Corizzo *et al. J Surveill Secur Saf* 2020;1:140-50
DOI: 10.20517/jsss.2020.15

Journal of Surveillance, Security and Safety

Original Article

Open Access

Feature extraction based on word embedding models for intrusion detection in network traffic

Roberto Corizzo[1], Eftim Zdravevski[2], Myles Russell[1], Andrew Vagliano[3], Nathalie Japkowicz[1]

[1]Department of Computer Science, American University, Washington, DC 20016, USA.
[2]Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University, Skopje 1000, North Macedonia.
[3]Department of Computer Science, Northwestern University, Evanston, IL 60208, USA.

Correspondence to: Dr. Roberto Corizzo, Department of Computer Science, American University, 4400 Massachusetts Avenue NW, Washington, DC 20016, USA. E-mail: rcorizzo@american.edu

How to cite this article: Corizzo R, Zdravevski E, Russell M, Vagliano A, Japkowicz N. Feature extraction based on word embedding models for intrusion detection in network traffic. *J Surveill Secur Saf* 2020;1:140-50. http://dx.doi.org/10.20517/jsss.2020.15

Received: 30 Apr 2020 **First Decision**: 15 Jun 2020 **Revised**: 27 Jun 2020 **Accepted**: 17 Jul 2020 **Available online**: 28 Dec 2020

Academic Editor: Xiaofeng Chen **Copy Editor**: Cai-Hong Wang **Production Editor**: Jing Yu

Abstract

Aim: The analysis of network traffic plays a crucial role in modern organizations since it can provide defense mechanisms against cyberattacks. In this context, machine learning algorithms can be fruitfully adopted to identify malicious patterns in network sessions. However, they cannot be directly applied to a raw data representation of network traffic. An active thread of research focuses on the design and implementation of feature extraction techniques that aim at mapping raw data representations of network traffic sessions to a new representation that can be processed by machine learning algorithms.

Methods: In this paper, we propose a feature extraction approach based on word embedding models. The proposed approach extracts semantic features characterized by contextual information that is hidden in the raw data representation.

Results: Our experiments conducted on three datasets showed that our feature extraction approach based on word embedding models has the potential to increase the classification performance of conventional machine learning algorithms that are applied to intrusion detection, and it is competitive with known feature extraction baselines in the state-of-the-art.

Conclusion: This study shows that word embedding models can be used to carry out intrusion detection tasks accurately. Feature extraction based on word embedding models requires a higher computational time than simpler techniques, but leads to a higher accuracy, which is important for the identification of complex attacks.

Keywords: Feature extraction, intrusion detection, network traffic, anomaly detection, word embeddings, language models

INTRODUCTION

Intrusion detection systems (IDS) play a fundamental role in modern organizations, providing defense mechanisms against cyberattacks. IDS monitor and analyze the traffic using different sources of information, with the purpose of identifying intrusions and other security breaches. Differently than firewalls, which limit access between networks to prevent intrusions, IDS evaluate a potential intrusion when it takes place, signal an alarm, and may terminate the connection. The most popular categories of IDS include network-based IDS and host-based IDS (HIDS)[1]. The former analyze network packets on an entire subnet[2,3], whereas the latter consist of an agent on a host that analyzes system calls, file system changes, and logs[4-7]. In this study, we focused on HIDS and, more specifically, machine learning-based tools to support it. One opportunity in this domain consists in monitoring and analyzing network traffic represented in the form of network sessions, also known as traces[8]. One of the most popular data representations for traces is that known as sequence of system calls[9], i.e., a sequence of requests that programs submit to the operating system kernel to perform any action. The ordering, type, length and other attributes of system calls made by an application process can provide a unique signature or trace. Such information is highly informative, and it is exploited in current IDS to help distinguish between normal and abnormal behaviors in a network session[10].

Relevant benchmark datasets such as the Defense Advanced Research Projects Agency dataset[11] and the Knowledge Discovery and Data Mining Tools Competition (KDD'99) dataset[12] have been analyzed in a large number of studies for the past two decades[2-4,13-16]. However, such datasets do not cover up-to-date attack scenarios, and therefore, they are not considered to be challenging at present. More recently, the Australian Defence Force Academy Linux Dataset (ADFA-LD)[5,17,18], as well as the Next-Generation Intrusion Detection System Dataset (NGIDS-DS)[18,19] and the Web Conference 2019 (WWW2019)[20] datasets, succeeded in filling this gap, presenting new and relevant types of attacks conceived to assess the accuracy of modern intrusion detection tools. The datasets present thousands of system call traces collected from a Linux local server, with normal and attack behaviors.

Traditional machine learning algorithms can be fruitfully exploited to identify malicious patterns in network sessions, which can be subsequently filtered. Examples of approaches in the literature include Support Vector Machines[13], Artificial Neural Networks[2], classification of association rules[14,15], decision trees[4], random forests[3], and ensembles of classifiers[16].

However, machine learning algorithms cannot be directly applied to a raw data representation of network traffic, such as sequences of system calls. For this reason, an active thread of recent research[5-7] focuses on the design and implementation of feature extraction techniques that aim at mapping sequences of system calls to a new representation that can be processed by machine learning algorithms. Figure 1 shows the typical analytical workflow that is carried out to perform machine learning-based intrusion detection.

Focusing on feature extraction approaches in the literature, pattern-based and frequency-based methods represent the most popular classes. Pattern-based approaches identify patterns in sessions, consisting of multiple co-occurring system calls in a trace, whereas frequency-based approaches[5,21,22] extract feature vectors in which entries represent the frequency of a system call in a trace. Although the former generally lead to a more accurate profile of the normal class, they are computationally more expensive. On the other hand, the latter are more computationally efficient, but the resulting representation does not take into account the position of system calls in the trace[6].

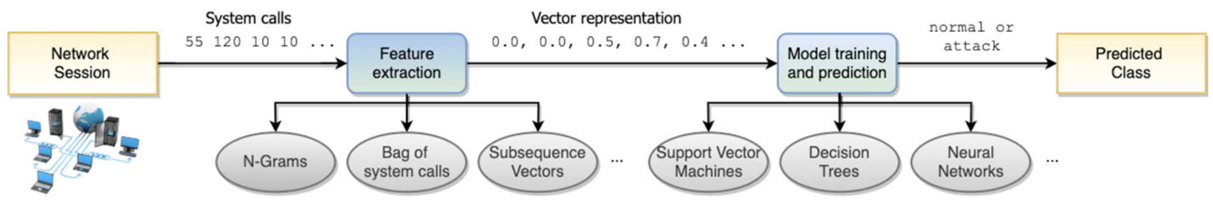

Figure 1. Analytical workflow for machine learning-based intrusion detection in network traffic. Network sessions in the form of sequences of system calls are fed to a feature extraction method, which returns vector data that can be exploited in the modeling step by machine learning and deep learning algorithms. The outcome is a returned class for each session (normal, attack)

One example of a pattern-based approach is the *N-gram* feature extraction method[7], which generates pattern data, converting each class into a two-dimensional array (or into a matrix) representation. In this representation, columns are grams, i.e., attributes, and rows are instances, i.e., traces. The entries in the matrix are the number of occurrences of each N-gram in the traces. Considering that the number of grams for any of the classes is very high compared to the number of instances, it is common to aim for a reduction in the number of attributes, taking into account the most frequent grams.

Focusing on frequency-based approaches, the Subsequence Vector method[5] transforms a trace into a vector, where entries are calculated as the product between the system call and its frequency in the trace. The limitations of this approach consist in the generation of sparse vector representations and in the independent treatment of each system call. Another similar method is known as Bag of System Calls[22], which enumerates all system calls and transforms system traces into fixed-length vectors that contain the frequencies of each system call. One alternative to exploit frequency vectors is to apply weighting schemes to the observed frequencies. This type of approach is followed in the study by Xie *et al.*[5], which proposes the application of Term-Frequency and Inverse Document Frequency (TF-IDF) to extract normalized frequency vectors. Another alternative consists in performing dimensionality reduction to obtain a more compact vector representation that does not present sparsity issues. One example of this type of approach can be found in the study by Xie *et al.*[6], which proposes the application of principal component analysis on frequency vectors.

However, one major challenge in feature extraction is to represent the contextual information of system calls in traces effectively. Contextual information in sequential data with a complex structure can be often hidden and difficult to extract[23,24], especially for pattern-based and frequency-based approaches that do not take into account the temporal dynamics of system calls in traces.

In this paper, we propose a new feature extraction method for sequential network traffic data in the form of sequence of system calls. Following the success of state-of-the-art feature extraction methods inspired by Natural Language Processing (NLP), our method leverages a word embedding-based approach to extract contextual information that can be exploited in the subsequent classification step by any machine learning algorithm. To the best of our knowledge, this is the first study that presents feature extraction based on word embedding models and, in particular, presents a combination approach with TF-IDF and Word2Vec models. Moreover, in our study, we also investigated feature extraction based on Doc2Vec. We performed experiments to evaluate the effectiveness of different machine learning classifiers with our extracted features, and compared them with different state-of-the-art feature extraction methods in a number of different scenarios.

METHODS

In this section, we provide a brief overview on word embedding models and some examples of their successful application. Subsequently, we describe our proposed feature extraction method for intrusion detection in network traffic, based on word embedding models.

Word embedding models are commonly adopted techniques for language modeling and feature learning in NLP. These techniques map words and sentences into low dimensional feature vectors that can be exploited by automated analytical tools. Examples of word embedding techniques include neural networks[25], probabilistic models[26], and approaches based on dimensionality reduction applied to a word co-occurrence matrix[27].

Some word embedding techniques aim at extracting a vector representation for a word in terms of co-occurring words, whereas others express a word in terms of vector of linguistic contexts[28]. Recently, particular interest has been devoted to the latter, since they attempt to characterize the semantics of words and sentences, on the basis of the intuition by which a word is characterized according to the company it keeps[29,30].

One example of a groundbreaking technique in this field is represented by Word2Vec[25]. Its ability to represent implicit relationships between words has resulted in substantial machine learning improvements on domains by contextual information. Some examples include the classification of news articles and tweets[31], the analysis of biological data for the prediction of therapeutic peptides[32], the detection of malware activity on Android devices[33], and the recommendation of contents in social networks[34]. Similarly to these studies, the method proposed in this paper leverages Word2Vec as a method to extract word embeddings. However, none of these approaches applies Word2Vec to network traffic sessions in the form of sequences of system calls. Our aim was to propose a pipeline that makes Word2Vec applicable to data in this domain. In addition, we proposed an approach to weight the feature extracted according to its importance.

The common result obtained in[31,33,34] is that performing the learning task on top of the newly extracted data representation obtained by means of word embedding models, leads to an improved accuracy. The motivation is that the newly extracted representation presents useful semantic features that were hidden in the initial raw data representation, thus facilitating machine learning tools to perform classification and improving the machine learning classification task. Following the same intuition, and motivated by the success in different domains, our proposed method leverages a Word2Vec word embedding model to extract contextual information that can be exploited in the subsequent classification step by any machine learning algorithm. In particular, we exploit Word2Vec to obtain a k-dimensional numerical embedding vector that entails the semantic representation of a system call. Given a set of labeled traces T_L, for which the class attribute is known (normal or attack), we train a Word2Vec model to generate semantic vectors for all traces $t \in T_L$. The feature extraction process from network traces exploiting a Word2Vec model is shown in Figure 2. One alternative to Word2Vec is represented by Doc2Vec, which extracts a unique representation for each document.

The novelty in this paper is to exploit Word2Vec in combination with a TF-IDF model[35]. More specifically, a TF-IDF model is trained to subsequently perform a weighted transformation of the semantic representation of a system call extracted by Word2Vec. The rationale for the adoption of such a model is that the representation vector of a trace should be weighted according to the saliency of the system calls it contains. More precisely, system calls that appear in several traces are less indicative of the content of a trace, whereas system calls that appear rarely, should be more discriminative. The TF-IDF weighting allows us to capture these properties and give more weight to system calls that are frequent in a trace but rare in the overall collection of traces.

Each trace $t \in T_L$ is represented as a bag of system calls $b(t) = \{s_1, s_2, \ldots, s_k\}$ of arbitrary length. Next, the Word2Vec model converts a system call $s \in b(t)$ into a semantic vector $w(s)$ that is multiplied by the TF-IDF score $weight(s)$ calculated as follows:

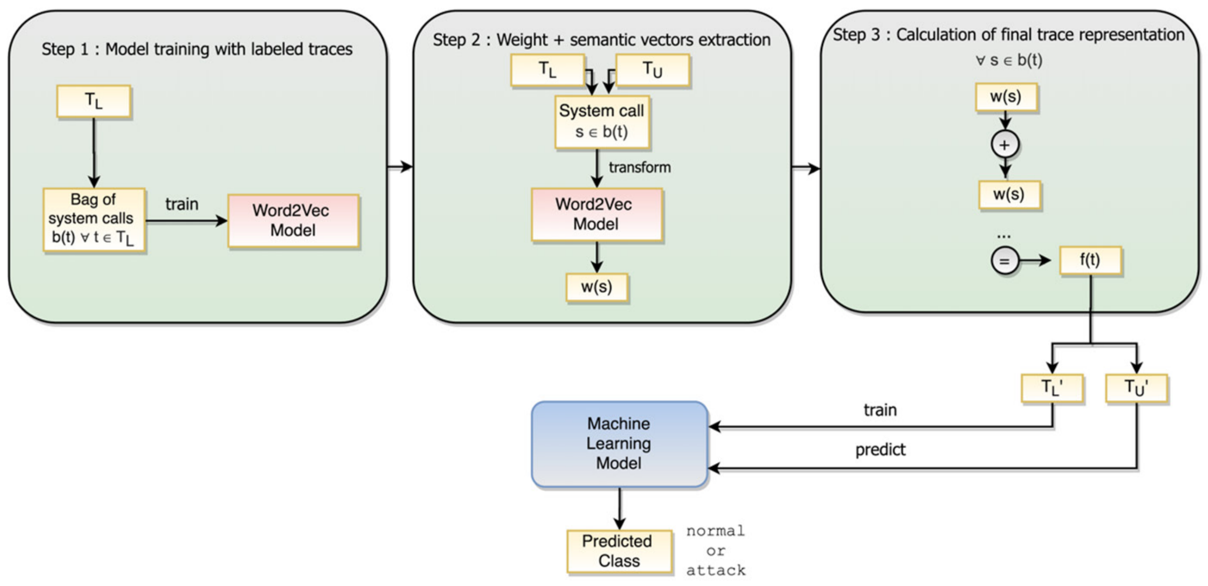

Figure 2. Graphical representation of feature extraction based on a Word2Vec model

$$weight(s) = freq_{s,t} \times log_2 \frac{|T_L|}{freq_{s,T_L}},$$

where $freq_{s,t}$ is the frequency of the system call s in the trace t, and $freq_{s,T_L}$ is the frequency of the system call s in the entire collection. To extract a single vector representation for each trace, we exploit the "additive compositionality" property of word embeddings. This property guarantees that similar words appear close to each other in the feature space, and that the sum of their embedding vector representation resembles an *AND* concatenation. By analogy, in our domain, if two traces (t_1, t_2) appear in the same context, their sum vectors obtained as the sum of the embedding vectors of the corresponding system calls will still be close to each other. Therefore, the final vector representation $f(t)$ of a trace t is computed as:

$$f(t) = \sum_{s \in b(t)} w(s) \times weight(s).$$

Following this process, we obtain a new dataset $T_L' \in \mathbb{R}^{|T_L| \times k}$, consisting of the semantic vector representation $f(t)$ for each labeled trace t in T_L. This dataset can be used to train any machine learning algorithm.

Consequently, during the prediction phase, the previously trained Word2Vec and TF-IDF models are exploited to extract features for a new collection of unlabeled traces T_U. The machine learning algorithm of choice can exploit the extracted representation to predict the class attribute of each trace $u \in T_U$. The overall feature extraction process with Word2Vec and TF-IDF is shown in Figure 3.

In summary, the Word2Vec and TF-IDF models are trained with a collection of labeled traces T_L, represented as a bag of system calls (Step 1). The outputs of these models are combined to extract a new representation (T_L', T_U')from both labeled and unlabeled traces (T_L, T_U). The representation extracted by Word2Vec is a vector for each system call. Simultaneously, the TF-IDF model extracts the weight corresponding to each system call (Step 2). The multiple vectors that represent the different system calls in a trace are subsequently calculated as the weighted sum of the system calls vector representations extracted by Word2Vec and the TF-IDF weights (Step 3). A machine learning model is trained on labeled traces after feature extraction T_L' and predicts the class of unlabeled traces T_U' after the feature extraction process.

In the following section, we present our experiments aimed at comparing the classification accuracy with our proposed feature extraction technique in comparison with state-of-the-art feature extraction methods.

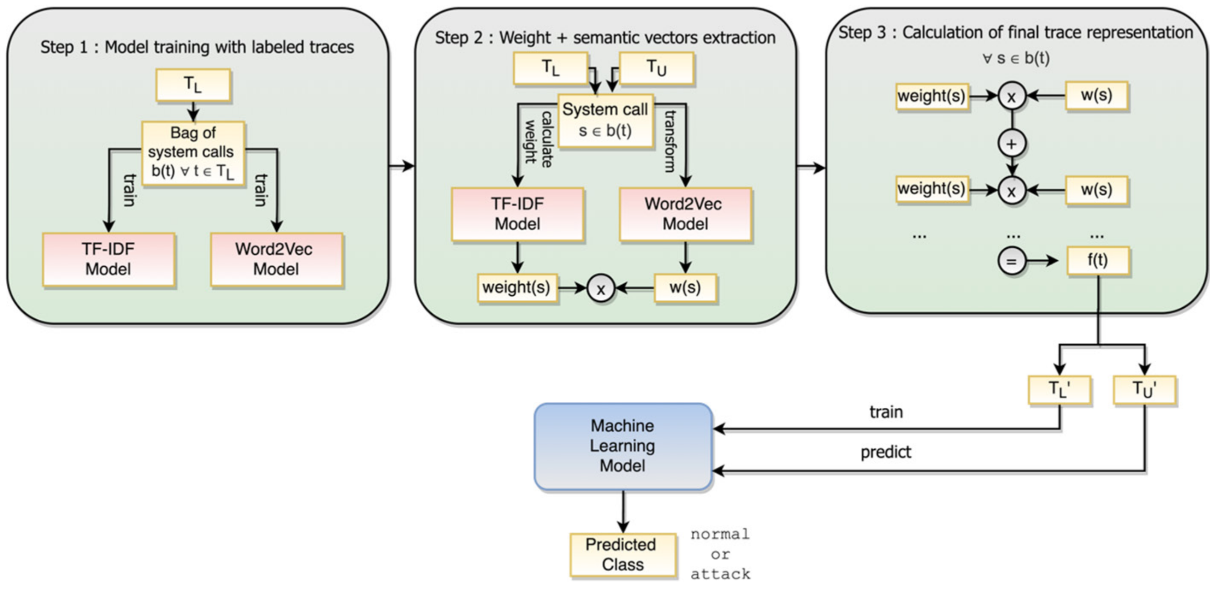

Figure 3. Graphical representation of feature extraction based on Word2Vec and Term-Frequency and Inverse Document Frequency (TF-IDF) models

The implementations of the proposed approach are publicly available in our GitHub repository (https://github.com/rcorizzo/hids-word-embedding). The implementations are available in the Python programming language, and exploit the Gensim library to train the Word2Vec, TF-IDF and Doc2Vec models. The input data format expected is in the form of text files containing sequences of system calls.

When designing the data processing pipelines, we utilized the behavioral patterns, considering that the communication between objects and in the data processing pipeline, and the input formats are the same. In particular, we used the strategy pattern by grouping the evaluated feature extraction algorithms into a single family of algorithms. We made sure that each algorithm from the Gensim library was encapsulated and had the same interface, so it could be interchanged without modifying the data processing pipeline. Similarly, we utilized the sci-kit learn library family of classification algorithms and were able to evaluate different combinations of algorithms with the tuning of their parameters, and feature extraction algorithms, to execute the whole pipeline without manual modifications. The features extracted by our implementations can be exploited by any machine learning method to perform intrusion detection as a binary classification task.

RESULTS

Competitor methods

Bag of system calls

Inspired by the study by Kang *et al.*[22], we enumerated the global set of system calls in the training data and adopted a key-value data structure. In this structure, a key corresponds to the combination of a trace ID and a system call, and the corresponding value represents the frequency of the system call in the trace. We used this data structure to generate the final dataset in matrix form.

Subsequence vector

Similarly to the aforementioned approach followed for Bag of System Calls, we enumerated the global set of system calls in the training data and generated a data matrix in which each row was a trace and each column was a system call. The entry in this matrix was initially calculated as the frequency of the system call. Subsequently, following the approach followed by Xie *et al.*[5], we re-calculated the entries in the matrix as the product between the system call and its frequency in each trace.

Table 1. Descriptive statistics for all datasets considered in this study

Dataset	Number of traces	Normal traces	Attack traces	Imbalance ratio
ADFA-LD[16]	5,951	5,205	746	6.98
NGIDS-DS[17]	37,377	19,256	18,121	1.06
WWW2019[18]	152,630	43,725	108,905	0.40

The reported imbalance ratio represents the proportion between the number of samples of the majority class and the number of samples of the minority class

Doc2Vec

The goal of Doc2Vec is to create a numeric representation of a document, regardless of its length. While word vectors represent the concept of a word, the document vector intends to represent the concept of a document. We propose this model as an alternative to Word2Vec for feature extraction applied directly to network traces.

Experimental setup

In our experiments, we assessed 5 feature extraction methods on 3 intrusion detection datasets. Descriptive statistics for all datasets considered in this study are reported in Table 1. For evaluation, we adopted a stratified 5-fold cross-validation scheme. The classification algorithm considered in our experiments was Extremely Randomized Trees (ERT), a state-of-the-art ensemble learning method based on decision trees. We emphasize that identifying the best machine learning algorithm is out of the scope of this paper. However, the features extracted with our method are general and, in principle, any machine learning algorithm can be used for the purpose of classification. Our aim was to show the potential of the features extracted using a conventional machine learning algorithm for classification.

For Word2Vec and Doc2Vec, we used a standard value for the embedding size ($k = 128$). For ERT, we used a standard configuration for the number of trees parameter ($T = 1000$). Since the datasets considered were imbalanced, we considered results in terms of macro precision, recall and F-score, to give the same importance to both classes in the average scores. We also report results in terms of area under the ROC curve (AUC). All the experimental results are reported in Table 2.

DISCUSSION

The results showed that word embedding-based feature extraction methods outperformed by a good margin all competitors with the NGIDS-DS dataset and the WWW2019 dataset. In these cases, the proposed variant of Word2Vec with TF-IDF weighting, appeared to obtain the best results. This behavior was not observed with the ADFA-LD dataset, where word embedding-based methods appear sub-optimal.

One possible explanation is that, when most of the system calls appearing in network traces are sparsely correlated, the semantic representation extracted by language models does not provide any advantage with respect to simpler frequency-based and pattern-based methods. On the contrary, the high-dimensionality of the new representation makes the classification task more difficult for the subsequent machine learning algorithm.

Another aspect that could disadvantage word embedding representations is that of the imbalance ratio between normal and attack traces. In fact, in the ADFA-LD dataset the imbalance ratio was 6.98, whereas the NGIDS-DS and WWW2019 datasets were more balanced, having an imbalance ratio of 1.06 and 0.40, respectively [Table 1]. This aspect is known to lead to increased challenges in classification tasks[36].

It is noteworthy that, among the word embedding-based methods, Doc2Vec performs poorly in all cases. This unexpected result shows that the preferred data granularity for traces in the context of intrusion

Table 2. Classification performance of extremely randomized trees models with different feature extraction techniques using different intrusion detection datasets

Dataset	Feature extraction technique	Precision (Macro)	Recall (Macro)	F-score (Macro)	Accuracy	AUC	F-score improvement over baseline (%)
ADFA-LD[16]	Bag of System Calls	0.9603	0.9244	**0.9414**	0.9752	0.9904	2.12%
	Subsequence Vector	0.9402	0.9053	0.9218	0.9670	0.9846	/
	Word2Vec	0.9376	0.8862	0.9096	0.9626	0.9791	-1.32%
	Word2Vec + TF-IDF	0.9246	0.8702	0.8948	0.9568	0.9762	-2.92%
	Doc2Vec	0.9006	0.5158	0.4985	0.8783	0.7457	-45.92%
NGIDS-DS[17]	Bag of System Calls	0.9689	0.9691	0.9690	0.9690	0.9937	1.38%
	Subsequence Vector	0.9557	0.9560	0.9558	0.9558	0.9899	/
	Word2Vec	0.9999	0.9999	0.9999	0.9999	0.9999	4.61%
	Word2Vec + TF-IDF	1.0000	1.0000	**1.0000**	1.0000	1.0000	4.62%
	Doc2Vec	0.7398	0.6560	0.6289	0.6648	0.7462	-34.20%
WWW2019[18]	Bag of System Calls	0.9568	0.9108	0.9303	0.9457	0.9823	13.68%
	Subsequence Vector	0.9830	0.8281	0.8183	0.9476	0.9048	/
	Word2Vec	0.9971	0.9929	0.9950	0.9959	0.9999	21.59%
	Word2Vec + TF-IDF	0.9990	0.9992	**0.9991**	0.9999	0.9999	22.09%
	Doc2Vec	0.8894	0.6478	0.6662	0.7981	0.7179	-18.58%

Results with three datasets: Australian Defence Force Academy Linux (ADFA-LD), Next-Generation Intrusion Detection System (NGIDS-DS), and Web Conference 2019 (WWW2019). Best results in terms of macro F-score are marked in bold. TF-IDF: Term-Frequency and Inverse Document Frequency

Table 3. Training and prediction execution time of the different feature extraction techniques using different intrusion detection datasets

Dataset	Feature extraction technique	Training time (min)	Prediction time (s)
ADFA-LD[16]	Bag of System Calls	0.13	0.25
	Subsequence Vector	0.15	0.28
	Word2Vec	1.95	0.36
	Word2Vec + TF-IDF	80.43	0.45
	Doc2Vec	0.88	0.30
NGIDS-DS[17]	Bag of System Calls	0.86	0.82
	Subsequence Vector	0.88	0.84
	Word2Vec	26.03	1.05
	Word2Vec + TF-IDF	1060.3	1.32
	Doc2Vec	4.05	0.88
WWW2019[18]	Bag of System Calls	1.23	1.18
	Subsequence Vector	1.21	1.16
	Word2Vec	18	1.45
	Word2Vec + TF-IDF	529.3	1.82
	Doc2Vec	26.08	1.22

Results with three datasets: Australian Defence Force Academy Linux (ADFA-LD), Next-Generation Intrusion Detection System (NGIDS-DS), and Web Conference 2019 (WWW2019). TF-IDF: Term-Frequency and Inverse Document Frequency

detection is that represented by system calls processed separately and aggregated using the compositionality property, rather than the whole trace represented directly as a vector.

In Table 3 we report the average execution times observed with the different feature extraction techniques. The execution was performed on a workstation equipped with an AMD Ryzen 5 1600 Processor (3200 MHz, 6 cores, 12 logical processors) with 32 GB of DDR4 RAM. The results show that frequency-based methods appear very efficient, even if they lead to sub-optimal results in terms of accuracy, as discussed before. More sophisticated feature extraction techniques are computationally more intensive, and in particular Word2Vec in combination with TF-IDF exhibits the highest execution time among all the techniques tested in this study. However, the leading time of the method is motivated by the training time of the TF-IDF dictionary which, in our experiments, is performed from scratch at every execution. In practice, there is the possibility to reduce this cost drastically by incrementally updating the TF-IDF model. Moreover, once models are trained and deployed, their prediction time appears similar for all of them, on the order of milliseconds. We argue that, in a production setting, training models from scratch is not required continuously, but periodically, and it can be performed offline, while previously learned models are still active to perform intrusion detection. For these reasons, a higher accuracy in the predictive task is still important to pursue, since it can lead to the identification of complex attacks that would not be detected by simpler feature extraction techniques. Such attacks could have a significant negative impact on the organizations targeted by attackers. Considering the adoption of techniques with a higher computational cost can also be mitigated by designing parallel or high-performance computing implementations[23,24].

In conclusion, even if the results presented in this study are not vast enough to demonstrate the superiority of the proposed method on a broad scale, they are meant to show the potential of word embeddings to extract a new representation for network traces that can be used to carry out intrusion detection tasks accurately. Feature extraction based on word embedding models requires a higher computational time than simpler techniques, but leads to a higher accuracy, which is important for the identification of complex attacks. In future work, we aim to perform an extensive evaluation with different learning scenarios and machine learning algorithms. We also aim to study in detail word embedding representations and understand how to enforce them with more sophisticated data processing steps.

DECLARATIONS

Authors' contributions

Methodology, data acquisition, implementation, redaction of manuscript and analysis of experimental results: Corizzo R
Methodology, implementation and experiments: Zdravevski E
Implementation of feature extraction prototypes: Russell M, Vagliano A
Hypothesis formulation, methodology, data acquisition and analysis of experimental results: Japkowicz N

Availability of data and materials

Datasets are publicly available at the references reported in the Results section.

Financial support and sponsorship

We acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) through the project "Lifelong Streaming Anomaly Detection" (Grant No. A19-0131-003).

Conflicts of interest

All authors declared that there are no conflicts of interest.

Ethical approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

REFERENCES

1. Axelsson S. Intrusion detection systems: a survey and taxonomy. Tech Rep 2000:99.
2. Bivens A, Palagiri C, Smith R, Szymanski B, Embrechts M. Network-based intrusion detection using neural networks. Intell Eng Syst Artif Neural Netw 2002;12:579-84.
3. Zhang J, Zulkernine M, Haque A. Random-forests-based network intrusion detection systems. IEEE Trans Syst Man Cybern C 2008;38:649-59.
4. Kruegel C, Toth T. Using decision trees to improve signature based intrusion detection. International Workshop on Recent Advances in

Intrusion Detection; 2003 Aug 8-10. Berlin: Springer; 2003. pp. 173-91.

5. Xie M, Hu J. Evaluating host-based anomaly detection systems: a preliminary analysis of ADFA-LD. 6th International Congress on Image and Signal Processing (CISP). Hangzhou, China; 2013. pp. 1711-6.
6. Xie M, Hu J, Yu X, Chang E. Evaluating host-based anomaly detection systems: Application of the frequency-based algorithms to ADFA-LD. International Conference on Network and System Security. Springer, Cham; 2015. pp. 542-9.
7. Aghaei E, Serpen G. Ensemble classifier for misuse detection using N-gram feature vectors through operating system call traces. Int J Hybrid Intell Syst 2017;14:141-54.
8. Ahmim A, Derdour M, Ferrag MA. An intrusion detection system based on combining probability predictions of a tree of classifiers. Int J Commun Syst 2018;31:e3547.1-17.
9. Wunderlich S, Ring M, Landes D, Hotho A. Comparison of system call representations for intrusion detection. International Joint Conference: 12th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2019) and 10th International Conference on EUropean Transnational Education (ICEUTE 2019); 2019. Seville, Spain; 2019. pp. 14-24.
10. Hofmeyr SA, Forrest S, Somayaji A. Intrusion detection using sequences of system calls. J Comput Secur 1998;6:151-80.
11. Lippmann R. DARPA Intrusion Detection Data Sets. Available from: https://www.ll.mit.edu/r-d/datasets. [Last accessed on 31 Jul 2020]
12. Hettich S, Bay S. KDD Cup 1999 Dataset. Available from: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. [Last accessed on 31 Jul 2020]
13. Amiri F, Yousefi MR, Lucas C, Shakery A, Shakery A. Mutual information-based feature selection for intrusion detection systems. J Netw Comput Appl 2011;34:1184-99.
14. Brahmi H, Brahmi I, Ben Yahia SB. OMC-IDS: at the cross-roads of OLAP mining and intrusion detection. Advances in Knowledge Discovery and Data Mining:16th Pacific-Asia Conference; 2012 May 29-June 1; Kuala Lumpur, Malaysia: Verlag, Springer; 2012. pp. 13-24.
15. Apiletti D, Baralis E, Cerquitelli T, D'Elia V. Characterizing network traffic by means of the NetMine framework. Comput Netw 2009;53:774-89.
16. Bilge L, Balzarotti D, Robertson W, Kirda E, Kruegel C. Disclosure: detecting botnet command and control servers through large-scale netflow analysis leyla. ACSAC '12: Proceedings of the 28th Annual Computer Security Applications Conference; 2012 Dec. Orlando, Florida, USA; 2012. pp. 129-38.
17. Hu J. The ADFA intrusion detection datasets. Available from: https://www.unsw.adfa.edu.au/australian-centre-for-cyber-security/cybersecurity/ADFA-IDS-Datasets/. [Last accessed on 31 Jul 2020]
18. Creech G, Hu J. Generation of a new IDS test dataset: time to retire the KDD collection. 2013 IEEE Wireless Communications and Networking Conference (WCNC); 2002. Shanghai, China; 2013. pp. 4487-92.
19. Haider W, Hu J, Slay J, Turnbull BP, Xie Y. Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling. J Netw Comput Appl 2017;87:185-92.
20. Li YF, Gao Y, Ayoade G, Tao H, Khan L, et al. Multistream classification for cyber threat data with heterogeneous feature space. The World Wide Web Conference, 2019 May. San Francisco, USA; 2019. pp. 2992-8.
21. Liu Z, Japkowicz N, Wang R, Cai Y, Tang D, et al. A statistical pattern based feature extraction method on system call traces for anomaly detection. Inform Software Tech 2020;126:106348.
22. Kang DK, Fuller D, Honavar V. Learning classifiers for misuse and anomaly detection using a bag of system calls representation. Proceedings from the 6th Annual IEEE System, Man and Cybernetics Information Assurance Workshop; 2005 June 15th-17th. West Point, NY, USA; 2005. pp. 118-25.
23. Corizzo R, Ceci M, Japkowicz N. Anomaly detection and repair for accurate predictions in geo-distributed big data. Big Data Res 2019;16:18-35.
24. Corizzo R, Ceci M, Zdravevski E, Japkowicz N. Scalable auto-encoders for gravitational waves detection from time series data. Expert Syst Appl 2020;151:113378.
25. Mikolov T, Sutskever I, Chen K, Corrado G, Dean J. Distributed representations of words and phrases and their compositionality. NIPS'13: Proceedings of the 26th International Conference on Neural Information Processing Systems; 2013 Dec. Lake Tahoe, USA; 2013. pp. 3111-9. arXiv1310.4546 [Preprint]. October 16, 2013. Available from: https://arxiv.org/abs/1310.4546. [Last accessed on 31 Jul 2020]
26. Globerson A, Gal C, Fernando P, Naftali T. Euclidean embedding of co-occurrence data. J Mach Learn Res 2007;8:2265-95.
27. Levy O, Goldberg Y. Neural word embedding as implicit matrix factorization. NIPS'14: Proceedings of the 27th International Conference on Neural Information Processing Systems; 2014 Dec. Montreal, Canada; 2014. pp. 2177-85.
28. Lavelli A, Sebastiani F, Zanoli R. Distributional term representations: an experimental comparison. CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management; 2004 Nov. Washington D.C, USA; pp. 615-24.
29. Firth JR. A synopsis of linguistic theory, 1930-1955. Studies in Linguistics Analysis. Oxford: Philological Society; 1957. pp. 1-32.
30. Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv 1310.3781 [Preprint] September 7, 2013. Available from: https://arxiv.org/abs/1301.3781. [Last accessed on 31 Jul 2020]
31. Jang B, Kim I, Kim JW. Word2vec convolutional neural networks for classification of news articles and tweets. PLoS One 2019;14:e0220976.
32. Wu C, Gao R, Zhang Y, De Marinis Y. PTPD: predicting therapeutic peptides by deep learning and word2vec. BMC Bioinformatics 2019;20:456.
33. Chen T, Mao Q, Lv M, Cheng H, Li Y. DroidVecDeep: android malware detection based on word2vec and deep belief network. TIIS

2019;13:2180-97.
34. Baek JW, Chung KY. Multimedia recommendation using word2vec-based social relationship mining. Multimed Tools Appl 2020:1-17.
35. Singhal A. Modern information retrieval: a brief overview. IEEE Data Eng Bull 2001;24:35-43.
36. Branco P, Torgo L, Ribeiro RP. A survey of predictive modeling on imbalanced domains. ACM Computing Surveys 2016;49:1-50.