Detecting Network Intrusion beyond 1999

Klein, Jan; Bhulai, Sandjai; Hoogendoorn, Mark; Van Der Mei, Rob; Hinfelaar, Raymond

published in
Proceedings - 2018 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2018
2019

DOI (link to publisher)
10.1109/WI.2018.00017

document version
Publisher's PDF, also known as Version of record

citation for published version (APA)
Klein, J., Bhulai, S., Hoogendoorn, M., Van Der Mei, R., & Hinfelaar, R. (2019). Detecting Network Intrusion beyond 1999: Applying Machine Learning Techniques to a Partially Labeled Cybersecurity Dataset. In *Proceedings - 2018 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2018* (pp. 784-787). [8609692] IEEE. https://doi.org/10.1109/WI.2018.00017

2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)

Detecting Network Intrusion Beyond 1999: Applying Machine Learning Techniques to a Partially Labeled Cybersecurity Dataset

Jan Klein[*], Sandjai Bhulai[‡], Mark Hoogendoorn[‡], Rob van der Mei[*] and Raymond Hinfelaar[§]
[*]*Centrum Wiskunde & Informatica*, Amsterdam. [‡]*Vrije Universiteit*, Amsterdam. [§]*Ministerie van Defensie*, The Hague.
Email: [*]{j.g.klein,r.d.van.der.mei}@cwi.nl, [‡]{s.bhulai,m.hoogendoorn}@vu.nl, [§]r.hinfelaar@mindef.nl

***Abstract*—This paper demonstrates how different machine learning techniques performed on a recent, partially labeled dataset (based on the Locked Shields 2017 exercise) and which features were deemed important. Moreover, a cybersecurity expert analyzed the results and validated that the models were able to classify the known intrusions as malicious and that they discovered new attacks. In a set of 500 detected anomalies, 50 previously unknown intrusions were found. Given that such observations are uncommon, this indicates how well an unlabeled dataset can be used to construct and to evaluate a network intrusion detection system.**

***Index Terms*—intrusion detection, cybersecurity, partially labeled, autoencoder, gradient boosting machine**

I. Introduction

With the continuing rise in the presence of cyberattacks, it becomes more and more important to protect computer networks and data from unauthorized access. To discover malicious activities, network intrusion detection systems (NIDSs) have been developed. There are different types of systems: for example, aimed at *misuse detection* or at *anomaly detection* [1]. The first relies on the properties of known attacks to recognize these in new network traffic, while the second type focuses on activities different from what is expected. Since malicious activities are assumed to exhibit abnormal behavior, anomaly detection is effective in discovering novel intrusions. Both approaches to network intrusion detection have been widely researched with the aid of machine learning (ML) [2].

However, in the operational setting, anomaly detection systems are highly underrepresented. Only recently the use of these systems has moderately increased. It is argued in [1] that this low popularity is due to "*(i)* a very high cost of errors; [and] *(ii)* a lack of training data; [...]". Especially intrusions labeled as harmless (false negatives) are undesirable, since they can result in serious damage to the network. There are few publicly available cyber datasets that can be used to evaluate NIDSs. In literature [3], [4], the NSL-KDD dataset is usually used for validation. This is a refined version of a dataset generated in 1999. Since it is almost 20 years old, it is impossible for this dataset to resemble current network traffic.

The aim of this paper is, firstly, to study the performance of different ML techniques on a semi-labeled recent dataset: some observations are known to be malicious, but for the rest it is unknown whether they are malicious or benign. We apply both unsupervised and supervised ML techniques for anomaly detection and examine their results. Secondly, this research explores the importance of each feature in detecting cyberattacks. Thirdly, the cybersecurity expert (part of the research team) analyzes two samples of observations which the models have assigned either a high or low probability of being malicious. He determines whether the models correctly classified the samples, possibly resulting in the discovery of previously unknown malicious activities. Fourthly, the results are compared with a benchmark technique from literature.

TABLE I
Additional raw features in conn.log

Name	Description
subnet_orig	source subnet of connection
subnet_resp	destination subnet of connection
PCR_bytes	relative difference between #bytes sent and received
PCR_pkts	relative difference betw. #packets sent and received
PCR_ip_bytes	relative difference betw. #IP bytes sent and received

II. Locked Shields 2017

The unique and recent dataset used in this research is based on the Locked Shields exercise of 2017 (LS'17), organized by the NATO Cooperative Cyber Defence Centre Of Excellence. In short, the participating teams were given control over a fictional country and were expected to maintain its IT networks and services. Information about the proceedings during the LS'17 exercise can be found in [5].

The LS'17 data consists of several log files collected by network security monitor Bro (developed by Vern Paxson). One of the largest files is conn.log, which contains general information on TCP/IP, UDP, and ICMP traffic of one of the teams. A description of the standard variables in this log file can be found in [6], while Table I describes the features that we added beforehand. After this preliminary exploration, conn.log consisted of 22 relevant variables with N = 15,369,736 observations. Lastly, some IP addresses appearing in this dataset were indicated as malicious, allowing us to label the corresponding observations as intrusions.

III. Methods

As mentioned before, the LS'17 dataset is semi-labeled: 8,442 ($\approx 0.055\%$) observations are indicated as malicious, while the categories of the remaining 15,361,294 ($\approx 99.945\%$) connections are not known.

978-1-5386-7325-6/18/$31.00 ©2018 IEEE
DOI 10.1109/WI.2018.00017

A. Pre-processing

Firstly, an adequate target dataset had to be assembled by extracting the appropriate features from the raw LS'17 data.

1) Feature extraction: For some of the considered ML algorithms solely numerical values are permitted. Therefore, one-hot encoding was used on the categorical variables subnet_orig and subnet_resp and on the features which indicate the protocol and service of the connection. However, the four features representing the source and destination IP addresses (hosts) and port numbers all have a large number of possible categories, and therefore, are called *overcategorized* variables. It would be inefficient to introduce a binary feature for every category. The overcategorized variables were replaced by extracting new features from them. The previously mentioned NSL-KDD dataset, described in [4], was taken as an example for this process. For instance, a variable was constructed which counts the number of connections in the last τ seconds going to the same destination host as connection $i \in \{1, \ldots, N\}$. Next, several new features were extracted from conn_state. The categories in this variable indicate whether observation i raised an error. Now, *fractional error variables* were constructed which indicate the fraction of connections inside some count feature of connection i that have an error state. Going further beyond the scope of the NSL-KDD dataset, the constructed count variables were also paired with each other. More details about this procedure can be obtained from the corresponding author upon request.

Since relevant information from the categorical variables had been extracted, they were removed from the dataset. Also, the features ts and history were discarded, since most of their added value was already captured by the variables mentioned before. The extracted features were aggregated over time windows of $\tau = 2$ and $\tau = 120$ seconds giving rise to a short-term and long-term analysis of the data. After the process of one-hot encoding and feature extraction, the transformed LS'17 dataset consisted of $P = 142$ variables.

2) Data preparation: Next, the first 120 seconds of data were removed, because the aggregated temporal variables were skewed at the start of the exercise. This resulted in the removal of 614 instances ($\approx 0.004\%$). Furthermore, the dataset was randomly split into a training, validation and test set with allocation percentages 70%/15%/15%, such that $N_{\text{train}} = 10{,}758{,}386$ and $N_{\text{val}} = N_{\text{test}} = 2{,}305{,}368$. Of the total of 8,442 labeled malicious observations, $M_{\text{train}} = 6{,}004$ ended up in the training set, $M_{\text{val}} = 1{,}221$ in the validation set and $M_{\text{test}} = 1{,}217$ in the test set. Moreover, $P_{\text{train}} = P_{\text{val}} = P_{\text{test}} = 142$.

B. Autoencoder

The first model considered was an unsupervised ML method called the autoencoder. This technique works well for anomaly detection and feature dimensionality reduction [7], [8], but this has not yet been shown in network intrusion detection.

The number of layers $L \geq 3$ in the network, the number of neurons in the middle layer $P_{(L+1)/2} \in \mathbb{N}$ and the Ridge regularization shrinkage parameter $\lambda \in \mathbb{R}_+$ were the hyperparameters, and therefore, had to be determined before the training procedure could start. The activation function used was the hyperbolic tangent function. The numbers of neurons in the other layers were determined by the geometric pyramid rule, which simplifies the structure of the model. This rule defines the number of neurons P_l in layer $l \in \{1, \ldots, L\}$ by

$$P_l = P_{(L+1)/2} \cdot \left(\frac{P_{\text{train}}}{P_{(L+1)/2}} \right)^{\frac{|2l-(L+1)|}{L-1}} .$$

The optimal values of the three hyperparameters were determined with the aid of the validation set. This set was fed to a trained autoencoder and for every observation i the mean squared difference (MSE_i) between its input and output was calculated. The assumption is that a malicious observation cannot be correctly reconstructed, and hence, results in a relatively high MSE_i. The hyperparameter combination $(L, P_{(L+1)/2}, \lambda)$ chosen was the one that maximized the *Discounted Cumulative Gain* (DCG). A large value of DCG indicates that the known malicious observations obtained relatively high MSEs. In a 'perfect' MSE ranking all known malicious instances obtain the highest MSEs and in the worst ranking they obtain the lowest, implying that DCG is bounded. Hence, for convenience, the *normalized DCG* (nDCG) was considered. This is a linearly scaled version of DCG such that it takes values in $[0, 1]$. Additionally, when the MSE ranks are randomly assigned, then $\mathbb{E}(\text{nDCG}) \approx 0.0532$ and $\text{Var}(\text{nDCG}) \approx 3.68 \times 10^{-6}$.

C. Gradient Boosting Machine

The second ML method considered is called the gradient boosting machine (GBM), which can also be applied for anomaly detection purposes [9]. GBM requires labeled training data, therefore all unknowns were classified as benign: making the malicious class highly underrepresented ($\approx 0.056\%$).

Here, the hyperparameters were the number of trees $T \in \mathbb{N}$ to be constructed and the zero-class sampling factor $\alpha_0 \in (0, 1)$, which indicates the fraction of benign labeled observations to sample during training such that the two classes can be balanced. The optimal values for the hyperparameters were determined by the validation set similar to what was done for an autoencoder. Each observation i in the validation set was fed to the trained GBM (given some parameter combination) and nDCG was calculated.

IV. Results

The procedures described in the Section III were conducted in R. Both ML techniques are available in the h2o package. In the benchmark evaluation the C5.0 algorithm was used, which is available in the C50 package.

A. Results Autoencoder & Gradient Boosting Machine

To construct an autoencoder network, the training set was used without the 6,004 (0.056%) known intrusions. These observations were assumed to be abnormal and go against the purpose of an autoencoder to design a representation of normal behavior. Note that there were still possibly many

TABLE II
OPTIMAL HYPERPARAMETER COMBINATIONS AUTOENCODER AND GBM

	$(L, P_{(L+1)/2}, \lambda)$	(T, α_0)	$\overline{\text{nDCG}}$	$s^2(\text{DCG})$
auto.	$(9, 15, 10^{-4})$	–	0.230	0.00139
GBM	–	$(300, 8c_{\text{train}})$	0.992	9.38×10^{-6}

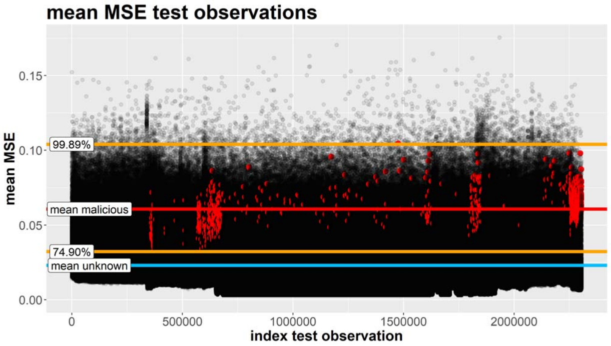

Fig. 1. mean MSE of test observations

unknown intrusions in the training set. For gradient boosting the complete training set was used with the assumption that the unknown observations were benign.

1) Hyperparameter tuning: Since the training procedure is a stochastic process, 60 models were trained to obtain an acceptable estimate for the expected value of nDCG per hyperparameter combination and for each method. The relatively optimal hyperparameters for both techniques are shown in Table II, with $c_{\text{train}} = M_{\text{train}}/(N_{\text{train}} - M_{\text{train}})$ the ratio between the number of labeled malicious and unlabeled observations in the training set.

2) Evaluation on known intrusions: The test set was used for the evaluation. For the autoencoder technique this resulted in a reconstruction MSE for each test observation.

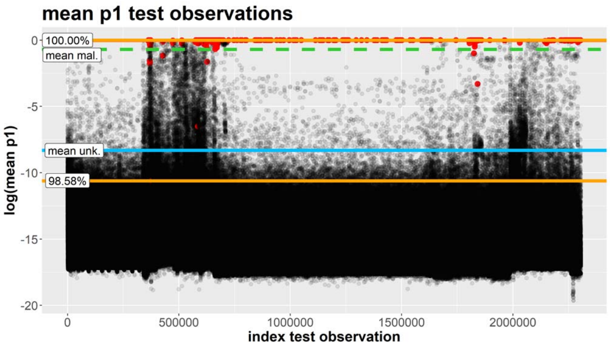

Fig. 2. log mean intrusion probability of test observations

TABLE III
TEST RESULTS AUTOENCODER AND GBM

	intr. inf	intr. sup	mean unk.	mean intr.	nDCG
auto.	0.0323 74.90%	0.104 99.89%	0.0230 62.01%	0.0607 95.02%	0.176
GBM	$2.5 \cdot 10^{-5}$ 98.59%	1.000 100.00%	$2.4 \cdot 10^{-4}$ 99.46%	0.98 99.95%	0.993

TABLE IV
FIVE MOST IMPORTANT FEATURES BY AUTOENCODER

feature	**share**
subnet_orig	8.14%
srv	6.74%
resp_p_same_srv_rate_120s	4.88%
resp_p_same_srv_rate_2s	3.70%
subnet_resp	3.69%

TABLE V
FIVE MOST IMPORTANT FEATURES BY GBM

feature	**share**
resp_p_same_srv_rate_120s	52.89%
resp_p_same_resp_h_rate_120s	16.19%
srv	8.00%
subnet_orig	7.69%
srv_same_resp_p_rate_120s	1.89%

This procedure was repeated 50 times to obtain 50 MSEs for each test instance. The mean was taken over all these repetitions to estimate the expected reconstruction MSE. The same procedure was done for the GBM, which resulted in 50 intrusion probabilities per test observation. However, each run yielded a different threshold probability $\theta \in (0, 1)$ representing the border between the predicted labels 0 (benign) and 1 (malicious). To allow for comparison, each time the threshold θ was transformed to be 0.5 and the probabilities were changed accordingly. This was done by applying the function $f_\theta : [0, 1] \to [0, 1]$ given by

$$f_\theta(p) = \frac{(1-\theta)p}{(1-2\theta)p + \theta}$$

to all probabilities. This function has desirable properties: *(i)* it is continuously differentiable, *(ii)* $f_\theta(0) = 0$, *(iii)* $f_\theta(\theta) = 0.5$, *(iv)* $f_\theta(1) = 1$ and *(v)* $f'_\theta(p) \geq 0$. The results are shown in Figures 1 and 2. In the first figure the mean MSE ($\overline{\text{MSE}}$) of every test observation for the autoencoder is plotted, while in the second the mean (transformed) intrusion probabilities ($\overline{p^{(1)}}$) for the GBM are shown. Note that the natural logarithm of the probabilities was taken to enhance the amount of information the plot conveys. In both figures, the black points are the unknown instances, while the red points are the labeled intrusions. The lowest orange line indicates the *intrusion infimum*: the level such that all known intrusions are above this line. Likewise, the highest orange line is the *intrusion supremum*: the level such that all labeled malicious instances are below this line. Moreover, the blue line indicates the mean $\overline{\text{MSE}}$ or mean $\overline{p^{(1)}}$ of the unknown instances and the red line that of the labeled observations. The green dashed line in Figure 2 corresponds to probability 0.5 (99.99th percentile) and represents the border between the predicted classes. The corresponding percentiles of these lines and the nDCGs of the two expected models are presented in Table III.

3) Feature Analysis: For the autoencoder, the per feature squared errors were obtained for each test observation. To determine which variables were important for the autoencoder, the share of each feature in the total squared error sum

TABLE VI
ANALYSIS BY THE EXPERT

	benign	malicious	unknown
high	65.4% (327)	10.8% (54)	23.8% (119)
low	98.6% (986)	0.1% (1)	1.3% (13)
class total	87.53% (1,313)	3.67% (55)	8.80% (132)

TABLE VII
ACCURACY MEASURES ON TEST SAMPLES

	nDCG	α	precision	recall	F_1 score
autoencoder	0.176	5.76%	0.142	0.982	0.248
GBM	0.993	0.0668%	1.00	0.0727	0.136
benchmark	0.956	0.403%	0.571	0.0727	0.129

was calculated. For the GBM, the function h2o.varimp was used to determine the relative predictive strength of each variable. The features which were partitioned into binary variables during one-hot encoding were unified again in both computations. The five most important variables per technique for the average known intrusion are shown in Tables IV and V. A complete explanation of these features can be obtained from the corresponding author upon request.

4) Evaluation by expert: The cybersecurity expert analyzed two samples of test observations. The 'high sample' contained 500 random observations from the top 6% which yielded the largest MSEs. The (relatively) 'low sample' was a random set of $1{,}000$ observations excluding the top 6%. Table VI shows the expert classification of the two samples. For the autoencoder, the precision (0.142), recall (0.982) and F_1 score (0.248) were all maximal when $\alpha = 5.76\%$ of the data was assumed to be anomalous. This threshold value α had to be imposed, because the technique is unsupervised. The GBM classified $\alpha = 0.0668\%$ of the records as malicious, resulting in precision 1.00, recall 0.0727 and F_1 score 0.136.

B. Results Benchmark

One of the classification techniques that Dhanabal et al. [4] used on the NSL-KDD dataset was the C4.5 decision tree algorithm. Here, the improved C5.0 algorithm [10] was applied to the LS'17 dataset with the subset of variables that matched the set used by Dhanabal as best as possible. Their feature set does not contain the variables in Table I, the features aggregated over 120 seconds, and the variables paired with the constructed count variables. This reduced the number of features from $P = 142$ to $P_{\text{bench}} = 39$.

1) Evaluation on known intrusions: Since C5.0 is a supervised ML technique, all unknown observations were considered to be benign. The hyperparameters were based on those selected for the GBM. The expected performance measure $\text{nDCG}_{\text{bench}} \approx 0.956$ was estimated by training 50 models.

2) Evaluation by expert: The performance of the C5.0 method was also evaluated by the cyber analyst. This technique classified $\alpha = 0.403\%$ of the test observations as malicious with precision 0.571, recall 0.0727 and F_1 score 0.129.

V. DISCUSSION

At first glance, the GBM seems better, because all known attacks were present in the top 1.42% of the data and the means of the two classes vastly differed (Figure 2 and Table III). Figure 1 shows that the autoencoder was able to notice that the labeled malicious activities do not conform to the normal behavior of the network traffic, because all test intrusions yielded MSEs in the top 25.10%. Unfortunately, this result is not desirable when dealing with millions of observations. There was a clear distinction between the means of both classes, however.

Next, two samples were thoroughly analyzed. As Table VI shows, one new malicious activity was discovered in the low sample and no fewer than 54 intrusions were found in the high sample (four of them were already known). Table VII gives a summary of the results obtained in this research. It shows that the autoencoder was able to discover almost all of the real attacks in the two samples (recall $= 0.982$), while the GBM found almost none of the actual intrusions (recall $= 0.0727$). Yet, only a small fraction of the found anomalies by the autoencoder were in fact malicious (precision $= 0.142$), while all predicted intrusions by the GBM were correctly classified (precision $= 1.00$).

The results of the benchmark C5.0 algorithm justified the addition of the new features, since Table VII shows that $\text{nDCG}_{\text{bench}} < \text{nDCG}_{\text{GBM}}$. This is a statistically significant difference according to a one-sided Mann-Whitney test on the two samples of DCGs (p-value $< 2.2 \times 10^{-16}$). Moreover, the feature analyses of the autoencoder and the GBM showed that the features introduced here are important in detecting and discovering intrusions (Tables IV and V). The variables aggregated over a time horizon of 120 seconds, the variables added by the expert, and the extra variables which were not present in [4] all had a large influence on the classification.

REFERENCES

[1] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning For Network Intrusion Detection" IEEE Symposium on Security and Privacy, 2010.

[2] M. Zamani and M. Movahedi, "Machine Learning Techniques for Intrusion Detection", 2015.

[3] S. Revathi and A. Malathi, "A Detailed Analysis on NSL-KDD Dataset Using Various Machine Learning Techniques for Intrusion Detection," International Journal of Engineering Research & Technology, vol. 2, no. 12, 2013, pp. 1848–1853.

[4] L. Dhanabal and S.P. Shantharajah, "A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms," International Journal of Advanced Research in Computer and Communication Engineering, vol. 4, no. 6, 2015, pp. 446–452.

[5] K. Maennel et al., "Improving and Measuring Learning Effectiveness at Cyber Defense Exercises," Springer 2017, pp. 123–138.

[6] The Bro Platform, "base/protocols/conn/main.bro", www.bro.org/sphinx/scripts/base/protocols/conn/main.bro.html

[7] S. Zhai et al., "Deep Structured Energy Based Models for Anomaly Detection," Proc. 33rd International Conference on Machine Learning, 2016.

[8] S. Glander, "Autoencoders and anomaly detection with machine learning in fraud analytics", shiring.github.io/machine_learning/2017/05/01/fraud, 2017.

[9] M. Reif et al., "Anomaly detection by combining decision trees and parametric densities," 19th International Conference on Pattern Recognition, 2008.

[10] RuleQuest, "Is See5/C5.0 Better Than C4.5?", www.rulequest.com/see5-comparison.html, 2017.