2016), where a seven-dimensional analysis of IP flow is performed. This has satisfactory performance on false-positive rates.

Goebel and Holz (2007) proposed an effective model for detecting IRC-based botnets within a given network based on an evaluation of well-known IRC channel name patterns. This approach is mainly based on passively monitoring a network for unregularly nicknames, servers and server ports.

Shiaeles *et al.* (2012) proposed a fuzzy estimator on the mean packet inter-arrival times. The approach contains two steps, including the actual detection of the DDoS attack and detecting the offending IP addresses. In the first step, the mean packet arrival for each IP address is compared with a fuzzy model. Then, in the DDoS situation, network traffic is compared with the previous fuzzy model to detect the attack. The model's accuracy result is 80 %, which is a low prediction performance.

Rahbarinia *et al.* (2014) introduced a P2P botnet traffic detection system that can identify malicious P2P botnets called *PeerRush*. The approach achieves these results without the need of DPI or signatures, and the model precisely identifies applications that use encrypted traffic. The features consist of inter-packet delays, duration of the connection and bi-directionality. The dataset labels are divided into two different classes; the first one is legal P2P networks, including Skype, eMule and uTorrent, the second one is illegal Zeus botnet traffic. The detection model performs both malicious and legitimate traffic, with an accuracy of 97.9 % and the false positive rate is 1.2 %.

Livadas *et al.* (2006) proposed machine learning techniques to identify the command and control traffic of IRC-based botnets. The proposed model firstly classifies network traffic into chat or non-chat labels using machine learning algorithms, and then tries to find out whether chat traffic is malicious or non-malicious. The first classifier in the model has $10 - 20$ % false negative rate and $30 - 40$ % false positive rate, which is relatively high. Lu *et al.* (2011) proposed a new method for detecting and clustering botnet traffic on large-scale network application communities. The approach is based on the 256 ASCII bytes on the payload of the packets over a predefined time window to categorise malicious IRC-based botnet network traffic from normal network traffic. As they said, same botnet is programmed by the botmaster on how and when to respond to received commands. The authors used standard deviation metric for botnet network flow decision.

Dittrich and Dietrich (2008) showed that new botnet paradigm is shifting. P2P - based botnets switch between TCP and UDP protocols and randomise port numbers in order to evade security modules such as firewall rules. Also, the newest botnets use probability-based delegation of a function by using arbitrary sleep call. For this reason, the detectors to be developed need to be adaptive to sense these changes in the network flow.

Contributions

This study detects network attacks using windowing based training samples by means of machine learning algorithms. Various algorithms were used to separate normal network traffic from some kind of attacks, including Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms. The main objective of the proposed model is to use a two-layer approach for exact classification of malicious flow. The first layer detects if the flow is malicious or not malicious. If malicious flow is detected, then the second layer tries to find out the exact malicious class. The proposed model does not rely on signature-based mechanisms or deep packet inspection. Both methods are easily attacked using encryption methods.

In summary, the proposed classification scheme makes the following contributions:

- A novel two-stage classification technique is presented for the detection of malicious network flow that is based on machine learning algorithms. Hence, the proposed model can avoid overfitting to pre-defined malicious classes. Layer 1 of the model decides whether this flow is malicious or not. If it is assigned the malicious label to the network flow, then layer 2 assigns the exact class to the source of network flow.
- Randomised sparse models were used with *L1* regularisation to select the appropriate feature set to increase the prediction performance and to decrease the training complexity. Hence, the proposed model requires less training time.
- The proposed model is very practical. It relies on sampled network flow data (most enterprises collect it using software tools in pcap file format).
- The implementation of the proposed classification system is described by using large volumes of packet capture (pcap) files that are publicly available on the Internet for benchmarking of network classification models.

The rest of the paper describes the approach for ML-based two-layer application identification and evaluation of the approach using traffic traces.

METHODOLOGY

The proposed model relies on two fundamental concepts. First, the model identifies whether the network traffic is legal or not. Then, it utilises the binary classification models during the network flow in order to identify the exact malicious class. In the second layer, multi-class classification-based classification methods are applied to detect the attack type. The proposed system extracts important features using randomised sparse models with *L1* regularisation and the extracted features are used to build two different models. The first model identifies if the network flow is legal or not and this structure can be considered as a filter. The model attempts to differentiate malicious traffic from legitimate traffic by leveraging on the data patterns of communication amongst hosts. Figure 1 shows the block diagram of the proposed model. The model consists of two different layers. If there is no malicious activity in the incoming packet, the proposed model completes the classification process in the first layer as a normal label to the traffic.

Classification algorithms

The model uses different machine learning-based classification techniques, which can be used to detect malicious network traffic. The selected algorithms are decision tree, random forest, neural net, AdaBoost and naive Bayes.

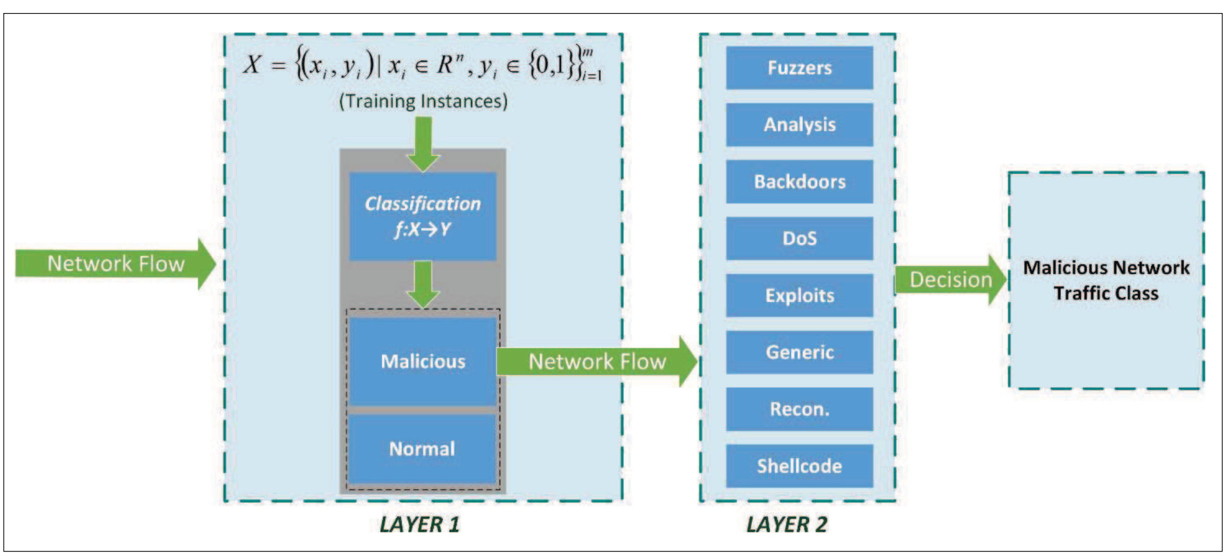


Figure 1: Architectural block diagram of the proposed model. Layer 1 acts like a filter. Layer 2 classifies the malicious traffic with exact label.

Dataset

Most of the intrusion detection work in literature uses a very old dataset, KDDCUP'99. This dataset contains 4 different types of attacks such as DoS, R2L, U2R and probing in tcp dump format. The number of attack labels this dataset has is quite low. In addition, the KDDCUP'99 dataset contains 41 predefined attributes. Tavallaee *et al.* (2009) showed some important issues which affect the performance of evaluated systems, and results in a very poor evaluation of anomaly detection approaches (Tavallaee *et al.*, 2009). For these reasons, it is not correct to use the KDDCUP'99 dataset as a benchmark for research purposes, but a more recent dataset should be used instead.

The Australian Centre for Cyber Security (ACCS) created a hybrid of the real modern normal and the current arranged attack activities of the network traffic (Moustafa & Slay, 2015). They published the dataset for research purposes and it can be downloaded from the Internet. In order to create a hybrid of the modern normal and abnormal network traffic, they utilised the IXIA PerfectStorm tool. The dataset contains nine different attack families, and are shown in the attack types Table 1.

In order to reduce the computational complexity of DPI methods, instead, we used packet features to build classification models. We used 40 different features from network traces. Table 2 shows all the features used to build models in our approach.

Table 1: UNSW dataset attack types

Attack	No. of rows	Attack description
Normal	2,218,761	Normal traffic data
Fuzzers	24,246	Fuzzification (randomly generated data) based attacks
Analysis	2,677	Port scan, spam and html files penetrations
Backdoors	2,329	Security system bypassing method in order to access a computer
DoS	16,353	System resource exhasuting method to interrupt or suspend the services of a host
Exploits	44,525	Using known security problem (vulnerability), attackers send exploits to penetrate a system
Generic	215,481	A technique works against all blockciphers
Reconnaissance	13,987	Information gathering
Shellcode	1,511	The payload in the exploitation of software vulnerability
Worms	174	The network based malicious piece of code that replicates itself

Table 2: Features used in model building

Feature	Description
Sport	Source port number
Dsport	Destination port number
Dur	Record total duration
Sbytes	Source to destination transaction bytes
Dbytes	Destination to source transaction bytes
Sttl	Source to destination time to live value
Dttl	Destination to source time to live value
Sloss	Source packets retransmitted or dropped
Dloss	Destination packets retransmitted or dropped
Sload	Source bits per second
Dload	Destination bits per second
Spkts	Source to destination packet count
Dpkts	Destination to source packet count

Continued –

– continued from page 604

Feature	Description
Swin	Source TCP window advertisement value
Dwin	Destination TCP window advertisement value
stcpb	Source TCP base sequence number
dtcpb	Destination TCP base sequence number
smeansz	Mean of the row packet size transmitted by the src
dmeansz	Mean of the row packet size transmitted by the dst
trans_depth	Represents the pipelined depth into the connection of http request/response transaction
res_bdy_len	Actual uncompressed content size of the data transferred from the server's http service
Sjit	Source jitter (mSec)
Djit	Destination jitter (mSec)
Sintpkt	Source interpacket arrival time (mSec)
Dintpkt	Destination interpacket arrival time (mSec)
tcprtt	TCP connection setup round-trip time, the sum of SYN_ACK and ACK_DAT
synack	TCP connection setup time, the time between the SYN and the SYN_ACK packets
ackdat	TCP connection setup time, the time between the SYN_ACK and the ACK packets
is_sm_ips_ports	If source (1) and destination (3) IP addresses equal and port numbers equal then, this variable takes value 1 else 0
ct_state_ttl	No. for each state according to specific range of values for source/destination time to live (10) (11)
ct_flw_http_mthd	No. of flows that has methods such as Get and Post in http service
is_ftp_login	If the ftp session is accessed by user and password then 1 else 0
ct_ftp_cmd	No. of flows that has a command in ftp session
ct_srv_src	No. of connections that contain the same service and source address in 100 connections according to the last time
ct_srv_dst	No. of connections that contain the same service and destination address in 100 connections according to the last time
ct_dst_ltm	No. of connections of the same destination address in 100 connections according to the last time
ct_src_ltm	No. of connections of the same source address in 100 connections according to the last time
ct_src_dport_ltm	No. of connections of the same source address and the destination port in 100 connections according to the last time
ct_dst_sport_ltm	No. of connections of the same destination address and the source port in 100 connections according to the last time
ct_dst_src_ltm	No. of connections of the same source and the destination address in 100 connections according to the last time
Label	0 for normal and 1 for attack records

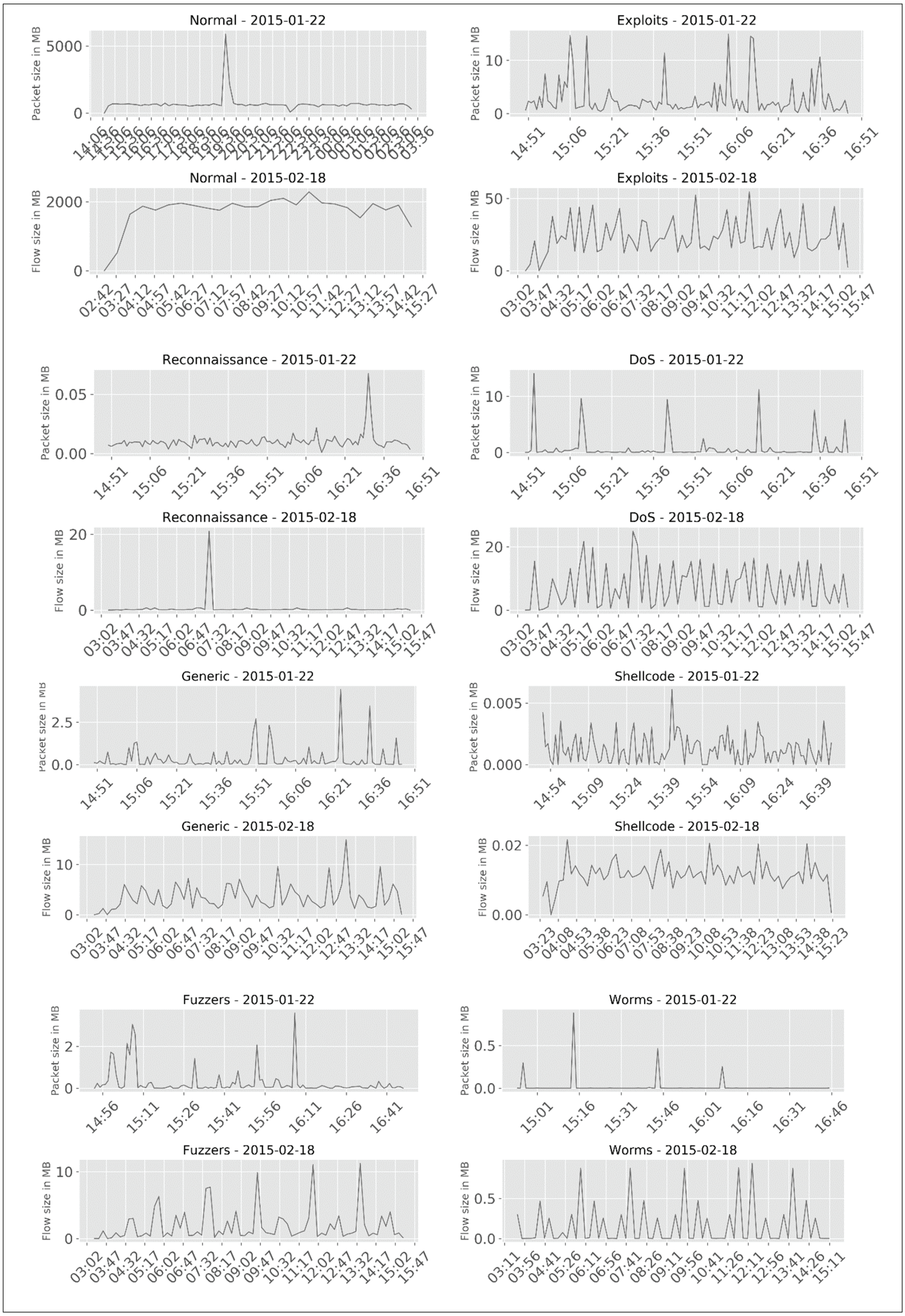
If we examine the dataset, it appears that the network traffic is at two different time intervals. It was observed that the size of the normal and attack network traffic are different in both different time intervals.

In order to make these differences clearer, the size of the network traffic was visualised for each class. Figure 2 shows normal and malicious network traffic flows at two different time frames.

Feature selection

Although we have used 36 different features, feature selection method was used to determine which attributes are important for the classification process. Feature extraction improves the accuracy of the classification model and at the same time reduces the model's time to learning (Guyon & Elisseeff, 2003). There are many studies about attribute selection in literature (Xue *et al*., 2013; Chandrashekar & Sahin, 2014). *L1*-norm-based sparse linear classifiers can also be used in attribute selection. By using the weight coefficient value that the linear model has assigned to the features, attributes with zero value are removed and the rest of the features are used to construct the classification model (Chen *et al*., 2017).

In machine learning, linear classifier models build sparse solutions using *L1*-norm-based cost function. Many of the predicted coefficients are zero, thus, sparse classifier functions are useful for the feature selection. Logistic regression and support vector machine classification algorithms can be used to create feature selection models.



Continued -

– continued from page 606

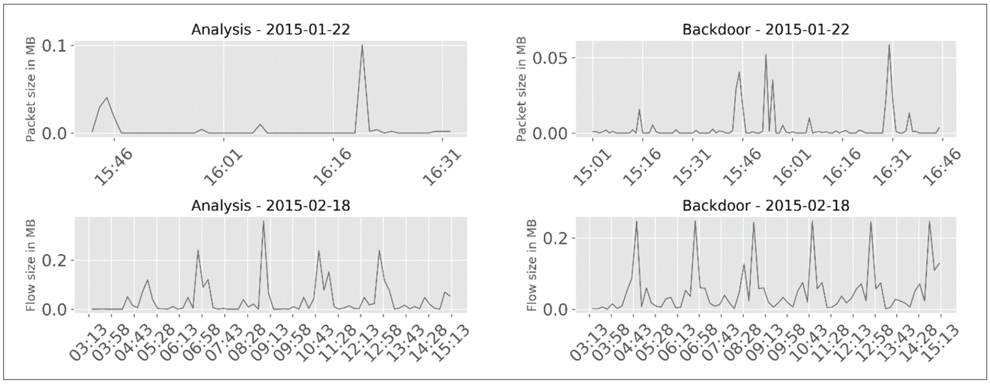


Figure 2: Normal and malicious network traffic flows at two different time frames

There are some well-known problems with L1 based sparse models for classification problems. The classification model tends to select a feature out of a group of highly correlated variables. Even when a correlation is not too high, L1 tries to select one single feature from the group of correlated variables. In order to overcome the correlation problem, Meinshausen and Bühlmann (2010) proposed randomisation-based techniques. Given a dataset from iid $X \in \mathbb{R}^{m\times n}$, then the random subset of the data of size n_S is defined as $(\boldsymbol{x}_s, y_s)\ s \in S\ where\ S \subset \{1, 2, \cdots, n\}$. The modified lasso fit is obtained as shown in equation (1).

$$\widehat{w}_s = \arg\min_{\mathrm{w}} \frac{1}{2n_s}\sum_{s\in S}(y_i - \boldsymbol{x}_i^t\boldsymbol{w})^2 + \alpha\sum_{j=1}^{p}\frac{|w_j|}{t_j} \qquad ...(1)$$

where $t_j \in \{t, 1\}$ are independent trails of a fair Bernoulli random variable, and the scaling factor *t* is $0 < t < 1$.

Evaluation metrics

The following standard model evaluation metrics were used to find the classification performance of botnet detection classifiers. Since the datasets that are used in the experiments are highly imbalanced, traditional accuracy-based performance evaluation is not enough to find out an optimal classifier. We used four different metrics; the overall prediction accuracy, average recall, average precision (Turpin & Scholer, 2006) and F_1 score, to evaluate the classification accuracy which are common measurement metrics in information retrieval (Makhoul *et al*., 1999).

Precision is defined as the fraction of retrieved samples that are relevant. Precision is shown in equation (2).

$$Precision = \frac{Correct}{Correct + False} \qquad ...(2)$$

Recall is defined as the fraction of relevant samples that is retrieved. Recall is shown in equation (3).

$$Precision = \frac{Correct}{Correct + Missed} \qquad ...(3)$$

The proposed evaluation model calculates the precision and recall for each class from prediction scores, then finds their mean. Average precision and recall are shown in equations (4) and (5).

$$Precision_{avg} = \frac{1}{n_{classes}}\sum_{i=0}^{n_{classes}-1} Prec_i \qquad ...(4)$$

$$Recall_{avg} = \frac{1}{n_{classes}}\sum_{i=0}^{n_{classes}-1} Recall_i \qquad ...(5)$$

F_1-measure is defined as the harmonic mean of precision and recall.

$$F_1 = 2 \times \frac{Prec_{avg} \times Recall_{avg}}{Prec_{avg} + Recall_{avg}} \qquad ...(6)$$

RESULTS AND DISCUSSION

In order to evaluate the proposed model, we conducted a series of experiments. The main objective of this work is to improve the classification performance of the model by using a 2-layer architecture. Information on feature selection process using the *L1*-norm based sparse linear model is given in tables and figures in the next section. In the 'classifier performance evaluation' section, we share the results obtained from the experiments with tables in the form of feature selection and without feature selection with five different classification methods.

Feature selection

We obtained the weight coefficient of each attribute found in the dataset using the *L1*-norm based sparse linear model. Zero-valued attributes were removed from the dataset before the training phase of the model creation process. Thus, 11 features were removed from the data set and the model was constructed using the remaining 25 attributes. The important values of the selected attributes are shown in Table 3 and Figure 3. By applying the feature extraction process to the dataset, it is aimed to complete the training phase of model building in a shorter time.

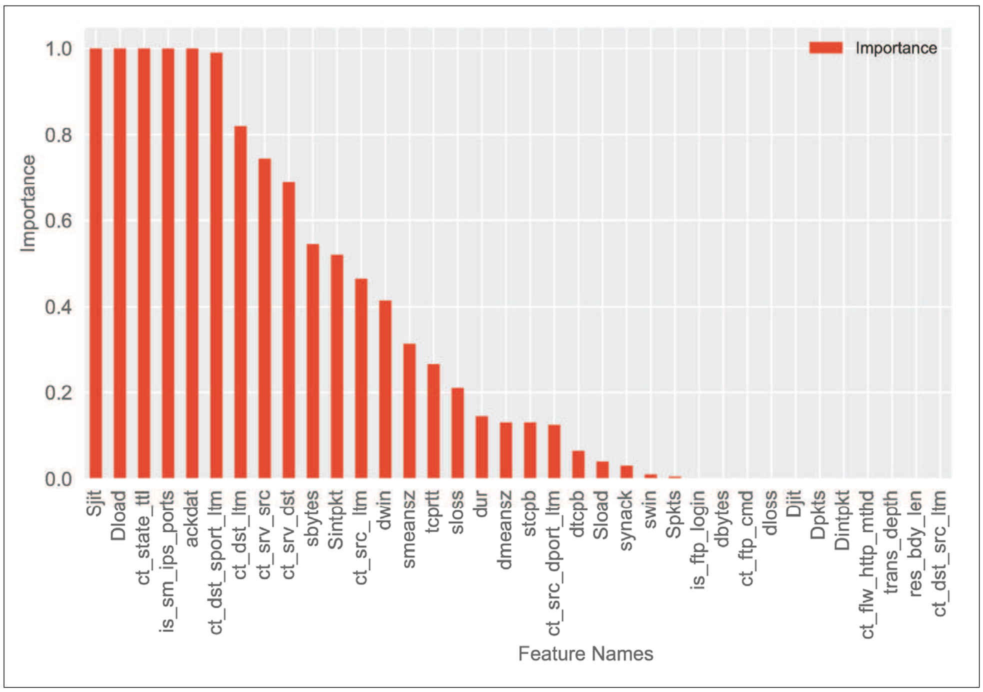


Figure 3: Feature importance values after training lasso model

Table 3: Sparcity of the feature weights after training the lasso model

Feature name	Importance	Feature name	Importance	Feature name	Importance
Sjit	1.000	dwin	0.415	Spkts	0.005
Dload	1.000	smeansz	0.315	is_ftp_login	0.000
ct_state_ttl	1.000	tcprtt	0.265	dbytes	0.000
is_sm_ips_ports	1.000	sloss	0.210	ct_ftp_cmd	0.000
ackdat	1.000	dur	0.145	dloss	0.000
ct_dst_sport_ltm	0.990	dmeansz	0.130	Djit	0.000
ct_dst_ltm	0.820	stcpb	0.130	Dpkts	0.000
ct_srv_src	0.745	ct_src_dport_ltm	0.125	Dintpkt	0.000
ct_srv_dst	0.690	dtcpb	0.065	ct_flw_http_mthd	0.000
sbytes	0.545	Sload	0.040	trans_depth	0.000
Sintpkt	0.520	synack	0.030	res_bdy_len	0.000
ct_src_ltm	0.465	swin	0.010	ct_dst_src_ltm	0.000

Classifier performance evaluation

The proposed system is implemented using 64-bit Python 2.7 using scikit-learn machine learning library for classification models, Pandas data analysis toolkit for parsing csv file. We divided 80 % of the input dataset into training and 20 % as the test dataset randomly. The training phase of each algorithm was repeated five times and the average of the model measurements was calculated.

The results produced by five different classification algorithms without feature selection are shown in Table 4. From the table, one can infer that the random forest algorithm-based classification model achieves the highest accuracy of 99.43 %. Also, the random forest has high traffic classification accuracy and precision/recall rates.

Table 4: Layer 1 classification results without feature selection

Classifier	Precision	Recall	F_1	Accuracy
Decision tree	0.92	0.99	0.95	0.9876
Random forest	0.98	0.98	0.98	0.9943
Neural net	0.81	0.01	0.03	0.8750
AdaBoost	0.89	1.00	0.94	0.9846
Naive Bayes	0.46	0.84	0.59	0.8312

Table 5 shows the corresponding confusion matrix of the best classification result of the random forest algorithm.

Table 5: Random forest classifier confusion matrix without feature selection

		Actual		
		Positive	Negative	Total
Predicted	Positive	663,488	1,950	665,438
	Negative	2,383	94,194	96,577
	Total	665,871	96,144	762,015

The results produced by five different classification algorithms with feature selection are shown in Table 6. From the table, one can infer that the random forest algorithm-based classification model achieves the highest accuracy of 99.40 %, which is almost the same with Table 4. Also, the random forest has high traffic classification accuracy and precision/recall rates.

Table 7 shows the corresponding confusion matrix of the best classification result of the random forest algorithm with feature selection.

Table 6: Layer 1 classification results with feature selection

Classifier	Precision	Recall	F_1	Accuracy
Decision tree	0.92	0.99	0.95	0.9869
Random forest	0.98	0.98	0.98	0.9940
Neural net	0.75	0.02	0.03	0.8748
AdaBoost	0.95	0.96	0.95	0.9884
Naive Bayes	0.44	0.85	0.58	0.8414

Table 7: Random forest classifier confusion matrix with feature selection

		Actual		
		Positive	Negative	Total
Predicted	Positive	663,160	2,279	665,439
	Negative	2,300	94,276	96,576
	Total	665,460	96,555	762,015

Table 8: Second classifier decision tree results

	All features			Feature selection		
	Prec	Recall	F_1	Prec	Recal	F_1
Fuzzers	0.78	0.76	0.77	0.77	0.79	0.78
Backdoor	0.00	0.00	0.00	0.00	0.00	0.00
DoS	1.00	0.00	0.00	1.00	0.00	0.00
Exploits	0.55	0.90	0.69	0.54	0.94	0.68
Generic	1.00	0.98	0.99	1.00	0.97	0.99
Recon.	0.74	0.67	0.70	0.91	0.60	0.72
Shellcode	0.00	0.00	0.00	0.00	0.00	0.00
Worms	0.00	0.00	0.00	0.00	0.00	0.00

Table 9: Second classifier random forest results

	All features			Feature selection		
	Prec	Recall	F_1	Prec	Recal	F_1
Fuzzers	0.78	0.76	0.77	0.77	0.79	0.78
Backdoor	0.00	0.00	0.00	0.00	0.00	0.00
DoS	1.00	0.00	0.00	1.00	0.00	0.00
Exploits	0.55	0.90	0.69	0.54	0.94	0.68
Generic	1.00	0.98	0.99	1.00	0.97	0.99
Recon.	0.74	0.67	0.70	0.91	0.60	0.72
Shellcode	0.00	0.00	0.00	0.00	0.00	0.00
Worms	0.00	0.00	0.00	0.00	0.00	0.00

Table 8 shows the results obtained using the decision tree classifier. Feature selection-based classification model results are almost the same with the all feature-based model.

Table 9 shows the results obtained using random forest classifier. The algorithm creates multiple trees, which are constructed using a random sample of the feature set. Feature selection based classification model results are almost the same with the all feature based model.

Table 10: Second classifier neural network results

	All features			Feature selection		
	Prec	Recall	F_1	Prec	Recal	F_1
Fuzzers	0.29	0.65	0.40	0.05	0.63	0.10
Backdoor	0.00	0.00	0.00	0.00	0.00	0.00
DoS	0.27	0.03	0.06	0.27	0.06	0.09
Exploits	0.66	0.08	0.14	0.62	0.84	0.71
Generic	0.82	0.98	0.89	0.49	0.39	0.44
Recon.	0.00	0.00	0.00	0.07	0.00	0.00
Shellcode	0.00	0.00	0.00	0.00	0.00	0.00
Worms	0.00	0.00	0.00	0.00	0.00	0.00

Table 10 shows the results obtained using neural network classifier.

Table 11: Second classifier AdaBoost results

	All features			Feature selection		
	Prec	Recall	F_1	Prec	Recall	F_1
Fuzzers	0.61	0.64	0.63	0.78	0.52	0.61
Backdoor	0.00	0.00	0.00	0.00	0.00	0.00
DoS	0.29	0.00	0.01	0.32	0.49	0.29
Exploits	0.53	0.76	0.63	0.59	0.62	0.53
Generic	0.98	0.97	0.98	0.98	0.97	0.98
Recon.	0.61	0.63	0.62	0.65	0.74	0.61
Shellcode	0.00	0.00	0.00	0.26	0.05	0.00
Worms	0.00	0.00	0.00	0.00	0.00	0.00

Table 11 shows the results obtained using AdaBoost classifier.

Table 12: Accuracy results

Classifier	All Feat. Acc.	Feat. Sel. Acc.
Decision tree	0.8692	0.8677
Random forest	0.8921	0.8904
Neural net	0.7216	0.1047
Adaboost	0.8352	0.8356

Table 12 shows the accuracy results of all classification algorithms at layer 2 of the proposed algorithm.

Training time

In order to demonstrate the time performance of the proposed method, we first trained the dataset only with single-stage multi-class classification algorithms. Table 13 shows the total training time of the selected algorithms for one-shot and the proposed method. We observed that the proposed method had better time performance over the training phase.

Table 13: One-shot classification training performance

Algorithm	One-shot time (sn)	Proposed (sn)
Decision tree	45.933	10.105
Random forest	5.687.722	799.609
Neural net	4.230.729	422.746
AdaBoost	4.104.794	183.194

CONCLUSION

In this study, we presented a novel model for the malicious network traffic classification that relies on 2-layers classification models and selects features based on sparse linear model. The accuracy of the model will be reduced if only one algorithm is used to determine the class label of a malicious network traffic. For this reason, the use of a layered classification model increases both classification performance and classification accuracy. In the proposed model, layer 1 acts like a filter, and harmless traffic does not move to layer 2.

Using the *L1*-norm-based sparse linear model, unnecessary features have been removed from the dataset, and thus the complexity of the proposed model is reduced. For this reason, the training phase of the model is simpler than the model in which all attributes are used. When the tables in the results section are examined, it is observed that the accuracy performance of the proposed model is high. Some of the harmful network traffic labels in layer 2 may have lower classification performance than other classes. The classes with low class distribution in the dataset also have low classification performance. As future work, we will focus on labels with low class distribution to improve classification performance. In layer 2, we are planning to use the appropriate model of protocol (http, ftp etc.) type instead of attack class at the same time. Hence the performance of classes with fewer examples in the dataset will increase.

REFERENCES

Bekerman D., Shapira B., Rokach L. & Bar A. (2015). Unknown malware detection using network traffic classification. *2015 IEEE Conference on Communications and Network Security (CNS)*, Florence, Italy, 28 − 30 September, pp. 134 – 142.
DOI: https://doi.org/10.1109/CNS.2015.7346821

Ben-Asher N. & Gonzalez C. (2015). Effects of cyber security knowledge on attack detection. *Computers in Human Behavior* **48**: 51 – 61.
DOI: https://doi.org/10.1016/j.chb.2015.01.039

Bhuyan M.H., Bhattacharyya D. & Kalita J. (2015). An empirical evaluation of information metrics for low-rate and high-rate DDoS attack detection. *Pattern Recognition Letters* **11**: 1 – 7.
DOI: https://doi.org/10.1016/j.patrec.2014.07.019

Cappers B.C. & Wijk J.J. (2015). SNAPS: Semantic network traffic analysis through projection and selection. *2015 IEEE Symposium on Visualization for Cyber Security (VizSec)*, Chicago, USA, 25 October, pp. 1 – 8.
DOI: https://doi.org/10.1109/VIZSEC.2015.7312768

Chandrashekar G. & Sahin F. (2014). A survey on feature selection methods. *Computers and Electrical Engineering* **40**(1): 16 – 28.
DOI: https://doi.org/10.1016/j.compeleceng.2013.11.024

Chen X., Zhou G., Chen Y., Shao G. & Gu Y. C. (2017). Supervised multiview feature selection exploring homogeneity and heterogeneity with $l_{1,2}$ -norm and automatic view generation. *IEEE Transactions on Geoscience and Remote Sensing* **555**(4): 2074 – 2088.
DOI: https://doi.org/10.1109/TGRS.2016.2636329

Dittrich D. & Dietrich S. (2008). P2P as botnet command and control: a deeper insight. *3rd International Conference on Malicious and Unwanted Software (MALWARE)*, Fairfax, USA, 7 – 8 October, pp. 41 – 48.
DOI: https://doi.org/10.1109/MALWARE.2008.4690856

Feinstein L., Schnackenberg D., Balupari R. & Kindred D. (2003). Statistical approaches to DDoS attack detection and response. *Proceedings of DARPA Information Survivability Conference and Exposition*, Washington DC, USA, 22 – 24 April, pp. 303 – 314.
DOI: https://doi.org/10.1109/DISCEX.2003.1194894

Fernandes G., Carvalho L.F., Rodrigues J.J. & Proença M.L. (2016). Network anomaly detection using IP flows with principal component analysis and ant colony optimization. *Journal of Network and Computer Applications* **64**: 1 – 11.
DOI: https://doi.org/10.1016/j.jnca.2015.11.024

Fernandes G., Rodrigues J.J. & Proença M.L. (2015). Autonomous profile-based anomaly detection system using principal component analysis and flow analysis. *Applied Soft Computing* **34**: 513 – 525.
DOI: https://doi.org/10.1016/j.asoc.2015.05.019

Goebel J. & Holz T. (2007). Rishi: identify bot contaminated hosts by IRC nickname evaluation. *Proceedings of the 1st Workshop on Hot Topics in Understanding Botnets (HotBots)* Cambridge, MA, USA, 10 April, p. 8.

Guyon I. & Elisseeff A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research* **3**: 1157 – 1182.

Han F., Xu L., Yu X., Tari Z., Feng Y. & Hu J. (2016). Sliding-mode observers for real-time DDoS detection. *IEEE 11th Conference on Industrial Electronics and Applications (ICIEA)*, Hefei, China, 5 – 7 June, pp. 825 – 830.
DOI: https://doi.org/10.1109/ICIEA.2016.7603695

Jang-Jaccard J. & Nepal S. (2014). A survey of emerging threats in cybersecurity. *Journal of Computer and System Sciences* **80**(5): 973 – 993.
DOI: https://doi.org/10.1016/j.jcss.2014.02.005

Jiang D., Xu Z., Zhang P. & Zhu T. (2014). A transform domain-based anomaly detection approach to network-wide traffic. *Journal of Network and Computer Applications* **40**: 292 – 306.
DOI: https://doi.org/10.1016/j.jnca.2013.09.014

Khattak S., Ramay N.R., Khan K.R., Syed A.A. & Khayam S.A. (2014). A taxonomy of botnet behavior, detection, and defense. *IEEE Communications Surveys Tutorials* **16**(2): 898 – 924.
DOI: https://doi.org/10.1109/SURV.2013.091213.00134

Livadas C., Walsh R., Lapsley D. & Strayer W.T. (2006). Usilng machine learning technliques to identify botnet traffic. *Proceedings of the 31st IEEE Conference on Local Computer Network*s, Tampa, USA, 14 – 16 November, pp. 967 – 974.
DOI: https://doi.org/10.1109/LCN.2006.322210

Lu W., Rammidi G. & Ghorbani A.A. (2011). Clustering botnet communication traffic based on n-gram feature selection. *Computer Communications* **34**(3): 502 – 514.
DOI: https://doi.org/10.1016/j.comcom.2010.04.007

Makhoul J., Kubala F., Schwartz R. & Weischedel R. (1999). Performance measures for information extraction. *Proceedings of DARPA Broadcast News Workshop,* pp. 249 – 252. Morgan Kaufmann Publishers, San Francisco, USA.

Meinshausen N. & Bühlmann P. (2010). Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **72**(4): 417 – 473.
DOI: https://doi.org/10.1111/j.1467-9868.2010.00740.x

Moustafa N. & Slay J. (2015). UNSW-NB15: a comprehensive dataset for network intrusion detection systems (UNSW-NB15 network dataset). *Military Communications and Information Systems Conference (MilCIS)*, Canberra, Australia, 10 – 12 November, pp. 1 – 6. DOI: https://doi.org/10.1109/MilCIS.2015.7348942

Rahbarinia B., Perdisci R., Lanzi A. & Li K. (2014). PeerRush: Mining for unwanted P2P traffic. *Journal of Information Security and Applications* **19**(3): 194 – 208. DOI: https://doi.org/10.1016/j.jisa.2014.03.002

Shiaeles S.N., Katos V., Karakos A.S. & Papadopoulos B.K. (2012). Real time DDoS detection using fuzzy estimators. *Computers and Security* **31**(6): 782 – 790. DOI: https://doi.org/10.1016/j.cose.2012.06.002

Silva S.S., Silva R.M., Pinto R.C. & Salles R.M. (2013). Botnets: a survey. *Computer Networks* **52**(2): 378 – 403. DOI: https://doi.org/10.1016/j.comnet.2012.07.021

Srihari V. & Anitha R. (2014). DDoS detection system using wavelet features and semi-supervised learning. *International Symposium on Security in Computing and Communications*, Delhi, India, pp. 291 – 303. DOI: https://doi.org/10.1007/978-3-662-44966-0_28

Tavallaee M., Bagheri E., Lu W. & Ghorbani A.A. (2009). A detailed analysis of the KDD CUP 99 dataset. *Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications* Ottawa, Canada, 8 – 10 July, pp. 53 – 58. DOI: https://doi.org/10.1109/CISDA.2009.5356528

Turpin A. & Scholer F. (2006). User performance *versus* precision measures for simple search tasks. *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, Seattle, USA, 6 – 11 August, pp. 11 – 18. DOI: https://doi.org/10.1145/1148170.1148176

Xue B., Zhang M. & Browne W.N. (2013). Particle swarm optimization for feature selection in classification: a multi-objective approach. *IEEE Transactions on Cybernetics* **43**(6): 1656 – 1671. DOI: https://doi.org/10.1109/TSMCB.2012.2227469