Proceedings of 2020 IEEE 10th International Conference on Intelligent Systems

Log Files Analysis For Network Intrusion Detection

Andre Brandao and Petia Georgieva
Department of Electronics, Telecomunications and Informatics / IEETA
University of Aveiro, Aveiro 3810-193, Portugal
Emails: {andrebrandao, petia}@ua.pt

***Abstract*—Information security remains an unsolved challenge for organizations, because every day brings new and sophisticated cyber-attacks. The implementation of network security operation systems is a new trend in the companies, to permanently monitor the logs and, in case of any anomalous traffic, to detect and hopefully prevent any security incident. These systems generate a huge amount of logs per second to be handled, leading to the need of automated ways to identify the cyber-attacks.**

In this paper we propose an effective Log-based Intrusion Detection System (LIDS), to predict an attack or not, based on carefully selected features. The logs from various sources are aggregated into one dashboard and the most discriminative features are first determined. For the attack prediction, a few machine learning techniques were comparatively tested, with the Decision tree being the winner. The proposed system is illustrated with the largest publicly available labelled log file dataset *KDD Cup 1999*.

***Keywords*—Intrusion detection; Network logs; machine learning classifiers**

I. Introduction

The number of automatic cyber-attacks has sharply increased, in the last years. As a consequence information security remains an unsolved challenge for organizations, and impacts negatively companies and people. The traditional defence is no longer effective in the new scenario of ever growing attacks and rapidly changing patterns attacks. Therefore, today more than ever, it is needed to protect the data information and network infrastructure, using other security devices, such as anti-virus servers, anti-spam devices, intrusion prevention systems (IPS), authentication servers, application firewalls, etc. All these mechanisms generate enormous quantity of logs in real time, making almost impossible for security engineers to inspect all of them, causing many undetected attacks. As the threats to the security increase, there is a need for a flexible and dynamic approach, that can react faster to security incidents. It is necessary to reduce the time for detection and analysis, and to tackle effective counter measures to any possible attack.

The classical intrusion detection systems (IDS) generate a high number of events and many false positives, making it difficult for humans to determine the actual attack. More recent IDSs focus on anomaly-based and misuse-based detection techniques, being the last one favoured in commercial products given their high accuracy. However, anomaly-based systems are more suitable to detect new forms of attacks.

In this paper we propose a Log-based Intrusion Detection System (LIDS) to predict if the network log is an attack or not. Log Analysis For Intrusion Detection is the process used to detect attacks on a specific environment using log files as the primary source of information.

II. Related Work

In order to avoid network attacks, the companies normally install several security devices such as Firewalls, IPS, anti-virus systems, etc., which are monitored by the Security Operation Center (SOC). These devices generate lots of logs. Additionally, there must be added the logs from Internet browsing, mail system, database, user authentication, Wireless Access, etc. On top of it, the logs from the networking devices have different structures, because they use different protocols or they are from different vendors (Cisco, Fortinet, SNMP, NetFlow, Linux or Windows platforms). All these problems leveraged the development of complex feature engineering approaches, for each individual client, such as log standardization, prioritization and normalization (building word dictionaries, identify keywords of event description, removing weak words, etc.). As a consequence, the commercial SOCs become very complex, with expensive and time-consuming maintenance. While larger businesses such as banks, insurances companies, service providers, etc. can afford such costly SOCs, small or medium companies are in a less favourable situation.

In order to address these issues some previous works focused into exploring the use of machine learning techniques with security logs. The ones that provided numerical results for the performance of their approaches are listed in Table V. Most of these works are based on the data from the KDD Cup 99 competition, which is the only sufficiently large and publicly available labelled dataset based on log files.

Portnoy and co-authors, [1] applied a variant of single-linkage clustering to the KDD Cup 99 with the assumption that the bigger clusters contain the normal data and the smaller ones the attacks. Based on unbalanced training set of 1–1.5% attack and 98.5 – 99% normal instances, they have achieved 65.7% detection rate. The ROC curve was used to visualize the trade-off between true positive (TP) and false positive (FP) rates.

Latter on, [2] applied both supervised and unsupervised algorithms taking into account all features as inputs and achieved the best prediction rates with the C4.5 algorithm.

The authors of [3] used private logs from a number of services to detect infected machines and suspicious activities applying PCA (Principal Component Analysis) and k-means.

978-1-7281-5456-5/20/$31.00 ©2020 IEEE

Different approach was followed in [4], where instead of using the logs to detect an attack, the alerts generated by sensors (IDSs such as for example Snort) were considered. These events are used to create meta-alerts that are then used to feed the classification models (Support Vector Machine, Bayesian Network and C4.5.

In [5], logs from the Ericsson Radio Base Station(RBS) were used, applying Self-organizing Map (SOM) and Density-based spatial clustering of applications with noise(DBSCAN). This dataset is not publicly available and according to the paper the labels are also not sufficiently reliable.

The work in [6] is based on the publicly available unlabelled Honeynet project dataset. After data normalization through the linearised Intrusion Detection Message Exchange Format (IDMEF) structure, the authors achieved 98.2% accuracy using a Decision Tree classifier.

Deep Neural Network (DNN) with 3 fully connected hidden layers and dropout layers was applied in [7] to KDD Cup 99 data. The network was trained with Back-prop, ReLU functions in the hidden layers and sigmoid function in the last layer to predict attack or benign traffic.

Publicly available data from Twitter streaming API was considered recently in [8] to detect cyber threats. Data was labelled manually, corresponding to the activity of selected subset of accounts over four months. End-to-end threat intelligence tool was proposed using a sequence of two DNNs: a binary classifier based on a Convolutional Neural Network (CNN) architecture inspired by Natural Language Processing (NLP) and a Bidirectional Long Short-Term Memory(BiLSTM)network borrowed from Named Entity Recognition (NER). Data-driven approach was proposed also in [9] based on reinforcement learning for Intrusion detection system with log files. A common pain point of both deep and reinforcement learning is the need of a big amount of labelled training data to generalize, which is a labour intensive problem. Data visualization methods, as discussed in [10], are also intensively explored to facilitate the root cause analyses of network intrusion detection.

III. Data

Since 1997 the ACM Special Interest Group on Knowledge Discovery and Data Mining have organised regularly the Data Mining and Knowledge Discovery competitions (KDD Cup). In 1999, in conjunction with the KDD'99 conference, they launched the KDD Cup 1999 on Computer network intrusion detection (https://www.kdd.org/kdd-cup/view/kdd-cup-1999). The goal of the competition was to build a classifier to separate legitimate and illegitimate computer network connections based on log files.

The DARPA Intrusion Detection Evaluation 1998 dataset was acquired by Lincoln Labs over nine weeks in a local-area network (LAN) simulating a typical U.S. Air Force LAN. They operated the LAN as if it were a true Air Force environment, but peppered it with multiple attacks, [11]. The 1999 KDD intrusion detection contest uses a version of this dataset, processed into about five million connection records for training and around two million connection records of test data (available at: https://tinyurl.com/y2l64r8y). A connection is a sequence of TCP packets with well-defined start and end times, where data is transferred between source and target IP addresses under an established protocol. Each connection is labelled as either normal, or as an attack, with exactly one specific attack type. Each connection record consists of about 100 bytes. The logs come from multiple services (*e.g.* http, smtp, telnet, etc.) but the data was merged into one structure of 41 features given in Table VI.

The features are derived in several categories [11]. *Basic features* such as duration, service, protocol type, number of data bytes transferred.

The "same host" features examine only connections in the last 2 sec. with the same destination host as the current connection, and calculate statistics related to protocol behaviour, service, etc. The "same service" features examine only connections in the last 2 sec. with the same service as the current connection. "Same host" and "same service" features are together called *Time-based traffic features of the connection records*.

Some probing attacks scan the hosts (or ports) using a much larger time interval than two seconds, for example once per minute. Therefore, connection records were also sorted by destination host, and features were constructed using a window of 100 connections to the same host instead of a time window. This yields a set of so-called *Host-based traffic features*.

Stolfo et al. [11], added the so called *Content features*, that look for suspicious behaviour in the data portions, such as the number of failed login attempts.

In this work we make use of KDD-CUP99 Data based on several arguments. First, it has been used over the years as a benchmark problem for log files analysis for network intrusion detection, as discussed in section refSOTA. Second, it is the only publicly available labelled dataset based on log files. The third reason is related to the fact that test and training data may come from different probability distributions. The datasets contain a total of 22 training attack types, with an additional 14 types in the test data only . This makes the dataset very realistic. Since some intrusion experts believe that most novel attacks are variants of known attacks, if the detection mechanism generalize well on test data, it will hopefully catch novel variants in real implementation.

We have extracted 10% of KDD-CUP99 with equal distribution among normal records and 22 attack types of records (see Fig. 1).

IV. Feature Selection

Selecting relevant features in network intrusion detection is an open research question. Finding the important features not only speed up the data manipulation but holds the promise to improve the detection rate.

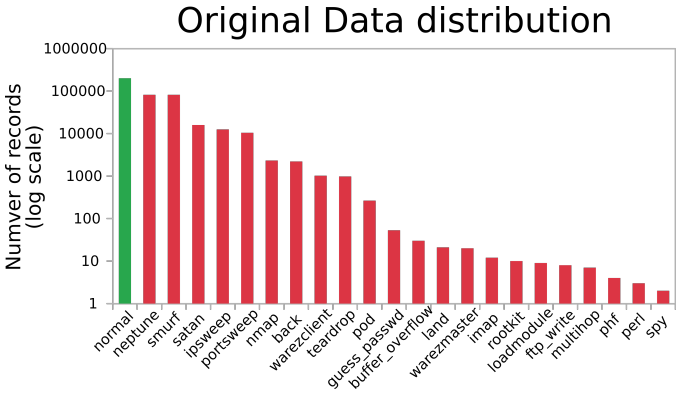

Fig. 1: KDD Cup 1999 Data (red attack), green(normal).

Most of the references discussed in section II use some kind of feature filtering as a preprocessing step without explicit feature selection. In [12], a feature selection based on the rough set theory is proposed, however without applying it for intrusion detection.

Here we propose to apply two feature selection approaches, namely Optimal Feature Selection (OFS) and Factor Analysis (FA).

A. Optimal Feature Selection

OFS is a deterministic greedy algorithm that takes the locally optimal choice at each stage, [13]. OFS can be implemented in two ways: forward selection or backward elimination.

Forward Selection. Build data models based on individual features. Choose the best *k1* features. Make all two by two combinations of the *k1* selected features and build new data models. Choose the next group of best *k2* features and make all three by three combinations of them. Repeat the process until no more improvement in the model performance. The choice of subsequent feature subset combinations may vary in order to reflect the experience gained during the extraction process and the application in hand.

Backward Elimination. The model is trained on the complete set of features and weights are assigned to each of them. The features with the smallest weights are then pruned from the set. This process is repeated until the model performance deteriorates below a certain threshold.

B. Factor Analysis

Factor Analysis (FA) is an explorative method similar to Principal Components Analysis (PCA), [14]. Both methods try to reduce the dimensionality of the dataset based on the concept of common variance. Common variance is the amount of variance that is shared among a set of features. Features that are highly correlated will share a lot of variance. PCA assumes that the total variance is build on all common variances, while the FA assumes that total variance can be partitioned into common and unique variance. Unique variance is any portion of variance that is not common. Thus, in the FA framework, the total variance is sum of the common variance and unique variance.

FA identifies the latent (unobserved) factors which represent the hidden organization of the observable measures. Factor scores (aka factor loadings) indicate how each latent factor is associated with the observable variables used in the analysis. If the goal is to simply reduce the variable list down into a linear combination of smaller number of components then PCA is the way to go. However, if the goal is to determine what are the features that describe a latent factor then the FA is more appropriate.

TABLE I: Factor Analysis loadings for *varimax* rotated matrix of 4-factor model(in bold for absolute values > 0.75).

	F1	F2	F3	F4
service	**0.91**	-0.13	0.07	0.02
protocol_type	**0.78**	0.10	-0.44	0.01
flag	**0.77**	0.04	0.44	0.23
count	**-0.79**	0.01	-0.01	-0.25
logged_in	-0.14	**0.88**	0.03	0.07
DST_host_count	-0.02	**0.88**	-0.11	0.09
DST_host_diff_srv_rate	-0.23	-0.11	0.75	-0.13
DST_host_same_srv_port_rate	-0.34	-0.02	-0.72	-0.13
DST_host_serror_rate	0.13	0.00	-0.27	0.73

V. Network Intrusion Prediction

A. KDD Cup99 feature selection

1) Optimal Feature Selection: The OFS (implemented in the statistical software RapidMiner) ranked the features in three categories: green (most important), yellow (neutral) and red (least important). 23 features were ranked as green: *protocol_type, service, src_bytes, DST_bytes, flag, hot, num_failed_logins, logged_in, hot_login, count, serror_rate, same_srv_rate, diff_srv_rate, srv_count, srv_serror_rate, DST_host_count, DST_host_srv_count, DST_host_same_srv_rate, DST_host_diff_srv_rate, DST_host_same_src_port_rate, DST_host_srv_serror_rate, DST_host_srv_diff_host_rate, DST_host_serror_rate.*

2) Factor Analysis: In Table I are presented the FA results of a 4-factor model performed with software R libraries. In the first column are represented the dominant features that identify the four hidden factors. The other columns are the factor loadings on these factors. Based on FA, a reduced feature set was constructed accounting for features with factor loadings with absolute values greater that 0.75:

Reduced feature vector= [*service*, *protocol_type*, *flag*, *count*, *logged_in*, *DST_host_count*]. Note that, according to both approaches, OFS and FA, the most important features are the categorical *service*, *protocol_type*, and *flag*. Their histograms, shown in Fig. 2, Fig. 3 and Fig. 4, confirm that their class distributions are different, though some overlapping is also observed.

Most of the other selected features (*DST_xxx*) belong to the so-called host-based traffic features. These are connection records

sorted by the destination host. They are constructed using a window of 100 connections to the same host instead of a time window. The reason for adding these features, according to [11], is that some probing attacks scan the hosts (or ports) using a much larger time interval than two seconds, for example once per minute. Hence, their discrimination capacity is reflected by the feature selection mechanisms.

One hot encoding was applied to the selected categorical features.

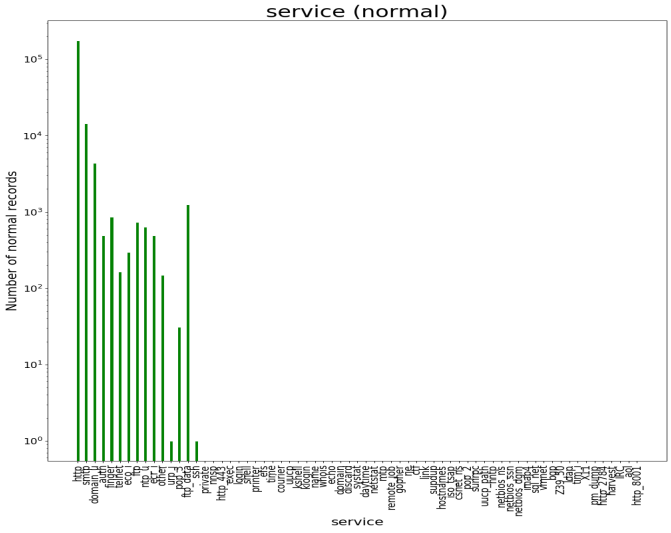

Fig. 2: Distribution of feature *service* - normal

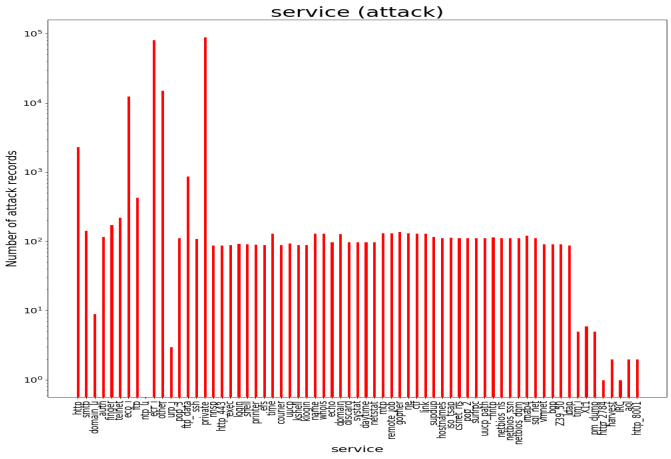

Fig. 3: Distribution of feature *service* -attack

B. Prediction Models

Typical classification models were selected - Decision Tree (DT), K-Nearest Neighbours (kNN) and Neural Network (NN). This choice reflects the focus of the present study in finding the important network intrusion descriptors, less dependent of the classifier characteristics. Data were split into training and validation subset (80%) and testing subset (20%).

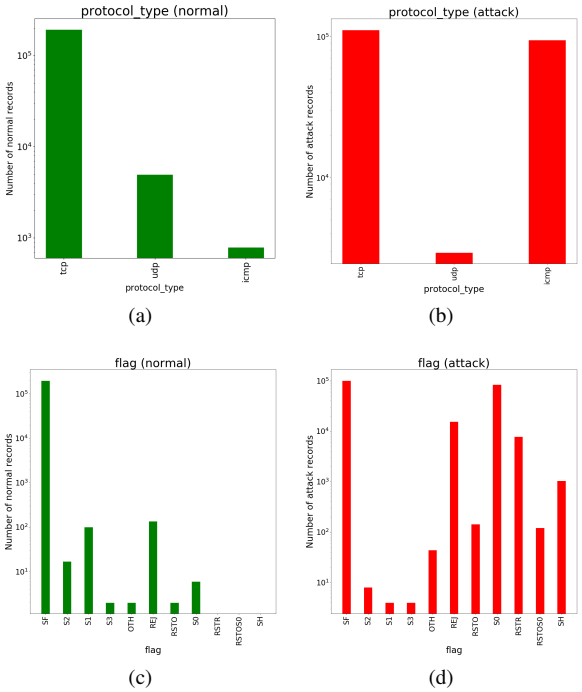

Fig. 4: Distribution of categorical features *protocol* and *flag*

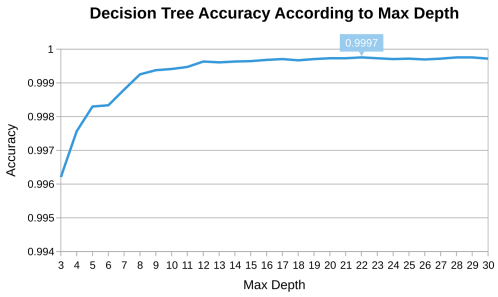

Fig. 5: DT model (variation of maximum depth)

The optimal structure and hyper-parameters of the classifiers were chosen after a grid search with data dimension of 23 features selected by the OFS.

1) Decision Tree: The DT model was evaluated varying the maximum depth of the tree in the range [3, 30]. The optimal DT structure was found with maximum depth of 22 (Fig. 5).

2) K-Nearest Neighbours: The kNN classifier was optimized varying the number of neighbours k in the range [3, 15], with the optimal value equal to 3 (Table II). As distance measure for computing the k - nearest neighbours was chosen cosine similarity. Cosine similarity is a measure of similarity between two non-zero vectors expressed by the cosine of the angle between them.

3) Neural Network: A fully connected NN with one hidden layer (HL) and ReLu activation functions was chosen. The learning rate was picked from the range [0.001, 0.1, 05]. The number of HL units was optimized in the range [3, 5, 10, 20, 50, 80]. Although the best performance was achieved for HL

TABLE II: kNN model (variation of number of neighbours)

	Number of neighbours k				
	3	5	7	11	15
Accuracy	**99.90**	99.87	99.86	99.84	99.83

TABLE III: NN model (variation of HL size and learning rate)

		Learning Rate		
		0.001	0.1	0.5
Hidden Layer Size	3	98.97	98.86	98.73
	5	99.63	99.45	98.86
	10	99.65	99.36	98.87
	20	99.62	99.42	98.92
	50	99.66	99.34	98.80
	80	**99.89**	99.08	98.70

with 80 units (Table III), simpler model with 5 HL units was chosen as the optimal structure due to the negligible difference in performance.

C. Model Evaluation

In order to assure a consistent comparison between the models, the classifiers with the same hyper-parameters, as determined above, were trained with the features selected by OFS or FA. The performance of the models on test data (in terms of confusion matrix and test accuracy) between the classifiers are summarized in the Table IV. The results suggest that the three classifiers have similar performance with the DT slightly better than the other classifiers. These results are in accordance to the reviewed literature, where DT-based classifiers are often the preferred model. The achieved state of the art performance of 99.97 % is due to the careful selection of discriminative features.

TABLE IV: Test performance of 3 classifiers (DT, kNN, KNN) with feature selection methods: Optimal Feature Selection (OFS), Factor Analysis (FA)

	TP	FP	TN	FN	Acc (%)
OFS&DT	39922	17	41809	10	99.97
OFS&kNN	39828	39	41848	43	99.90
OFS&NN	39835	196	41620	107	99.63
FA&DT	39901	28	41798	31	99.93
FA&kNN	39806	39	41848	65	99.87
FA&NN	39795	246	41570	147	98.52

VI. Conclusions

Selecting relevant feature is an important problem in learning systems. KDD Cup99 dataset was used for benchmarking intrusion detection problem based on network traffic logs and a suitable combination of features. The elimination of the insignificant features simplified the problem and did not hurt the detection rate. Our findings show that if the most relevant features are determined, the choice of the classifier is less critical. It does become more important to focus into data mining to attain best results.

With regards to future work, there is most certainly room for improvement. Alternative avenues could be explored either in the form of new algorithms or in the form of new features. In many attacks nowadays the attack vector used is a phishing email, and to detect the phishing emails it's necessary to make a higher level analysis, content and context analysis, using for example natural language processing. As such, a complete Intrusion Detection System would benefit from being composed of different models for different tasks, that combined will be more effective for detecting various forms of attack.

Acknowledgment

This work was funded by National Funds through the FCT - Foundation for Science and Technology, in the context of the project UID/CEC/00127/2019.

References

[1] L. Portnoy, "Intrusion detection with unlabeled data using clustering," Ph.D. dissertation, Columbia University, 2000.

[2] P. Laskov, P. Düssel, C. Schäfer, and K. Rieck, "Learning intrusion detection: supervised or unsupervised?" in *International Conference on Image Analysis and Processing*. Springer, 2005, pp. 50–57.

[3] T.-F. Yen, A. Oprea, K. Onarlioglu, T. Leetham, W. Robertson, A. Juels, and E. Kirda, "Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks," in *Proceedings of the 29th Annual Computer Security Applications Conference*. ACM, 2013, pp. 199–208.

[4] K. Stroeh, E. R. M. Madeira, and S. K. Goldenstein, "An approach to the correlation of security events based on machine learning techniques," *Journal of Internet Services and Applications*, vol. 4, no. 1, p. 7, 2013.

[5] W. Li, "Automatic log analysis using machine learning: awesome automatic log analysis version 2.0," 2013.

[6] E. G. Vasquez Villano, "Classification of logs using machine learning technique," Master's thesis, NTNU, 2018.

[7] K. R. Vigneswaran, R. Vinayakumar, K. Soman, and P. Poornachandran, "Evaluating shallow and deep neural networks for network intrusion detection systems in cyber security," in *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE, 2018, pp. 1–6.

[8] N. Dionísio, F. Alves, P. M. Ferreira, and A. Bessani, "Cyberthreat detection from twitter using deep neural networks," *arXiv preprint arXiv:1904.01127*, 2019.

[9] B. Deokar and A. Hazarnis, "Intrusion detection system using log files and reinforcement learning," *International Journal of Computer Applications*, vol. 45, no. 19, pp. 28–35, 2012.

[10] V. Bulavas, "Investigation of network intrusion detection using data visualization methods," in *2018 59th International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS)*. IEEE, 2018, pp. 1–6.

[11] J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan, "Cost-based modeling and evaluation for data mining with application to fraud and intrusion detection," *Results from the JAM Project by Salvatore*, pp. 1–15, 2000.

[12] V. Rampure and A. Tiwari, "A rough set based feature selection on kdd cup 99 data set," *International journal of database theory and application*, vol. 8, no. 1, pp. 149–156, 2015.

[13] L. Bozhkov and P. Georgieva, "Brain neural data analysis with feature space defined by descriptive statistics," in *Iberian Conference on Pattern Recognition and Image Analysis*. Springer, 2015, pp. 415–422.

[14] L. R. Tucker and R. C. MacCallum, "Exploratory factor analysis," *Unpublished manuscript, Ohio State University, Columbus*, 1997.

TABLE V: Related work (data, methods, performance)

Reference	Data (Labelled (Yes/No)	Methods	Best Performance
2000 [1]	KDD Cup 99 (Yes)	Single-Linkage Clustering	Detection Rate: 65.7%
2005 [2]	KDD Cup 99 (Yes)	C4.5 algorithm	TP: 95% FP: 1%
2013 [5]	RBS (Radio Base Station) from Ericsson (No)	K-means, DBSCAN, SOFM/SOM	F-score: 95% (SOFM-DBSCAN)
2018 [6]	Honeynet Project (No)	Decision Tree	Acc: 98.2%
2018 [7]	KDD Cup 99 (Yes)	Deep Learning and others	Acc: 92.9% (DNN)
2019 [8]	Twitter (Yes)	DNN	F1-score: 92%
this work	KDD Cup 99 (Yes)	Feature selection & Decision Tree	**Acc:99.97%, FP=0.05%**

TABLE VI: KDD Cup 1999 Data Features

Feature	Description	Type
Basic features		
duration	connection length (measured in sec.)	cont.
protocol_type	protocol type (udp, tcp, etc.)	categorical
service	service on the destination (telnet, http, etc.)	categorical
src_bytes	quantity of data bytes from source to destination	cont.
DST_bytes	quantity of data bytes from destination to source	cont.
flag	connection status (normal or error)	categorical
land	1 if connection is from/to the same host/port; 0 otherwise	binary
wrong_fragment / urgent	# of ”wrong” fragments / # of urgent packets	cont.
Content features		
hot	# of ”hot” indicators	cont.
num_failed_logins	# of failed login attempts	cont.
logged_in	successfully login (1); otherwise (0)	binary
num_compromised	# of ”compromised” conditions	cont.
root_shell	root shell is obtained (1); otherwise (0)	binary
su_attempted	1 if ”su root” command attempted; 0 otherwise	binary
num_root	# of ”root” accesses	cont.
num_file_creations	# of file creation operations	cont.
num_shells	# of shell prompts	cont.
num_access_files	# of operations on access control files	cont.
num outbound cmds	# of outbound commands in an ftp session	cont.
hot_login	1 if the login belongs to the ”hot” list; 0 otherwise	binary
guest_login	1 if the login is a ”guest” login; 0 otherwise	binary
Time-based traffic features of the connection records (2 sec. window)		
count	# of connections to the same host as the current connections in the last 2 sec.	cont.
serror_rate	% of connections that have ”SYN” errors	cont.
rerror_rate	% of connections that have ”REJ” errors	cont.
same_srv_rate	% of connections to the same service	cont.
diff_srv_rate	% of connections to different services	cont.
srv_count	# of connections to the same service as the current connection in the last 2 sec.	cont.
srv_serror_rate	% of connections that have ”SYN” errors	cont.
srv_rerror_rate	% of connections that have ”REJ” errors	cont.
srv_diff_host_rate	% of connections to different hosts	cont.
Host-based traffic features		
DST_host_count	# of connections to the same destination host as the current connection	cont.
DST_host_srv_count	# of connections to the same destination service as the current connection	cont.
DST_host_same_srv_rate	% of connections to the same destination service	cont.
DST_host_diff_srv_rate	% of connections to different destination services	cont.
DST_host_same_src_port_rate	% of connections to the same source port	cont.
DST_host_srv_diff_host_rate	% of connections to different hosts	cont.
DST_host_serror_rate	% of connections that have ”SYN” errors	cont.
DST_host_srv_serror_rate	% of connections that have ”SYN” errors	cont.
DST_host_rerror_rate	% of connections that have ”REJ” errors	cont.
DST_host_srv_rerror_rate	% of connections that have ”REJ” errors	cont.