ISSN 2601-8683 (Print)
ISSN 2601-8675 (Online)
European Journal of
Formal Sciences and Engineering
July - December 2022
Volume 5, Issue 2

Detection of Anomalies in the Computer Network Behaviour

Danijela Protić

MSc, Centre of Applied Mathematics and Electronics, Belgrade, Serbia

Miomir Stanković

PhD, Mathematical Institute, Serbian Academy of Science and Arts, Belgrade, Serbia

Abstract

The goal of anomaly-based intrusion detection is to build a system which monitors computer network behaviour and generates alerts if either a known attack or an anomaly is detected. Anomaly-based intrusion detection system detects intrusions based on a reference model which identifies normal behaviour of the computer network and flags an anomaly. Basic challenges in anomaly-based detection are difficulties to identify a 'normal' network behaviour and complexity of the dataset needed to train the intrusion detection system. Supervised machine learning can be used to train the binary classifiers in order to recognize the notion of normality. In this paper we present an algorithm for feature selection and instances normalization which reduces the Kyoto 2006+ dataset in order to increase accuracy and decrease time for training, testing and validating intrusion detection systems based on five models: k-Nearest Neighbour (k-NN), weighted k-NN (wk-NN), Support Vector Machine (SVM), Decision Tree, and Feedforward Neural Network (FNN).

Keywords: intrusion detection, Kyoto 2006+, k-NN, wk-NN, SVM, decision tree, FNN

Introduction

Intrusion detection systems (IDSs) protect computer networks from malicious activities which compromise network security and affect the confidentiality, integrity and availability of the data. IDSs can be grouped into the signature-based, anomaly-based, and hybrid (Ganpathy et al., 2015, 44-50).The basic idea of signature-based detection is to represent an attack in the form of pattern in such a way that any known attack and its variation can be detected. The main disadvantage of this approach is difficulty for detecting unknown attacks. Anomaly-based IDS detects changes in the network behaviour. The goal of anomaly-based detection is to build a statistical model

that describes the normal behaviour of the computer network and then looks for activities which differ from the created model. It detects both intrusions and/or misuse, and classifies them as either 'normal' or 'anomaly'. The biggest challenge in anomaly-based detection is to identify what is considered normal network behaviour. Machine learning (ML) models can be trained as binary classifiers in order to recognize the notion of normality. In a supervised ML, the data have to be collected over a period of time to create a model of normal behaviour of users, hosts, and networks.

A number of records needed for training the complex computer networks can be large, which makes evaluation of the IDS computationally expensive since the processing time and memory usage rise with a size of the dataset. In anomaly-based detection some of the recorded data can be discarded to decrease time needed for training and increase accuracy of IDSs. This paper presents one pre-processing technique applied to reduce the size of the Kyoto 2006+ dataset. The proposed algorithm cuts off all categorical features, features which are intended for further analyses of the evaluated models, and features containing instances which cannot be normalized into the range [-1, 1], excluding the feature 'Label' that flags either normal network behaviour or an anomaly (Ramasamy & Rani, 2018, 1060-1067). After the pre-processing, of 24 features describing each instance nine features left. In this paper we present results on the accuracy and computation time testing for five supervised learning models: k-Nearest Neighbour (k-NN), weighted k-NN (wk-NN), Support Vector Machine (SVM), Decision Tree and Feedforward Neural Network (FNN).

1. The Kyoto 2006+ Dataset

The Kyoto 2006+ dataset is built on the three years (November 2006 to August 2009) of the real network traffic data, collected on five different computer networks inside and outside the Kyoto University. The data set is designed to provide evaluation of the network-based intrusion detection systems (NIDS). It consists of 14 statistical features derived from the KDD Cup '99 dataset (1999) and 10 additional features which can be used for evaluation and further analyses of NIDS (Protic, 2018, 580-595). The Kyoto 2006+ dataset is captured using honeypots, darknet sensors, e-mail servers, web crawlers and other intrusion detection systems (Sing, 2014, 31-35). During the observation period, more than 50 million sessions of normal traffic, over 43 million sessions of known attacks and almost 426 thousand sessions of unknown attack were recorded (Song et al, 2011). Of the 41 features derived from the KDD Cup '99 dataset, authors discarded redundant data and content features which are not suitable for NIDS (See Table 1).

Table 1 First 14 features from the KDD Cup '99 dataset

No	Feature	Description
1	Duration	The length of the connection (seconds).
2	Service	The connection's server type (dns, ssh, other).

3	Source bytes	The number of data bytes sent by the source IP address.
4	Destination bytes	The number of data bytes sent by the destination IP address.
5	Count	The number of connections whose source IP address and destination IP address are the same to those of the current connection in the past two seconds.
6	Same_srv_rate	% of connections to the same service in the Count feature.
7	Serror_rate	% of connections that have 'SYN' errors in Count feature.
8	Srv_serror_rate	% of connections that have 'SYN' errors in Srv_count (% of connections whose service type is the same to that of the current connections in the past two seconds) feature.
9	Dst_host_count	Among the past 100 connections whose destination IP address is the same to that of the current connection, the number of connections whose source IP address is also the same to that of the current connection.
10	Dst_host_srv_count	Among the past 100 connections whose destination IP address is the same to that of the current connection, the number of connections whose service type is also the same to that of the current connection.
11	Dst_host_same_src_port_rate	% of connections whose source port is the same to that of the current connection in Dst_host_count feature.
12	Dst_host_serror_rate	% of connections that have 'SYN' errors in Dst_host_count feature.
13	Dst_host_srv_serror_rate	% of connections that have 'SYN' errors in Dst_host_srv_count feature.
14	Flag	The state of the connection at the time of connection was written (tcp, udp).

(Source: Song et al., 2011)

In this way, the 14 features left consisted of one categorical feature 'Flag' and 13 continuous features. Moreover, authors extracted 10 additional features: 'Label' which indicated normal traffic or attacks, four features describing source and destination addresses and port numbers, two features describing start time and duration of the session, and three features for IDS, malware and Ashula detection (See Table 2).

Table 2. Additional features

No	Feature	Description
1	IDS_detection	Reflects if IDS triggered an alert for the connection; '0' means any alerts were not triggered and an arabic numeral means the different kind of alerts. Parenthesis indicates the number of the same alert.
2	Malware_detection	Indicates if malware, also known as malicious software, was observed at the connection; '0' means no malware was observed, and string indicates the corresponding malware observed at the connection. Parenthesis indicates the number of the same malware.
3	Ashula_detection	Means if shellcodes and exploit codes were used in the connection; '0' means no shellcode nor exploit code were observed, and an arabic numeral means the different kinds of the shellcodes or exploit codes. Parenthesis indicates the number of the same shellcode or exploit code
4	Label	Indicates whether the session was attack or not; '1' means normal. '-1' means known attack was observed in the session, and '-2' means unknown attack was observed in the session.
5	Source_IP_Address	Means source IP address used in the session. The original IP address on IPv4 was sanitized to one of the Unique Local IPv6 Unicast Addresses. Also, the same private IP addresses are only valid in the same month; if two private IP addresses are the same within the same month, it means their IP addresses on IPv4 were also the same, otherwise are different.
6	Source_Port_Number	Indicates the source port number used in the session.
7	Destination_IP_Address	It was also sanitized.
8	Destination_Port_Number	Indicates the destination port number used in the session.
9	Start_Time	Indicates when the session was started.
10	Duration	Indicates how long the session was being established.

(Source: Song et al., 2011)

2. Feature Selection

One of the major issues associated with the Kyoto 2006+ dataset is its size. Features selection reduces the data dimensionality by determining whether a feature is relevant or not for evaluation of the NIDS model. Using effective features in designing classifiers not only reduce the dataset but also improve performances of the classifier (Jayakumar, Revathi & Karpagam, 2015, 728-734).

In this paper we present two-step pre-processing algorithm for feature selection given as follows:

Step 1: Discard all categorical features and all features which are intended for further analyses, excluding feature 'Label'.

Step 2: Cut features containing instances which cannot be normalized into the range [-1, 1].

Of the 24 features of the Kyoto 2006+ dataset, 17 features are discarded after the first algorithm step. Nine features (5-13) left after the pre-processing is done. These features are normalized to fall into the range [-1, 1] by applying the hyperbolic tangent function given with Eq. (1):

$$\tanh(n) = \frac{2}{1+e^{-2n}} - 1 \qquad (1)$$

In this way the values of instances are scaled so that the effect of one feature cannot dominate the others. Moreover, normalized instances speed up FNN. Network training is more efficient when this pre-processing is performed on inputs. If the inputs are greater than 3 ($e^{-3} \approx 0.05$) sigmoid functions, which are used in the hidden network layer, become essentially saturated. If this happens at the beginning of the training process the gradients will be very small and the network training will be slow.

3. Machine Learning Models

Supervised ML algorithms use the known dataset to evaluate a model that generates prediction of unknown data. Assume that all data points belong either to the class 'normal' or class 'anomaly' (binary classification). Than each training data point x_i, from a vector of d-dimensional feature space, can be labelled by y_i as follows (see Eq. (2)):

$$y_i = \begin{cases} y_n, & x_i \in class_{normal} \\ y_a, & x_i \in class_{anomaly} \end{cases} \qquad (2)$$

The training dataset is denoted as follows $\{(x_i, y_i),\ i = 1, \cdots, k\}$.

This paper presents five models used for binary classification: k-NN, wk-NN (Hechenbichler & Schliep, 2004), SVM (Burgess, 1998, 283-298), Decision Tree (Sebastiani, 2002, 13) and FNN (Protic & Milosavljevic, 2006, 643-646).

3.1 k-Nearest Neighbour

k-NN is nonparametric method where new observation is placed into the class of observation from the learning set (Hechenbichler & Schliep, 2004). In this paper we present results on k-NN based prediction of unknown instances which finds the largest similarity of instances based on the Euclidean distance measure (Eq. (3)):

$$d(x_i, y_i) = \sqrt{\sum_{s=1}^{p} (x_{is} - y_{is})^2} \qquad (3)$$

3.2 Weighted k-Nearest Neighbour

The main idea of wk-NN is to extend the k-NN method in so that the observations within the learning set, which are particularly close to the new observation, should get a higher weight in the decision than such neighbours which are more distant one from that observation (Hechenbichler & Schliep, 2004). To reach this aim the distances have to be transformed into the similarity measures (Eq. (4)), which can be used as weights (Eq. (5)):

$$dist = \sqrt{\sum_{i=1}^{p} (x_i - y_i)} \qquad (4)$$

$$w = \frac{1}{dist^2} \qquad (5)$$

3.3 Support Vector Machine

The goal of SVM algorithm is to find a hyperplane that distinctly classifies the data points into various classes (Burgess, 1998, 283-298). To separate instances into the classes 'normal' or 'anomaly' the algorithm finds a linear hyperplane that has the maximum distance ρ=2/||w|| between data points of both classes. For the training set

$\mathbf{x}(x_i, y_i), x_i \in R^d, y_i \in \{-1,1\}$, the discriminant function takes the form (Eq. (6)):

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b = \sum_i w_i x_i + b \qquad (6)$$

where $\boldsymbol{w}$ is normal vector to the hyperplane that can be determined as follows (Eq. (7)):

$$\begin{aligned} & x_i \mathbf{w} + b \geq +1, && y_i = +1 \\ & x_i \mathbf{w} + b \leq -1 && y_i = -1 \\ & y_i (x_i \mathbf{w} + b) - 1 \geq 0, && \forall i,\ 1 \leq i \leq n \end{aligned} \qquad (7)$$

The idea is to find min||w|| which maximizes the distance ρ.

3.4 Decision Tree

Decision Tree prediction is based on the principle of recursive partitioning by monitoring decisions from the root to the last node (Sebastiani, 2002, 13). It is one of the graph-like algorithms which use branching methods to illustrate every possible outcome of the decisions, where nodes represent features, links represent decision rules and leafs represent the outcomes. In the experiments we applied the Iterative Dichotomy 3 algorithm (ID3) (Colin, 1996, 107-110) which calculates entropy and information gain to build a tree. Entropy is a measure which controls how the tree decides to split the data. If the target feature can take on k different values then entropy of S relative to this k-wise classification can be calculated as follows (Eq. (8)):

$$Entropy(S) = -\sum_{i=1}^{k} p_i \log_2(p_i) \quad (8)$$

where p_i is the proportion S belonging to class i. Information gain represents expected reduction in entropy based on the decrease in entropy after the dataset is split on the feature. The feature with highest information gain will split first. Information gain can be calculated with the formula (Eq. (9)):

$$Gain(S,A) = Entropy(S) - \sum_{v \in avlues(A)} \frac{|S_v|}{|S|} \cdot Entropy(S_v) \quad (9)$$

Gain(S,A) of a feature A relative to a collection of examples S provides information about the target function value, given the value of some other feature A (when A splits the set S into the subsets S_v).

3.5 Feedforward Neural Network

FNN is based on the back-propagation algorithm. The nonlinear transfer function of the FNN is given with Eq. (10):

$$y_i(\mathbf{w},\mathbf{W}) = F_i\left(\sum_{j=1}^{q} W_{ij} f_j\left(\sum_{l=1}^{m} w_{jl} x_l + w_{j0}\right) + W_{i0}\right) \quad (10)$$

where x_l are inputs, y_i are outputs, **w** and **W** are weight matrices, f_j and F_i denote transfer functions of hidden and output layers, *m* represents the number of inputs, *q* represents the number of outputs, and w_{j0} and W_{i0} denote biases (Protic & Milosavljevic, 2006, 643-646). The objective of FNN presented in this paper is to minimize output error in accordance with the Levenberg-Marquardt algorithm (Protic, 2015, 11-28). It should be pointed out that the FNN has to be large enough to improve network generalization and provide an adequate fit.

4. Results

In our experiments, performances of the models are measured based on accuracy (ACC), which represents the ratio of number of instances correctly classified to the total number of instances given with Eq. (11) (Ambedkar & Babu, 2015, 25-29):

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \quad (11)$$

where true positive (TP), true negative (TN), false positive (FP) and false negative (FN) denote detected network behaviour as follows:

TP – 'anomaly' is detected as 'anomaly',

TN – 'normal' is detected as 'normal',

FP – 'normal' is detected as 'anomaly' and

FN – 'anomaly' is detected as 'normal'.

In our previous work we have presented results based on normalized and not-normalized dataset of instances and four ML models, namely k-NN, wk-NN, SVM and Decision Tree (Protic & Stankovic, 2018, 43-48). Here we present results of the experiments conducted to the normalized instances and five models: k-NN, wk-NN, SVM, Decision Tree and FNN. Accuracy of the models and the corresponding computation time (sum of training, testing and validation time) are given in Table 3.

Table 3 Accuracy and computation time

No	Size of the dataset	Accuracy [%] Comp. time [s]	FNN	k-NN	wk-NN	SVM	Decision Tree
1	158572	Accuracy [%]	98.8	98.3	98.4	98.1	97.2
		Comp. time [s]	26	275.72	277.32	449.35	3.8452
2	129651	Accuracy [%]	97.67	91.8	91.8	98.3	97.3
		Comp. time [s]	20	175.84	173.32	254.3	3.3104
3	128740	Accuracy [%]	98.32	98.2	98.1	97.8	97.2
		Comp. time [s]	8	193.82	194.81	280.82	3.3033
4	136625	Accuracy [%]	99.21	99.3	99.4	99.1	98.3
		Comp. time [s]	20	194.83	194.23	217.32	8.3169
5	90129	Accuracy [%]	98.99	99.0	99.1	99.0	98.4
		Comp. time [s]	11	101.28	101.753	86.283	2.2308

6	93999	Accuracy [%]	98.12	96.5	96.5	98.0	97.5
		Comp. time [s]	7	109.25	108.77	111.83	2.2613
7	81807	Accuracy [%]	98.3	98.8	98.8	97.9	98.9
		Comp. time [s]	10	91.25	91.26	227.28	2..2615
8	57278	Accuracy [%]	99.14	99.36	99.3	99.2	99.3
		Comp. time [s]	2	42.704	43.235	33.754	1.743
9	58317	Accuracy [%]	98.97	99.1	99.2	99.1	98.9
		Comp. time [s]	3	31.714	31.738	34.234	1.7482
10	57278	Accuracy [%]	99.2	99.4	99.5	99.2	99.4
		Comp. time [s]	2	43.734	43.272	30.239	1.2901

The experiments are conducted on 10-days records from the Kyoto 2006+ dataset (991.395 instances in total). All models are trained so that out of the total number of randomly selected instances 70% are used for training, 15% for testing and 15% for validation of the models. Experiments are performed using Intel(R) Core(TM) i7-2620M CPU 2.70GHz processor with 16GB RAM Installed Memory.

Results show the highest accuracy (99.5%) of wk-NN model. The results also point to high accuracy for k-NN model (99.36%). Computation time for evaluating Decision Tree model is significantly shorter comparing to the computation time for other model's evaluation, except for the FNN. Number of parameters in the network structure with nine inputs, one hidden-layer and one output is large enough to provide an adequate fit. The highest accuracy of FNN (99.21%) is achieved when the network is trained with the largest subset (136.625 instances in total). As expected, the time period needed for network learning is longer than for the networks trained with the smaller datasets (20s). Time period of training, testing and validating the FNN is significantly shorter than k-NN, wk-NN and SVM, and fall into the range [2s, 26s]. The results also show that the SVM model has a lower accuracy and longer computation time comparing to the other models.

Conclusion

Anomaly-based intrusion detection systems recognize deviations from normal computer network behaviour. A main challenge in anomaly-based detection is to determine the normal network behaviour and flag the anomaly. Machine learning models can be trained to classify the data into categories 'normal' or 'anomaly'. In supervised machine learning the data have to be recorded over a period of time to create a model of normal behaviour. This process can take significant time and may be computationally expensive for complex computer networks. To decrease the period of learning and increase the accuracy of the model the number of features can be significantly reduced. In this paper, we present one pre-processing technique based on the feature selection. A subset containing nine features and normalized instances is used for evaluation of IDSs based on k-NN, wk-NN, SVM, Decision Tree and FNN models. The results show that the highest accuracy gives the wk-NN model, while the shortest computation time has Decision Tree model. Overall, FNN shows higher accuracy and shorter computation time, compared to the other models. In our further work we will present the results of experiments conducted on hybrid model based on wk-NN and FNN which detects variation in the decision on detected anomalies.

References

[1] Ambedkar, Ch. & Babu V. K. (2015). Detection of Probe Attacks Using Machine Learning Techniques. *International Journal of Research Studies in Computer Science and Engineering*. *2*(3), 25-29.

[2] Burgess, M. (1998). Computer Immunology. *12th USENIX Conference on System Administration*, Boston, MA, USA, 06-11 December 1998, 283-298.

[3] [Dataset] The UCI KDD Archive Information and Computer Science University of California, Irvine. Last modified: October 28, 1999. Retrieved from http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[4] Ganpathy, S., Vijayakumar, P., Yogesh, P. & Kannan, A. (2015). An Intelligent CRF Based Feature Selection for Effective Intrusion Detection. *The International Arab Journal of Information Technology*, published online March 8, 44-50.

[5] Hechenbichler, K. & Schliep, K. (2004). Weighted *k*-Nearest-Neighbor Techniques and Ordinal Classification. *Sonderforschungsbereich* 386, Paper 399 (2004). Retrieved from https://epub.ub.uni-muenchen.de/1769/1/paper_399.pdf

[6] Jayakumar, K., Revathi, T. & Karpagam, S. (2015). Intrusion Detection using Artificial Neural Networks with Best Set of Features. *The International Arab Journal of Information Technology*. *12*(6A), 728-734.

[7] Protic, D. (2015). Feedforward neural networks: The Levenberg-Marquardt optimization and the optimal brain surgeon pruning. *Military Technical Courier*. *3*(63), 11-28.

[8] Protic, D. (2018). Review of KDD Cup '99, NSL-KDD and Kyoto 2006+ datasets. *Vojnotehnički glasnik/Military Technical Courier*, *66*(3), 580-596. DOI: 10.5937/vojteh-16670.

[9] Protic, D. & Milosavljevic, M. (2006). NNARX model of speech signal generating system: test error subject to modeling mode selection. *25th International conference on microelectronics*, 643-646.

[10] Protic, D. & Stankovic, M. (2018). Anomaly-Based Intrusion Detection: Feature Selection and Normalization Influence to the Machine Learning Models Accuracy. *European Journal of Formal Science and Engineering*. *1*(3), 43-48. ISSN 2601-8675 (Print), ISSN 2601-8683 (Online).

[11] Ramasamy, R. & Rani, S. (2018). Modified Binary Bat Algorithm for Feature Selection in Unsupervised Learning. *The International Arab Journal of Information Technology. 15*(6), 1060-1067.

[12] Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Comput.Surv. 34*(1),1-47.

[13] Singh S., Khan Z. & Saxena K. (2014). Intrusion Detection Based on Artificial Intelligence Technique. *International Journal of Computer Science Trends and Technology. 2*(4), pp. 31-35.

[14] Song, J., Takakura, H., Okabe, Y., Eto, M., Inoue, D. & Nakao, K. (2011). Statistical Analysis of Honeypot Data and Building of Kyoto 2006+ Dataset for NIDS Evaluation. In: *Proc.1st Work-shop on Building Anal. Datasets and Gathering Experience Returns for Security*. Salzburg. April 10-13, 29-36.