Hindawi
Security and Communication Networks
Volume 2019, Article ID 2317976, 12 pages
https://doi.org/10.1155/2019/2317976

Research Article

Evaluation of Deep Learning Methods Efficiency for Malicious and Benign System Calls Classification on the AWSCTD

Dainius Čeponis and Nikolaj Goranin

Department of Information Systems, Vilnius Gediminas Technical University, Saulėtekio al. 11, LT-10223, Vilnius, Lithuania

Correspondence should be addressed to Dainius Čeponis; dainius.ceponis@vgtu.lt

Received 1 April 2019; Revised 24 July 2019; Accepted 16 August 2019; Published 11 November 2019

Guest Editor: Hyoungshick Kim

The increasing amount of malware and cyberattacks on a host level increases the need for a reliable anomaly-based host IDS (HIDS) that would be able to deal with zero-day attacks and would ensure low false alarm rate (FAR), which is critical for the detection of such activity. Deep learning methods such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are considered to be highly suitable for solving data-driven security solutions. Therefore, it is necessary to perform the comparative analysis of such methods in order to evaluate their efficiency in attack classification as well as their ability to distinguish malicious and benign activity. In this article, we present the results achieved with the AWSCTD (attack-caused Windows OS system calls traces dataset), which can be considered as the most exhaustive set of host-level anomalies at the moment, including 112.56 million system calls from 12110 executable malware samples and 3145 benign software samples with 16.3 million system calls. The best results were obtained with CNNs with up to 90.0% accuracy for family classification and 95.0% accuracy for malicious/benign determination. RNNs demonstrated slightly inferior results. Furthermore, CNN tuning via an increase in the number of layers should make them practically applicable for host-level anomaly detection.

1. Introduction

The best method to detect unsanctioned or malicious usage of a company's system is to use an intrusion detection system (IDS). IDSs are classified into two main types: network-based IDS (NIDS) and host-based IDS (HIDS) [1]. NIDSs work on the network level and are capable of detecting any malicious activity that can be observed on a company's local network. The HIDSs monitor the activities on end-user machines. They can collect and analyze information such as machine parameters (CPU and RAM usage), modified files, modified registry items (Windows operating system), system calls, etc. While research on NIDS has reached a relatively advanced level and a number of anomaly-based solutions are available on the market [2], HIDSs are stuck in signature-based or file-integrity monitoring approach, making them immune to zero-day attacks [3]. The importance of HIDS becomes critical [4, 5] and requires development of anomaly-based solutions. The earlier approaches based on threshold parameters were not successful because of high false alarm rate (FAR), but recent advances in deep learning (DL) techniques demonstrate the potential of convolutional neural networks (CNNs) and recurrent neural networks (RNNs) in activity classification. Therefore, comparative analysis of such methods is necessary to evaluate their efficiency in malicious activity classification and ability to distinguish malicious and benign activity [6]. The evaluation of these methods allows the selection of the most appropriate and accurate method for anomaly-based HIDS development. Accuracy plays a crucial role because the high false-positives rate would lead to distrust in the system and ignorance of alerts.

Many researchers use machine learning (ML) methods to achieve proper IDS accuracy in detecting intrusive actions. Training and testing data are required to apply ML methods. Most of the recent research was conducted with the old datasets generated in 1998-1999 [7, 8] named DARPA and KDD Cup 99, respectively. Overall, 42% and 20% of the researchers used DARPA dataset and KDD Cup 99, respectively [9]. Both databases focused on NIDS-related data and lacked the information required to train HIDS-suitable methods.

Some attempts [10] have been made to fulfill the growing need for Windows-oriented HIDS datasets. According to statcounter.com, Windows operating systems were still used by more than 70% of the desktop users in 2018 (see Figure 1) [11].

One of the latest HIDS-related datasets for Windows OS is ADFA-IDS [12]. It has a collection of system calls produced on Linux and Windows operating systems. However, ADFA family datasets only have minimal data required for intrusion detection, as they contain only system call identification—system dynamic link library (dll) file name and the called function name. Even the authors of the ADFA-IDS agree that the dataset is incomplete: only basic information was collected, and an insufficient number of vulnerabilities were used to generate malicious activity [13].

We have previously generated AWCSTD to fulfill the demand for a more extended dataset for Windows operating system [12]. It uses more malware samples (12110) and collects more system calls sequences (112.56 million) than any similar public dataset. Most importantly, it also contains 16.3 million systems calls generated by 3145 benign software samples at present. The same method has been used to generate benign system calls sequences as the one previously used in [12] for malignant ones. In total, six virtual machines with Windows 7 operating system preinstalled were utilized. The virtual machines had tools such as Notepad++, 7zip, and data logging tools installed. This allows using the dataset not only for training neural networks on classifying malicious activities but also for training them in distinguishing between malicious and legal activity in general.

In this paper, we present the results of efficiency evaluation for RNNs and CNNs achieved with AWSCTD with different initial data parameters. The remainder of the paper is organized as follows: the Introduction gives a general view on the importance of datasets for HIDS training and the need for evaluating the efficiency of different DL methods; the Related Work section presents results achieved by other research teams; the Materials and Methods part of the article describes the dataset preparation and methods used; the Results and Discussion present the results with comments on their reasons and applicability; Conclusions summarize the results and define future work.

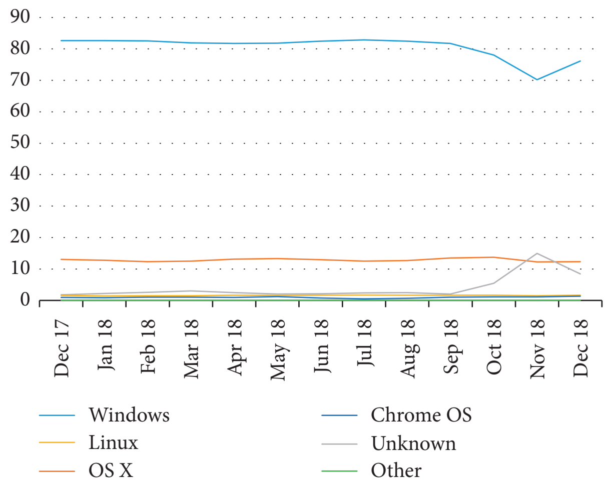


FIGURE 1: Desktop operating systems market share in 2018.

1.1. Related Work.

Attackers often use various techniques to transform and hide malicious activity from the signature-based IDS [14]. Anomaly-based techniques are used to tackle such problems. They have not only introduced a better detection rate of unknown attacks but also increased the number of false-positives [15] because of primitive approaches applied. Advances in deep learning methods combined with extensive training datasets are required to build a benign behavior profile and decrease the FAR.

Several ML classification and clustering methods such as neural networks, support vector machines, k-means clustering, k-nearest neighbors, and decision trees [16–18] have been used to improve anomaly-based IDS. The authors of ADFA-WD (Windows-based) have achieved a 72% detection rate with Naïve Bayes method, and the data were based on transforming system call traces into frequency vectors [13]. Later, the authors of [18] achieved a 61.2% detection rate with ADFA-LD (Linux-based) when CNNs were used. In [19], the authors claim 86.4% accuracy in malicious activity classification with the help of the hybrid neural network, but the dataset was not published, thereby offering no chance to check the accuracy of the results. Similar classification was performed by us on AWSCTD using SVM, and an accuracy of 92.4% was achieved [17]. In addition, the tested decision trees method, which is lighter in terms of training and testing times, has shown comparable results of 92.1% accuracy. This is an essential point, as fast model training is a critical factor in cybersecurity, where new attack samples are introduced very often. Hence, the results in [13, 18] demonstrate very high FAR, whereas the results in [17, 19] do not solve the malicious/benign classification task.

Since 2006, deep-structured learning, commonly called deep learning or hierarchical learning, has emerged as a new area of machine learning research [20]. The architecture of deep neural networks is based on many layers of neural networks (NNs). Artificial NNs (ANNs) were naturally developed from biological neural networks. The first paper on neural networks was produced in 1943; professors McCulloch and Pitts published a paper titled “A Logical Calculus of the Ideas Immanent in Nervous Activity” that logically explained the human neural network and conceptualized the ANNs for the first time in history [21]. An artificial neural network consists of a group of processing elements that are interconnected and convert a set of inputs to a set of preferred outputs. The result of the transformation is determined by the characteristics of the elements and the weights associated with the interconnections among them. The network can adapt to the desired outputs by modifying the connections between the nodes [22]. A fundamental property of neural networks is the concept of programming by example. A large number of weights makes it difficult to fix them and obtain the desired result. Instead, the network is programmed by example and repetition. It is trained by presenting input-output pairs repeatedly. Each time an input is presented, the network guesses the output. The output part of the input-output pair is used to determine whether the network is right or wrong. If wrong, the network is corrected by a learning algorithm using a gradient method on the output error to modify the weights. After each modification, the network gets closer to the desired transfer function as represented by the sample base [23].

The most significant disadvantage of applying neural networks to intrusion detection is the "black box" nature of the neural network. The "Black Box Problem" has overwhelmed neural networks in many applications [24]. This Black Box neural networks feature does not allow the researchers to clearly see and analyze learning results. This also makes the network tinkering process more difficult—researchers cannot analyze and modify networks for better outcomes. DNNs and new hardware capabilities have revived the current state of the ANN research. Two powerful DNN designs were introduced: convolutional neural network (CNN) and recurrent neural network (RNN). CNNs, or ConvNet, are mainly applied for the image recognition tasks because they can scale adequately on large images. They are based on several convolutions and pooling layers combinations that lead to the last, simple ANN layer for final classification. The usage of convolution and pooling layers allows reduction of the feature maps size [25, 26]. RNN is capable of working with time series-based data. The development of RNNs began in 1997 when long short-term memory (LSTM) networks were introduced [27]. The natural capability to accept and work with sequences has allowed to show outstanding performance in speech recognition and machine translation [28]. In 2014, the gated recurrent unit (GRU) was introduced for RNN [29]. It is similar to LSTM but has fewer parameters because it lacks an output gate. GRU is mainly applied in natural language analysis and translation.

Primary DNN applicability for anomaly detection still concentrates on NIDS and the use of KDD dataset [30–32]. The most popular method is RNN with LSTM that provides up to 96% accuracy. However, CNN has also demonstrated applicability for such tasks [19]. This encouraged us to evaluate both CNN and RNN with AWSCTD for anomaly detection (malicious/benign classification) on the end-user machine level.

2. Materials and Methods

2.1. Dataset. For our experiment, AWSCTD containing system calls sequences from Windows OS was used [12]. It was generated using publicly available malware files from Virus Share [33] and publicly available information about any malware found from Virus Total [34]. Later, the collected database was updated with the additional information provided by the Virus Total that included scan results and behavioral information.

For experiments described in this article, AWSCTD was appended with 16.3 million system calls generated by a set of 3145 benign applications (samples were taken from Virus Share and carefully filtered to contain only samples with zero detection rate). The system call collection method for the benign application was the same as for malware system call collection described in [12]. This was done to train CNNs and RNNs for malicious/benign activity classification. It is expected that the number of benign applications with related system calls will increase in the future.

The disproportion of system calls versus the number of applications in case of malicious (16.3/3145) and benign programs (112.56/12110) can be explained by the fact of more "aggressive" and "active" malware behavior compared with legal applications.

2.2. Feature Processing. The data generated by the malware and benign samples were stored in an SQLite database. The system calls sequences were stored in the format provided by Dr. Memory DrSTrace tool (see Figure 2). To evaluate the influence of a number of system calls on classification/detection rate, eight files in csv format were generated with 10, 20, 40, 60, 80, 100, 200, 400, 600, 800, and 1000 of the first system calls in every line in a file, respectively (see Figure 3).

Every system call was assigned with the unique number—a sequence of these numbers represents a system calls sequence produced by the specific malware or benign sample. A special tool was developed by the authors to extract the required number of system calls from the SQLite database.

System calls by benign applications were added to two sets:

(1) Set of six classes (five malicious and one benign)—to be used in the classification accuracy test, i.e., assigning the activity to legal or to one of the five classes of malware programs.
(2) Set of two classes (malware and benign)—to be used in the anomaly detection test, i.e., determining if the activity is malicious or not.

For both of these sets, additional subsets were generated, in which sequences of repeated system calls longer than 3 were replaced with a maximum of 2 repeated system calls (e.g., sequence "4655532" was transformed into "465532") according to the recommendation in [19].

Consequently, 66 sets (files) in total for training and testing were generated. The labelling of sets is presented in Table 1. It is necessary to mention that sets with removed repeated sequences had fewer samples. The main reason is that almost all system calls were identical (e.g., one of the malware samples contained only calls to NtCreateFile).

Datasets AllMalware and AllMalware2 were selected to test if there is any difference in the accuracy as compared with simple ML methods performed earlier [17]. Commercially applicable accuracy of 92.4% was achieved with the Support Vector Machines method, and comparable accuracy of 92.1%. was achieved by the decision tree method.

In this research, deep learning methods were applied to check if they can provide higher accuracy using the same datasets.

Five malware families were selected from AWSCTD for our experiment, each family with at least 100 samples of

```
NtQueryValueKey
     arg 0: 0x158 (type=HANDLE, size=0x4)
     arg 1: 16/18 "Category" (type=UNICODE_STRING*, size=0x4)
     arg 2: 0x2 (type=int, size=0x4)
     arg 3: 0x02c5f094 (type=<struct>*, size=0x4)
     arg 4: 0x90 (type=unsigned int, size=0x4)
     arg 5: 0x02c5f070 (type=unsigned int*, size=0x4)
    succeeded =>
     arg 3: <NYI> (type=<struct>*, size=0x4)
     arg 5: 0x02c5f070 => 0x10 (type=unsigned int*, size=0x4)
     retval: 0x0 (type=NTSTATUS, size=0x4)
```

Figure 2: DrsTrace generated system call sample.

```
7, 10, 10, 9, 3, 3, 3, 9, 10, 10, AdWare
20, 20, 20, 10, 10, 9, 9, 9, 18, 11, Trojan
39, 39, 9, 9, 44, 45, 2, 15, 32, 32, WebToolbar
10, 20, 48, 9, 9, 9, 36, 11, 11, 11, Downloader
9, 18, 21, 22, 9, 9, 26, 9, 18, 23, DangerousObject
10, 40, 26, 26, 29, 29, 13, 9, 16, 41, Clean
```

Figure 3: CSV file sample with all possible classes.

Table 1: Training and testing sets labelling.

Set label	Sequence variations	Comments
AllMalware	10, 20, 40, 60, 80, 100, 200, 400, 600, 800, 1000	Only malware samples
AllMalware2	10, 20, 40, 60, 80, 100, 200, 400, 600, 800, 1000	Only malware samples with no more than two identical sequences in repetition
AllMalwarePlusClean	10, 20, 40, 60, 80, 100, 200, 400, 600, 800, 1000	Malware samples plus and benign samples as additional class
AllMalwarePlusClean2	10, 20, 40, 60, 80, 100, 200, 400, 600, 800, 1000	Malware samples and benign samples as an additional class with no more than two identical sequences in repetition
MalwarePlusClean	10, 20, 40, 60, 80, 100, 200, 400, 600, 800, 1000	Only two classes to train: malware and benign
MalwarePlusClean2	10, 20, 40, 60, 80, 100, 200, 400, 600, 800, 1000	Only two classes to train: malware and benign samples with no more than two identical sequences in repetition

unique family representatives. A family descriptor provided by Kaspersky was used. Table 2 shows the number of unique samples in each training set (family names according to Kaspersky).

SQLite database-based data were converted into easily readable CSV files. The sample data of 10 system calls long sequences file is presented in Figure 3.

The first ten numbers represent a unique system call number followed by the label for the malware family name or legal program ("Benign" label) if it is a benign sample.

Table 2: Data samples count by class.

Class label	Samples count
Trojan	1755
AdWare	4333
WebToolbar	618
Downloader	710
DangerousObject*	105
Benign	2350
Total	9871

*DangerousObject is a malicious software that was detected by KL Cloud Technologies but was not classified exactly.

2.3. Machine Learning Models Used. The experiment was designed and executed on *Keras* [35], and *Tensorflow* [36] library was used as the backend. The following hardware was used in the experiment environment:

(i) CPU: Intel(R) Core (TM) i5-3570 3.80 GHz (4 Cores, 4 Threads)
(ii) GPU: GTX 1070 (1920 Cuda Cores)
(iii) RAM: 16 GB (DDR3)
(iv) OS: Ubuntu 18.04

RNN and CNN methods were used in the experiment. The configuration for the ten system calls can be seen in Figure 4. RNN configuration with LSTM and GRU had three layers: *Input*, *CuDNNLSTM* or *CuDNNGRU*, and *Dense*. CNN had *Input*, *Convolution1D* (with sliding window value of 6), *GlobalMaxPooling1D*, and *Dense* layers. The SVM model was also used to compare the results with previous research [17].

Despite relatively straightforward configuration, the models achieved more than 90% classification accuracy with almost all data samples. The classifiers were trained and

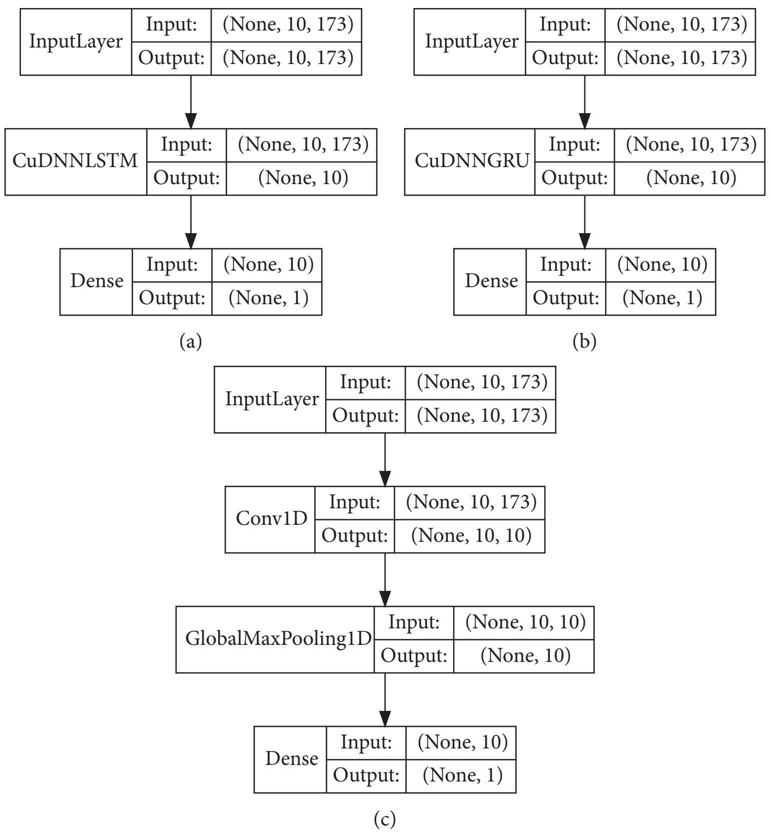


FIGURE 4: Configuration of training (a) LSTM, (b) GRU, and (c) CNN models for MalwarePlusClean data sample.

tested using a 5-fold cross-validation technique. Cross-validation is a technique for evaluating predictive models by partitioning the original sample into a training set to train the model and a test set to evaluate it. The callback of *EarlyStopping* was used to stop the training process when it did not improve for six epochs. Furthermore, we used *one-hot encoding* to provide data for the training models. Ten system calls samples had unique 173 system calls (see Figure 4 for the value of input shape dimensions). Larger data samples had more unique system calls: for example, 400 had unique 488 system calls. In comparison, the data samples of Kolosnjaji et al. [19] had only 60 unique system calls, which means that our dataset is more diverse.

3. Results and Discussion

This section will cover the results of the following tests: malware classification task, family classification task, and anomaly detection task.

3.1. Results of Malware Classification Task. As stated earlier, the results achieved with DL methods were compared with those achieved through classical ML methods in [17]. The data labelled AllMalware were used in that test. Although the original results [17] have shown that SVM method can achieve 92.4% accuracy with the 100 first malicious system calls sequences, it can be seen that DL methods demonstrate significantly better results with the same dataset (see Table 3). The accuracy is calculated as follows: the method of machine learning is trained with a portion of the dataset (80%), whilst another portion of the dataset is used for testing with the trained model, i.e., data used for testing had not been used for training. Therefore, the percentage of correctly classified records is defined as the accuracy.

CNN achieved 92.8% accuracy on the same length of 100 sequence calls. It has also shown better accuracy for 200, 400, 600, 800, and 1000 system calls sequences than SVM (92.7% vs. 89.6%, 93.0% vs. 87.3%, 93.1% vs. 86.1%, 93.0% vs. 84.7%, and 93.1% vs. 83.2%, respectively). This implies that a practically applicable accuracy (>90%) can be maintained even with larger datasets by applying CNN.

Sequence-based DL methods (LSTM and GRU) demonstrated worse results than SVM—the achieved accuracy was equal to 88.1% and 88.3% on the first ten system calls, respectively. CNN not only demonstrated better accuracy but also achieved a somewhat similar classification time compared with the much simpler SVM model. Similar times were maintained even with a larger data sample. In terms of accuracy, SVM demonstrates degrading results in comparison with CNNs.

3.2. Results of Family Classification Task. Benign samples were introduced next in the training process. As described earlier, two sets were used in tests: with repetitive system calls (AllMalwarePlusClean) and without repetitive calls (AllMalwarePlusClean2). The introduction of a new family

Table 3: Malware-type classification with the help of DL and SVM methods (AllMalware set).

Count	Accuracy (percent)				Classification time (seconds)			
	LSTM	GRU	CNN	SVM	LSTM	GRU	CNN	SVM
10	88.1	88.3	87.3	89.4	0.0000926	0.0000840	0.0000401	0.0000440
20	88.8	88.1	89.0	89.4	0.0001043	0.0000994	0.0000510	0.0000583
40	89.1	90.6	91.2	91.6	0.0001327	0.0001377	0.0000618	0.0000842
60	88.2	90.5	91.2	91.9	0.0001786	0.0001704	0.0000890	0.0000848
80	91.8	91.6	92.3	92.7	0.0002194	0.0002221	0.0001157	0.0000937
100	91.6	91.9	92.8	92.4	0.0002559	0.0002566	0.0001290	0.0001165
200	90.4	91.5	92.7	89.6	0.0004440	0.0004363	0.0003019	0.0002392
400	87.6	90.3	93.0	87.3	0.0009840	0.0008142	0.0006222	0.0006739
600	87.4	91.4	93.1	86.1	0.0023443	0.0023052	0.0016681	0.0015564
800	82.1	88.5	93.0	84.7	0.0043159	0.0037894	0.0023929	0.0022612
1000	75.5	89.6	93.1	83.2	0.0068075	0.0056159	0.0033276	0.0032245

to the training set resulted in a decrease in the accuracy (see Table 4).

The removal of repetitive system calls increased the accuracy of the results. The best accuracy of 93.9% was achieved with CNN and 1000 of the first system calls (AllMalwarePlusClean2 data sample). However, a relatively similar outcome of 93.5% was obtained with only 600 of systems calls, which required much less time for training. As results for 600 and 1000 system calls differ only in the error rate, it can be said that a set of 600 system calls is more preferable for practical applications. On smaller sets, the results by CNN were low (86.9%) but still higher than those by LSTN and GRU (85.8% and 85.6%, respectively).

Figure 5 presents the family classification task results by family in case of a set of 100 system calls.

It can be clearly seen that the number of samples in the training data has a huge impact on the correct classification. WebToolbar, Downloader, and DangerousObject labelled samples have more incorrect label assignments than AdWare, Benign, and Trojan. The lowest classification score has a DangerousObject class—zero. That outcome was expected because Kaspersky itself is not sure about the label, and in our prior research, even the best performing SVM model also generated zero correct classification results for this class [17]. Both models of GRU and CNN classified this family as belonging to the Trojan class. Even CNN model, which generates the best performance (90.0% for that specific data collection), shows that DangerousObject class should be labelled as Trojan.

3.3. Results of the Intrusion Detection Test.

Finally, the intrusion detection test was performed, i.e., the applicability of DL methods for determining if an activity is malicious or benign was evaluated (see Table 5). All malicious system calls were merged into one family, and the second family contained only benign system calls. As in previous case, sets both with repetitive system calls (MalwarePlusClean) and with removed repetitive system calls (MalwarePlusClean2) were used.

In this case, the set without repetitive system calls produced comparable results with the full set. This implies that the system calls minimization technique is effective and can be used to achieve better accuracy in family classification and intrusion detection tasks whilst minimizing the model training time.

Accuracies of 94.5%, 94.8%, and 99.3% were obtained by CNN for the 100, 400, and 1000 first system calls, respectively (MalwarePlusClean2). CNN has also shown the best results for all data samples in the two-class classification task (i.e., intrusion detection) of all ML methods used: usable accuracy of 93.2% was obtained even for the 20 first system calls.

In the two-class confusion matrices (see Figure 6), it can be seen that fewer Malware samples are assigned to Benign by CNN as compared with GRU results.

This characteristic is essential in the target field; malignant actions classification as benign must be minimal for the IDS. Benign samples decision is somewhat comparable for the GRU and CNN models.

GRU and CNN models demonstrate outstanding results in the means of the receiver operating characteristic curve (ROC) and area under the curve (AUC) [37]. In Figures 7 and 8, the ROC diagrams with the combination of the AUC values are represented for the MalwarePlusClean samples with the 100 system calls. ROC and AUC are displayed for every fold. Mean ROC and AUC are represented with the blue line.

The best mean AUC value of 0.98 is generated by the CNN model for both classes, i.e., there is a 98% chance that the model will be able to distinguish between Malware class and Benign class. The comparable result of 0.97 is achieved by GRU. High AUC value indicates that both models (GRU and CNN) have good class separation capacity.

3.4. Evaluation of System Call Sequence Size on the Model Training Time and the Number of Epochs Needed to Reach the Saturation.

The evaluation of system call sequence size on the model training time was performed on the AllMalwarePlusClean set. Figure 9 presents the training time for LSTN, GRU, and CNN with sequences of 10, 100, 200, and 400 system calls, respectively. It can be seen that the increase in sequence length results in the exponential increase of training time, making extremely long sequences not applicable for everyday use.

GRU training time was equal to 57.7 minutes with the sequence of 400 system calls. The best performing CNN

Table 4: All malware plus clean samples accuracy results. The total of six classes was classified.

Count	AllMalwarePlusClean				AllMalwarePlusClean2			
	LSTM	GRU	CNN	SVM	LSTM	GRU	CNN	SVM
10	77.2	78.3	78.7	79.9	80.1	79.4	80.1	81.2
20	83.0	82.3	83.4	83.6	85.8	85.6	86.9	86.7
40	84.1	84.1	85.1	85.3	87.3	87.6	88.8	88.0
60	85.0	85.8	86.1	86.2	87.5	86.4	88.5	88.2
80	87.1	86.8	88.5	87.7	87.5	88.0	89.2	87.9
100	87.1	86.5	88.2	87.4	87.7	87.9	88.8	87.6
200	85.8	87.5	89.4	86.5	86.2	87.3	89.6	86.3
400	80.9	88.1	89.9	81.9	80.3	87.3	89.6	78.1
600	79.8	89.1	93.2	75.5	77.6	89.4	93.5	74.8
800	68.3	85.3	93.5	73.9	41.5	92.3	93.6	73.3
1000	77.1	89.1	93.6	72.9	64.0	84.1	93.9	72.3

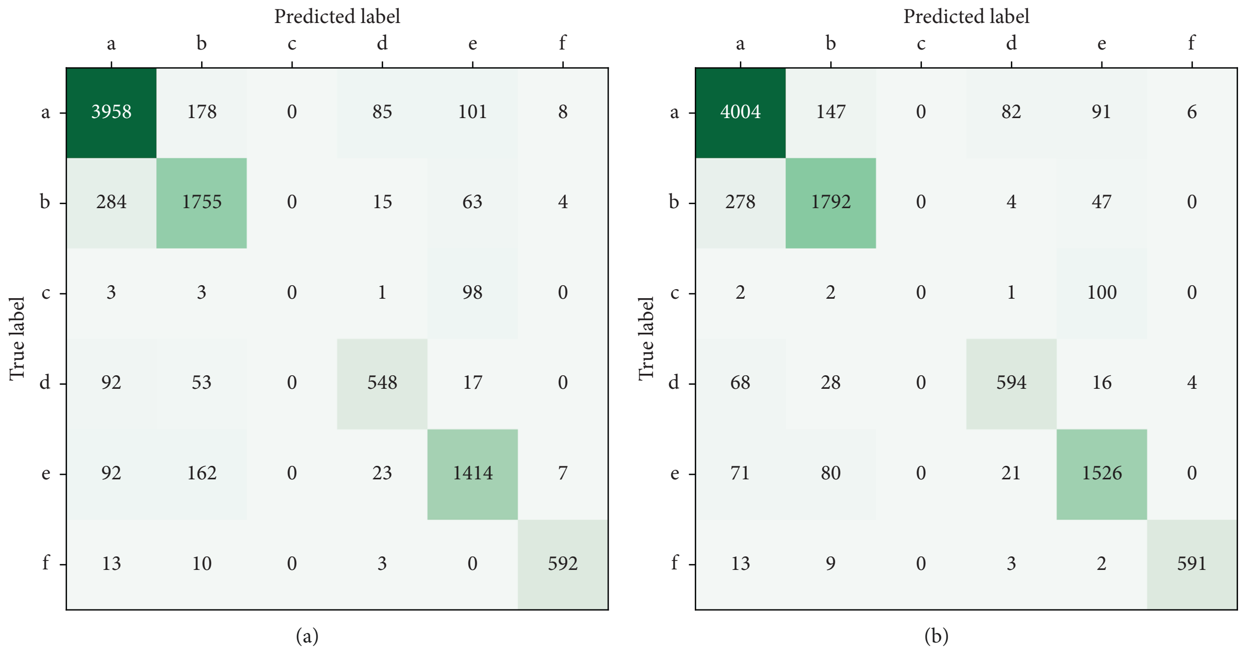


Figure 5: Confusion matrix of the GRU and CNN methods for the 100 system calls sequence of the AllMalwarePlusClean data sample. Class labels: AdWare (a); Benign (b); DangerousObject (c); Downloader (d); Trojan (e); WebToolbar (f).

Table 5: Malicious and benign samples (malware or clean) classification accuracy.

Count	MalwarePlusClean				MalwarePlusClean2			
	LSTM	GRU	CNN	SVM	LSTM	GRU	CNN	SVM
10	87.0	87.4	87.4	87.4	88.6	88.6	88.9	88.4
20	90.0	90.2	91.1	90.2	92.0	92.4	93.0	91.8
40	91.2	90.6	91.8	90.5	92.9	92.4	94.1	92.0
60	91.2	91.1	93.3	91.1	93.2	93.0	93.5	91.9
80	92.7	92.0	94.2	91.5	93.0	93.2	94.5	91.7
100	92.7	92.5	94.3	91.5	93.0	92.1	94.5	91.2
200	90.7	90.8	94.3	88.6	92.4	91.8	94.9	88.9
400	87.0	90.9	94.8	88.4	89.2	92.0	94.8	88.7
600	86.4	88.7	98.5	87.2	85.8	90.1	98.6	87.1
800	84.2	84.7	98.8	86.9	85.1	82.8	98.9	87.0
1000	84.2	70.3	98.9	87.2	83.8	72.4	99.3	87.0

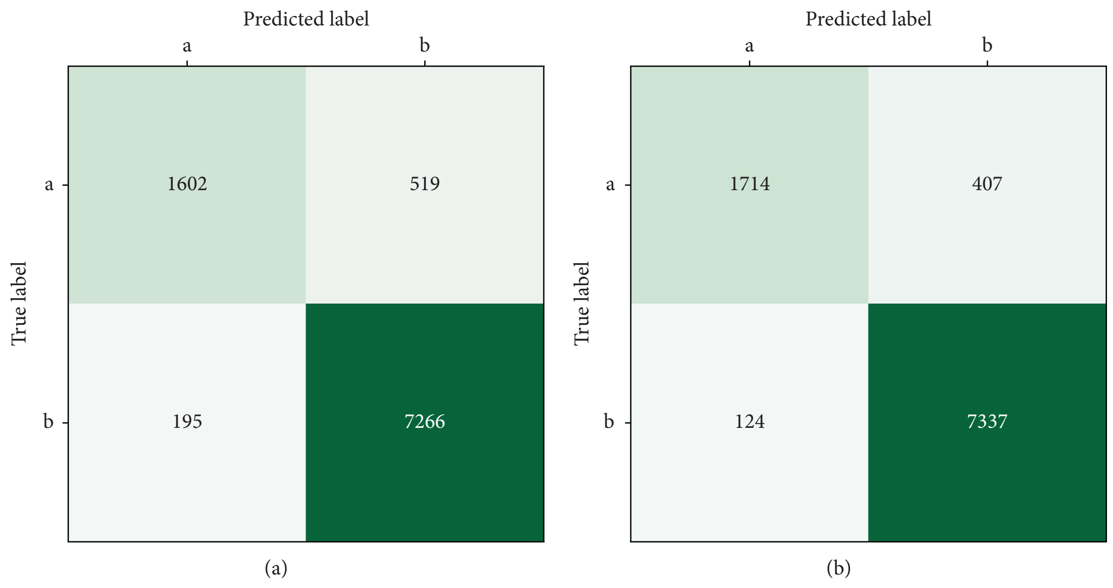


Figure 6: Confusion matrix of the GRU and CNN methods for the 100 system calls sequence of the MalwarePlusClean data sample. Class labels: benign (a); malware (b).

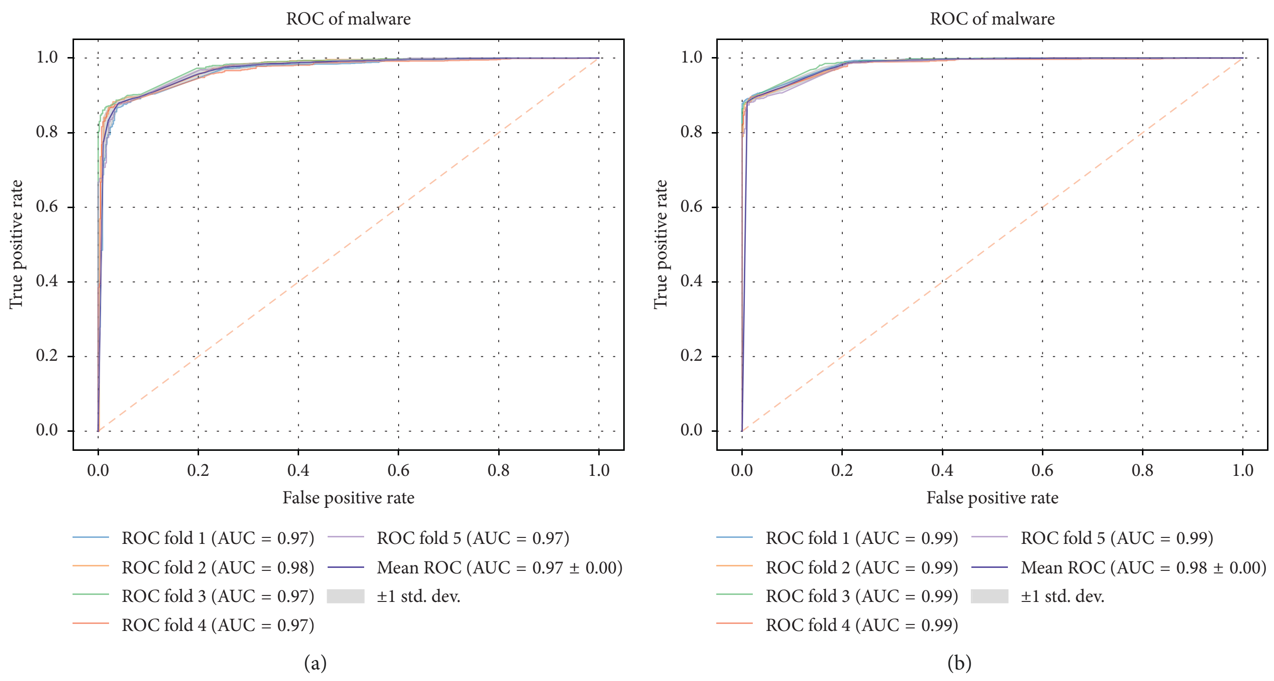


Figure 7: ROC diagrams of the malware class for the 100 system calls of MalwarePlusClean data. (a) GRU. (b) CNN.

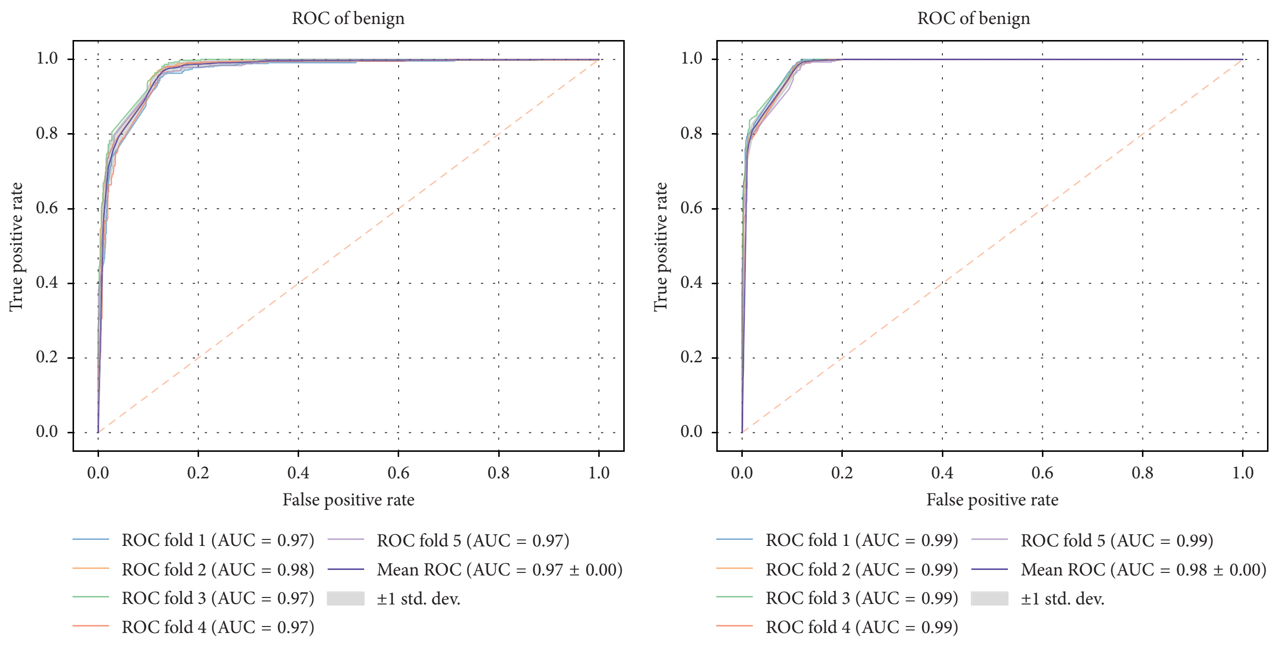
model training took 29.6 minutes with the same dataset. In comparison, 100 system calls sequences training time is much faster. For GRU and CNN, it took 4.6 and 3.9 minutes, respectively.

The evaluation of data model size impact on training time leads to the conclusion that using the first 100 system calls sequences is an optimal solution in terms of time and accuracy balance.

The classification accuracy vs. the number of epochs needed to reach the saturation measurement is presented in Figure 10.

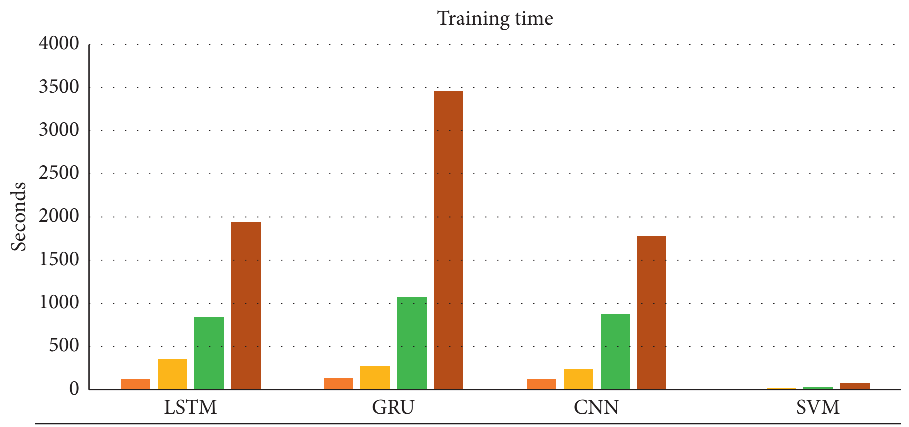
As it can be seen, there is a reverse dependency of the number of epochs before saturation on the system call sequence length, e.g., for the top-performing CNN model, 75 epochs are required to train 10 system calls, and only 30 epochs are required for 400 system calls.

The computed equal error rate (EER) [18] values for the DL methods (LSTM, GRU, and CNN) can be seen in Table 6. When comparing models, lower EER means better performance of the model. In our case, CNN shows best values of 9.7% and 4.8% for the Benign and Malware classes, respectively.



(a) (b)

Figure 8: ROC diagrams of the benign class for the 100 system calls of MalwarePlusClean data. (a) GRU. (b) CNN.



	LSTM	GRU	CNN	SVM
🟧 10	122.32	137.17	126.69	5.31
🟨 100	346.50	277.37	238.05	16.18
🟩 200	836.68	1075.62	876.75	32.87
🟫 400	1942.37	3462.47	1776.72	78.12

Figure 9: Training time comparison of the AllMalwarePlusClean data collection. 10, 100, 200, and 400 sequence calls as data points.

4. Conclusions

A comparative analysis of DL methods, including LSTN, GRU, and CNN was performed in order to evaluate their efficiency for attack classification as well as their ability to distinguish malicious and benign activity. The analysis was performed on an exhaustive AWSCTD, which includes 112.56 million system calls from 12110 executable malware samples and 3145 benign software samples with 16.3 million system calls. The application of such a set increases the classification and intrusion identification accuracy even with vanilla models by 13–38%, compared with the results achieved by other researchers. Furthermore, model tuning should decrease the FAR even more. In general, the achieved accuracy of over 90% allows the application of DL techniques in hybrid or enterprise-oriented security solutions that combine automatic detection of major part of anomalies, leaving unclear cases for human-expert analysis.

All three LSTM, GRU, and CNN models have reached higher than 90% accuracy while solving a malware classification task with a sequence of 80 system calls. All three models generated improved results over simple NN and SVM models on larger data samples, while the latter demonstrates considerably better training times. The best results were obtained with CNNs with up to 90.0% accuracy while performing family classification task and 99.3% rate while solving intrusion

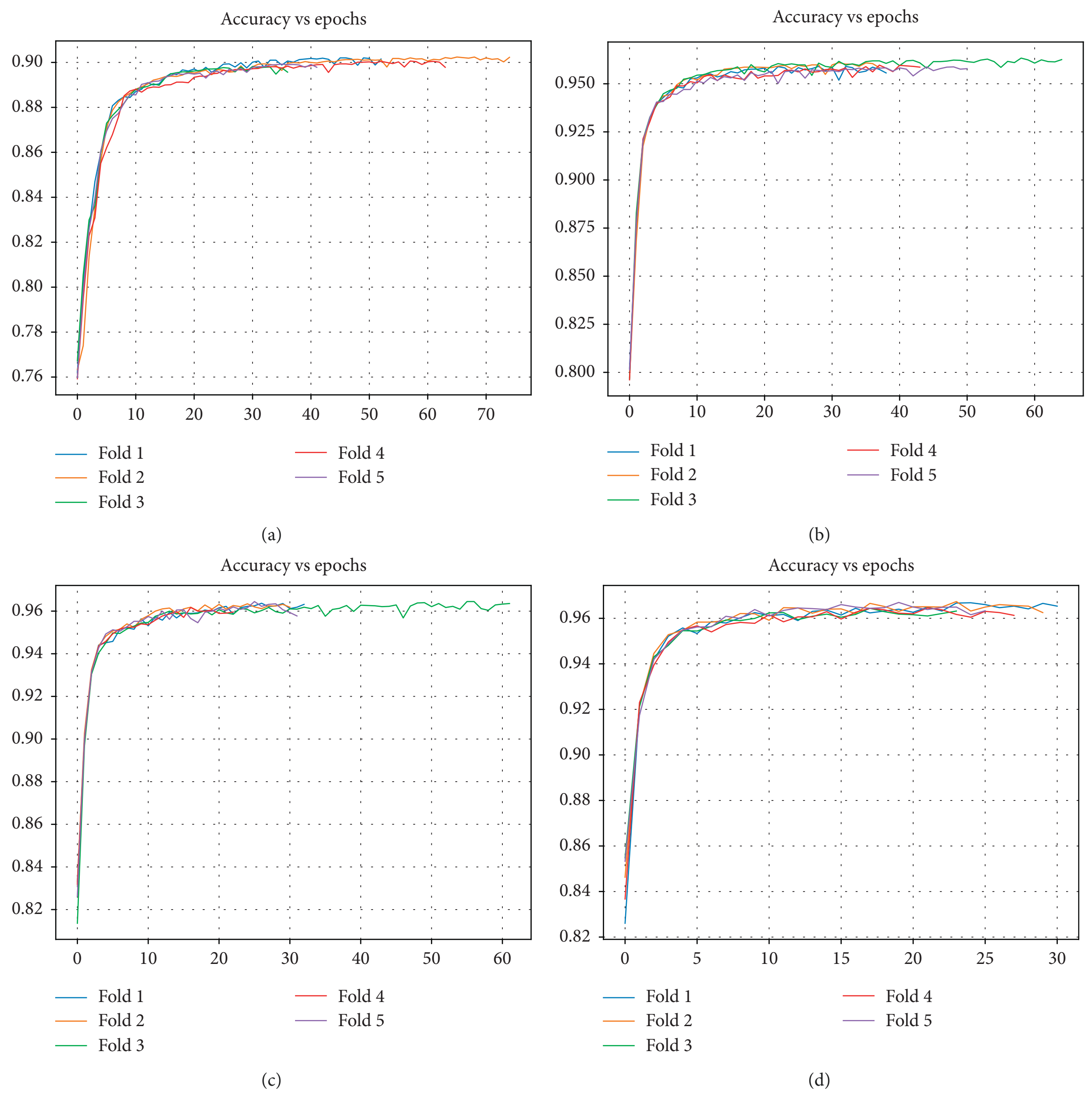


Figure 10: Accuracy vs. epochs needed to reach the saturation while training for the intrusion detection task. Compared CNN models for the 10, 100, 200, and 400 system calls sequences. (a) CNN 10. (b) CNN 100. (c) CNN 200. (d) CNN 400.

Table 6: EER values in percent of the DL methods generated for the MalwarePlusClean of 100 system calls dataset.

DL method	Benign	Malware
LSTM	10.9	7.5
GRU	11.0	8.0
CNN	9.7	4.8

detection task. CNN outperforms sequence-based LSTM and GRU models in all the cases. CNN also shows the best results as compared with EER values of DL methods used. A system calls minimization technique, when repetitive system calls were removed, had a positive influence on all results.

The increase of sequence length resulted in an exponential increase of the model training time, making extremely long sequences not applicable for everyday use. One of the best performing CNN models training took 29.6 minutes, which can be explained by a limited amount of resources on the machine used for the experiments. A reverse dependency of the number of epochs before saturation on the system call sequence length was determined, e.g., for the top-performing CNN model, 75 epochs are required to train 10 system calls and only 30 epochs for the 400 system calls.

Data Availability

The data used to support the findings of this study will be available from the corresponding author upon request after six months from paper publication.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: methods, systems and tools," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 303–336, 2014.

[2] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: techniques, systems and challenges," *Computers & Security*, vol. 28, no. 1-2, pp. 18–28, 2009.

[3] J. Hu, "Host-based anomaly intrusion detection," in *Handbook of Information and Communication Security*, pp. 235–255, Springer, Berlin, Germany, 2010.

[4] R. Bace and P. Mell, *NIST Special Publication on Intrusion Detection Systems*, NIST, Gaithersburg, MD, USA, 2001.

[5] A. Hay, D. Cid, R. Bary, and S. Northcutt, *OSSEC Host-Based Intrusion Detection Guide*, Syngress, Burlington MA, USA, 2008.

[6] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[7] R. P. Lippmann, D. J. Fried, I. Graf et al., "Evaluating intrusion detection systems without attacking your friends: the 1998 DARPA intrusion detection evaluation," in *Proceedings of the DARPA Information Survivability Conference and Exposition. DISCEX'00*, vol. 2, pp. 12–26, Hilton Head, SC, USA, January 2000.

[8] T. Brugger, "KDD cup'99 dataset (network intrusion) considered harmful," *KDnuggets Newsletter*, vol. 7, no. 18, p. 15, 2007.

[9] C. Azad and V. K. Jha, "Data mining in intrusion detection: a comparative study of methods, types and data sets," *International Journal of Information Technology and Computer Science*, vol. 5, no. 8, pp. 75–90, 2013.

[10] K. Berlin, D. Slater, and J. Saxe, "Malicious behavior detection using windows audit logs," in *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security—AISec '15*, pp. 35–44, Denver, CO, USA, October 2015.

[11] StatCounter Global Stats, "Desktop operating system market share worldwide," January 2019, http://gs.statcounter.com/os-market-share/desktop/worldwide/2018.

[12] D. Čeponis and N. Goranin, "Towards a robust method of dataset generation of malicious activity for anomaly-based HIDS training and presentation of AWSCTD dataset," *Baltic Journal of Modern Computing*, vol. 6, no. 3, 2018.

[13] W. Haider, G. Creech, Y. Xie, and J. Hu, "Windows based data sets for evaluation of robustness of host based intrusion detection systems (IDS) to zero-day and stealth attacks," *Future Internet*, vol. 8, no. 3, p. 29, 2016.

[14] M. Xie and J. Hu, "Evaluating host-based anomaly detection systems: a preliminary analysis of ADFA-LD," in *Proceedings of the 2013 6th International Congress on Image and Signal Processing (CISP)*, vol. 3, pp. 1711–1716, Hangzhou, China, December 2013.

[15] M. A. Aydin, A. H. Zaim, and K. G. Ceylan, "A hybrid intrusion detection system design for computer network security," *Computers and Electrical Engineering*, vol. 35, no. 3, pp. 517–526, 2009.

[16] S. Agrawal and J. Agrawal, "Survey on anomaly detection using data mining techniques," *Procedia Computer Science*, vol. 60, pp. 708–713, 2015.

[17] N. Goranin and D. Čeponis, "Investigation of AWSCTD dataset applicability for malware type classification," *International Scientific Journals Security & Future*, vol. 2, no. 4, pp. 83–86, 2018.

[18] N. N. Tran, R. Sarker, and J. Hu, "An approach for host-based intrusion detection system design using convolutional neural network," in *Mobile Networks and Management*, pp. 116–126, Springer, Cham, Switzerland, 2018.

[19] B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep learning for classification of malware system call sequences," in *AI 2016: Advances in Artificial Intelligence, LNAI*, vol. 9992, pp. 137–149, Springer, Cham, Switzerland, 2016.

[20] L. Deng and D. Yu, "Deep learning: methods and applications," *Foundations and Trends® in Signal Processing*, vol. 7, no. 3-4, pp. 197–387, 2013.

[21] J. Kim, N. Shin, S. Y. Jo, and S. H. Kim, "Method of intrusion detection using deep neural network," in *Proceedings of the 2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pp. 313–316, Jeju, Korea, February 2017.

[22] S. Pervez, I. Ahmad, A. Akram, and S. U. Swati, "A comparative analysis of artificial neural network technologies in intrusion detection systems," in *Proceedings of the 6th WSEAS International Conference on Multimedia, Internet & Video Technologies*, pp. 84–89, Lisbon, Portugal, September 2015.

[23] H. Debar and B. Dorizzi, "An application of a recurrent network to an intrusion detection system," in *Proceedings of the IJCNN International Joint Conference on Neural Networks*, vol. 2, pp. 478–483, Baltimore, MD, USA, June 1992.

[24] I. Ahmad, A. B. Abdullah, A. S. Alghamdi, N. A. Baykara, and N. E. Mastorakis, "Artificial neural network approaches to intrusion detection: a review," in *Proceedings of the 8th Wseas International Conference on Telecommunications and Informatics (TELE-INFO'09)*, pp. 200–205, World Scientific and Engineering Academy and Society (WSEAS), Istanbul, Turkey, 2009.

[25] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," pp. 1–28, 2016, https://arxiv.org/abs/1603.07285.

[26] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[28] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., pp. 3104–3112, Curran Associates, Inc., Red Hook, NY, USA, 2014.

[29] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: encoder-decoder approaches," in *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, Doha, Qatar, October 2014.

[30] J. J. Kim, J. J. Kim, H. L. T. Thu, and H. Kim, “Long short term memory recurrent neural network classifier for intrusion detection,” in *Proceedings of the 2016 International Conference on Platform Technology and Service (PlatCon)*, pp. 1–5, Jeju, Korea, February 2016.
[31] M. Z. Alom, V. Bontupalli, and T. M. Taha, “Intrusion detection using deep belief networks,” in *Proceedings of the 2015 National Aerospace and Electronics Conference (NAECON)*, pp. 339–344, Dayton, OH, USA, March 2016.
[32] Z. Wang, “Deep learning-based intrusion detection with adversaries,” *IEEE Access*, vol. 6, pp. 38367–38384, 2018.
[33] VirusShare.com, December 2017, https://virusshare.com/.
[34] VirusTotal, V. T., December 2017, https://virustotal.com/.
[35] F. Chollet, “Keras,” 2015, http://github.com/fchollet/keras.
[36] M. Abadi, P. Barham, J. Chen et al., “Tensorflow: a system for large-scale machine learning,” in *Proceedings of the OSDI*, vol. 16, pp. 265–283, Savannah, GA, USA, November 2016.
[37] T. Fawcett, “An introduction to ROC analysis Tom,” *IRBM*, vol. 35, no. 6, pp. 299–309, 2005.