JKAU: Comp. IT. Sci., **Vol**. **8** No. 2, pp: 93 – 109 (1440 A.H. / 2019 A.D.)
Doi: 10.4197/Comp. 8-2.7

Enhanced Host-Based Intrusion Detection Using System Call Traces

Yaqoob S. Ikram and **Mohamed A. I. Madkour**

Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia

Jacob.sayid@hotmail.com

Abstract. To detect zero-day attacks in modern systems, several host-based intrusion detection systems are proposed using the newly compiled ADFA-LD dataset. These techniques use the system call traces of the dataset to detect anomalies, but generally they suffer either from high computational cost as in window-based techniques or low detection rate as in frequency-based techniques. To enhance the accuracy and speed, we propose a host-based intrusion detection system based on distinct short sequences extraction from traces of system calls with a novel algorithm to detect anomalies. To the best of our knowledge, the obtained results of the proposed system are superior to all up-to-date published systems in terms of computational cost and learning time. The obtained detection rate is also much higher than almost all compared systems and is very close to the highest result. In particular, the proposed system provides the best combination of high detection rate and very small learning time. The developed prototype achieved 90.48% detection rate, 22.5% false alarm rate, and a learning time of about 30 seconds. This provides high capability to detect zero-day attacks and also makes it flexible to cope with any environmental changes since it can learn quickly and incrementally without the need to rebuild the whole classifier from scratch.

Keywords: Anomaly detection; zero-day attacks; system call traces; machine learning.

1. Introduction

As online services are spreading over the Internet, the security and privacy concerns are arising which make it risky choice for organizations [1]. Generally, the Internet is a fertile field for crackers and hackers to target due to various combinations of attack vectors available such as SQL injection, XSS, IP spoofing, ARP spoofing, DNS poisoning, DOS and DDOS. For example, DDOS attack on Amazon caused BitBucket.org, which is hosted in AWS, to be unavailable for few hours [2]. Utilizing firewall is a good way to protect online services from such attacks. However, relying on a single line of defense is not enough in cases like firewall bypass by insiders or due to a zero-day vulnerability in firewall. Thus, there is a need to have a second line of defense by utilizing accurate and reliable intrusion detection systems (IDS). Recently, huge data breaches of large companies and organizations including Dropbox [3], Yahoo [4] and LinkedIn [5] occurred. These showed up the weakness of employed IDS to detect such attacks and severity of issued attacks.

To mitigate new generation zero-day attacks, researchers proposed several IDS and evaluated these IDSs on newly compiled

dataset (ADFA-LD) that is based on modern systems. Obtained results need to be improved in terms of accuracy and speed of learning normal behavior. Thus, the main problems addressed in this paper are the accuracy and speed of IDS to learn.

The main contribution in this paper is host based anomaly detection using distinct short-sequences extraction. A novel algorithm is utilized to extract only distinct short sequences of system calls per normal trace to create a normal profile. Then, a companion classification algorithm is used to evaluate the IDS. The developed prototype achieved 90.48% detection rate (DR), 22.5% false alarm rate (FAR), and a learning time of about 30 seconds. A second technique is investigated based on anomaly detection using frequency-based feature extraction from traces of system calls. A novel frequency-based technique is used to extract highly representative features from raw dataset and then applied semi-supervised machine learning techniques on these features. The developed prototype achieved 65% detection rate and 26.6% false alarm rate with learning time of few milliseconds. This shows that the proposed feature extraction using distinct short-sequences performs better than the frequency-based technique. The rest of the paper is organized as follows: Section 2 gives background and the related work is given in Section 3. Section 4 covers the details of the proposed solution and Section 5 validates the obtained results. The last section presents conclusion and future work.

2. Background

Anomaly-based IDS has been implemented with various techniques and methods. Each technique has its own advantages and disadvantages. These techniques are classified into statistical anomaly detection, machine learning based anomaly detection and data mining based anomaly detection [6]. In the following subsection, we provide a background about anomaly detection techniques used in system call traces to derive the classification models.

A. Anomaly Detection Techniques in System Call Traces

Two categories of techniques are there to detect anomaly in system call traces [7]: short-sequence-based and frequency-based. The short-sequence-based technique creates a profile of normal behavior by extracting sub sequences from the original logged traces. A sliding window technique is used to extract these sub sequences. To detect abnormal behavior, a significant deviation of test traces from the extracted normal traces triggers the abnormality. Pre-determined threshold decides whether a deviation is significant or not [8]. Short-sequence-based technique considers the positional information of system call traces which results in creating more accurate profile. In learning process, numerous approaches are used such as hidden Markov model (HMM) [9-11], support vector machines (SVM) [12] and artificial neural network (ANN) [13-14]. Generally, the learning process in short-sequence-based technique takes long time [12]. In contrast, frequency-based technique needs shorter learning time. It transforms the normal traces into vector depending on the frequency of system calls in the normal trace which makes it very efficient in terms of computation. A significant drawback of this technique is the low accuracy of detection since it ignores the positional information in the collected traces. Detection models can be derived from the transformed traces through various methods including k-nearest neighbor (kNN) [15-16], k-means clustering [17] and SVM [18] articles should present attractive studies, new advances, and knowledge about a topic of significance. Original Research Article must be supported by the results.

B. Datasets for HIDS Evaluation

In the literature, authors used to design their proposed IDS systems using publicly available benchmark datasets. The use of datasets comes handy because it creates a unified and unbiased baseline for evaluation. Moreover, it saves time for authors to prepare a real-time environment to test and evaluate their IDS. In this subsection, both old and up to date datasets are overviewed.

1. KDD Datasets

Over the past decade, authors had evaluated their IDS using old KDD dataset which was prepared in 1998-2000 [19]. KDD was relevant at the first time of compilation. Over time, it was outdated and no longer proper to be used for validating IDS on modern systems. Researchers used KDD although of its irrelevance because alternatives were not available. The KDD dataset included wide range of data collected under Solaris-based system environment. Among the collected data, system call sequences were available by extracting them from BSM audit data. Forrest [20] introduced the use of these system call traces for detecting intrusions. Later, several improvements are proposed [21, 8, 22, 7]. In addition to aging, the computer technology is highly dynamic and changes rapidly. This resulted in KDD dataset to be irrelevant quickly and impractical [23-24]. Moreover, generalization of results obtained from Solaris-based system is not logical since Solaris-based system has limited usage in cyber community compared to Linux [25-26]. The most critical point in KDD collection is the existence of several data artifacts within it which make it not reliable [27].

2. ADFA Dataset

Given all critiques in KDD datasets, researchers decided to prepare newer datasets that overcome some or all of the issues existed in the old datasets. One attempt was UNM dataset compiled in 2004 [28]. The dataset was, however, focused on processes of only selected programs instead of the whole system [29]. This means the normal profile is going to model the behavior of these programs only. Moreover, UNM dataset is compiled in UNIX based operating system that has fewer deployments in cyber community compared to Linux operating system. Recently, a modern intrusion detection dataset (ADFA) is compiled and made available for developing and evaluating IDS systems [29]. The ADFA dataset is based on modern kernels such as Linux Kernel 2.6.38. For example, Ubuntu is a popular operating system that is based on Linux Kernel and it has the highest deployments on one of the most popular cloud platforms called OpenStack [30]. Consequently, the ADFA datasets can be generalized for testing and evaluating HIDS for modern systems. It's important to note that ADFA datasets target HIDS which comes handy when firewalls are bypassed by attackers. For HIDS system to be improved, evaluated precisely and deployed in modern systems, it needs an accurate baseline of normal behavior in modern systems. Once the HIDS learns this normal profile, it compares the future system behavior against it to detect intrusions. ADFA datasets target two popular host operating systems, namely Windows and Linux. This research will shed light on the Linux dataset ADFA-LD. The dataset was prepared in Ubuntu Linux environment of version 11.04, and this environment was fully patched in the time of dataset compilation. Web services are enabled in the host by installing Apache 2.2.17 [31] along with PHP 5.3.5 [32]. To allow data storage and dynamic content, MySQL 14.14 [33] was installed. FTP and SSH services were started also by their default ports to access the host remotely. In addition, a web application called Tiki Wiki 8.1 [34] was installed and prepared to serve the online users. This version of the web application is chosen intentionally

as it has a critical exploit that allows for remote code execution by injecting PHP code [35]. This whole setup is considered a realistic Linux server that has all these services enabled with some vulnerabilities existed. Several types of attacks are selected carefully and issued against the Linux kernel in order to represent real attempts to hack a well-protected system. ADFA-LD included password brute-force attacks and payload-based attacks. The brute-force attacks are issued to guess FTP or SSH users and passwords. Payload-based attacks are issued to execute arbitrary code on the server remotely. The remote code execution is expressed by different attack types in ADFA-LD due to the highly different methods used to achieve each one. This means the attack patterns are going to be different also. Note that FTP and SSH credentials stealing attempts are highly valid for the attacker as these services are used to be accessible from remote machines for administration purposes. Table 1 summarizes the attacks included in ADFA-LD and their vectors. Note that for payload upload and management, an open source hacking tool called Metasploit [36] is used.

Traces of system call are considered a very accurate data source for anomaly based intrusion detection system [29]. Therefore, it's selected to prepare the ADFA-LD datasets. ADFA-LD consists of three groups. These groups are validation, training and attack. Validation and training groups contain normal system call traces. i.e., traces that had been collected using audit program while the system was being used for normal operations such as document writing with Latex and web browsing. Attack group contains traces that had been collected while attacks were being issued against the host. Note that the collected traces in ADFA-LD training group were configured to filter out any traces outside the range 300 bytes to 6 kilobytes. Additionally, validation traces in the range between 300 bytes and 10 kilobytes were filtered out. Table 2 shows each group of traces in ADFA-LD with number of traces in that group. In validation folder of ADFA-LD dataset, there are 4372 traces each of which in a separate file, followed by an extra file that does not contain a trace. This makes the total number of traces in validation group different than what's depicted by Xie [15]. Table 3 shows detailed distribution of attack traces. Each trace in ADFA-LD consists of system call indices instead of names for simplicity. The unistd.sh file in Linux operating system can be referenced to map the numerical indices to the corresponding system call names.

Table 1. Summary of attacks in ADFA-LD and their vectors.

Attack	Vector
Hydra-FTP [37]	FTP service
Hydra-SSH [37]	SSH Service
Add-Super-User	Poisoned executable through social engineering
Java Meterpreter [36]	Web: crafted payload upload
Linux Meterpreter [36]	Poisoned executable through social engineering
C100 Web shell [38]	Web: crafted payload upload

Table 2. ADFA-LD trace groups.

Trace group	Number of trace files
Training	833
Validation	4372
Attack	746

Table 3. Distribution of attack traces to each type of attack.

Attack Name	Number of trace files
Hydra FTP	162
Hydra SSH	148
Java Meterpreter	125
Meterpreter	75
Web shell	118
Add-user	91

3. Related Work

A. HIDS Using KDD

In KDD era, attacks footprints were clear and focused on single processes rather than being spread on multiple processes as in modern systems. To detect anomalies in system call traces, several HIDS systems were proposed based on KDD which were considered the basis for later researches. For example, statistical analysis of KDD audit trails was employed [39]. They achieved a detection rate (DR) of 90% with high false alarm rate (FAR) of 40%. Yeung and Ding [40] used hidden Markov model (HMM) and entropy analysis of system calls. The best results obtained were 91.7% at DR with low FAR of 10%. Data mining techniques brought good results when applied on KDD. For instance, SVM was applied on subset of KDD and achieved 99.6% at DR with only 4.17% FAR [18]. Moreover, KNN was utilized to classify process traces achieving 96.3% DR and 6.2% FAR [41].

B. HIDS Using ADFA-LD

Since KDD no longer represents modern systems, applying same algorithms for feature extraction and detecting attacks in the new ADFA-LD resulted in a lower performance. Thus, novel approaches are proposed by authors that can detect new generation attacks. One of the breakthroughs in the field of detecting modern zero-day attacks is the semantic approach in feature extraction proposed by Creech and Hu. However, this approach suffers from a very large learning time of few weeks to create a huge dictionary of normal profile from normal traces [12, 42]. Creech and Hu selected SVM and extreme learning machine (ELM) as predictors that accept the transformed dataset as input for learning. When applied on ADFA-LD, 90% DR is achieved and 12.5% FAR using extreme learning machine (ELM) as predictor. With one-class SVM, the semantic approach for feature extraction achieved a performance of 80% DR and 17.5% FAR. To avoid the long learning time of the semantic approach in feature extraction, Xie *et al.* [15, 17] proposed frequency-based technique as an alternative to short-sequence-based for feature extraction by converting traces into multi-dimensional frequency vectors of equal size. Using KNN classification, only 60% DR was achieved at 30% FAR whereas KMC classification achieved a little bit higher DR and lower FAR in comparison. Moreover, Xie *et al.* [43] proposed another novel technique for HIDS development which utilizes short-sequence-based technique efficiently to extract features and SVM for classification. They achieved 70% DR with 25% FAR. The main drawback in the proposed methods that utilize frequency-based techniques is the inability to extract accurate features that can bring clear differences between attack traces and normal traces. A novel approach similar to frequency-based technique is proposed by Haider *et al.* [44] in which statistical features are extracted from system call traces to form normal profile. The results obtained were promising by achieving 78% DR and 21% FAR using KNN for classification. However, the main drawback in the approach is that the proposed statistical feature extraction algorithm has randomization steps which result in different outputs each time extraction is done on the same input of data. This leads to creating an instable profile that leads to random detection rates which is not reliable.

4. Design and Implementation

The present paper considers the new benchmark ADFA-LD dataset for developing an anomaly-based HIDS using frequency-based and short-sequence-based techniques. ADFA-LD dataset is compiled in Linux operating system environment. Like any typical operating systems, Linux OS has

running processes during operation time. These processes utilize various resources of computer through system calls. ADFA-LD contains system call traces which are sequences of system calls called by various processes during their execution time. A system call in ADFA-LD trace is represented as a number instead of the name for simplicity. Each number refers to a unique system call in Linux kernel 2.6.38. Figure 1 represents portion of a trace (UTD-0002) from training set in ADFA-LD. To learn the normal behavior, the HIDS proposed in this paper must be trained and evaluated at first using a set of normal traces. Next it should be ready to decide whether a new trace is normal or abnormal. Direct matching of traces obviously does not work well. Therefore, two categories of techniques are used for feature extraction as in the literature; short-sequence-based and frequency-based. To have an accurate HIDS, the extracted features should contain enough information that can be useful for differentiating normal traces from abnormal ones. Moreover, the detection technique should be precise enough to detect abnormalities. Along with HIDS accuracy, the time of learning normal profile, evaluation speed and resource consumption are all important characteristics of a good HIDS. All these characteristics will be taken into consideration for the proposed enhanced HIDS.

54	175	120	175	175	3	...	140

Fig. 1. A trace representation in ADFA-LD

A. Overall Design

Figure 2 shows how the proposed HIDS works. Basically, any HIDS consists of three main components; the data source, which is ADFA-LD dataset in this context, the feature extraction component and the classification algorithm. The data source in the proposed HIDS consists of both training and testing data. In the training phase, the classification algorithm takes as input training part of ADFA-LD which contains normal behavior. The output of this phase is a detection model. This model takes the testing part of ADFA-LD as input which consists of both normal and abnormal behavior. Based on the detection model, evaluation phase decides whether the input data, which is a system call trace, is abnormal or not. If abnormal an alarm will be raised. Note that in the proposed HIDS design, system call traces are first transformed before going as input to classification algorithm using feature extraction component. Feature extraction is very crucial component for HIDS that is based on system call traces. It extracts informative and representative features from the raw data. If the extracted features are irrelevant, then the classification algorithms would not give good results. In this paper, feature extraction techniques of two categories are proposed, namely frequency-based extraction and short-sequence-based extraction.

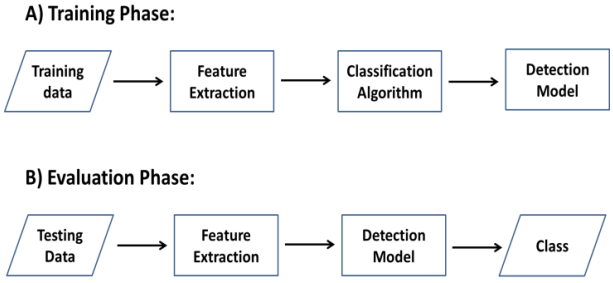

Fig. 2. Overall Architecture of the proposed HIDS.

B. Short-Sequence-Based Feature Extraction

In short-sequence-based technique, the trace is divided into short sequences based on a window size. One advantage of this technique is that it preserves the positional information of system calls and theoretically decreases the possibility of mimicry attacks on the IDS [45]. There are several ways to take the advantage of this technique. In this paper, a novel algorithm that is inspired from [20] and [8] is used for extracting sequences of system calls in a trace and determining the abnormality of any trace. The algorithm takes as input the trace and

produces sets of distinct or unique sequences of size "w" for that trace. The parameter "w" determines number of system calls in an extracted sequence. Applying the extraction technique to all normal traces, a database of normal profile is built which contains distinct sequences for each normal trace. To understand the extraction algorithm better, Fig. 3 elaborates on the flow of the algorithm steps. Suppose a normal trace is recorded and it consists of the following system calls separated by spaces: (54 175 120 175 175 3 175 175 3). Let the window size w=3. The extraction output is going to be: {(54 175 120), (175 120 175), (120 175 175), (175 175 3), (175 3 175), (3 175 175)}. Extracting the distinct short sequences can be expressed by equation 1 as follows:

$$F_i(w) = distinctEx(T_i, w) \qquad (1)$$

C. Short-Sequence-Based Classification Algorithms

After creating a database of normal profile using ADFA-LD training data as shown in Algorithm 1, short sequence extraction algorithm is again used on any test trace to extract distinct short sequences. Now to detect anomaly, the classification algorithm that learned the normal profile would scan the test trace as shown in Algorithm 2. The proposed algorithm calculates the similarity of the test trace to each normal trace in normal profile. This is done by direct matching of distinct short sequences of the test trace to short sequences of the normal traces. In case a short sequence in normal trace matched another in test trace, one hit point is retrieved. To calculate the final and overall similarity score, we considered three different methods (A, B, and C) that are variations of Algorithm 2 as shown below. Practical experiments showed that the first method provides the best results.

A. The summation of matching hits of test trace short sequences to all short sequences of normal traces is considered the similarity score as seen in steps of Algorithm 3.

B. The maximum matching hits of test trace short sequences to the short sequences of all 833 normal traces in ADFA-LD training data is retrieved as similarity score.

C. For each normal trace, the total matching hits of test trace short sequences to those in normal trace is divided by the total number of distinct short sequences in the normal trace. Then, the summation of previous calculation for each normal trace is considered the similarity score.

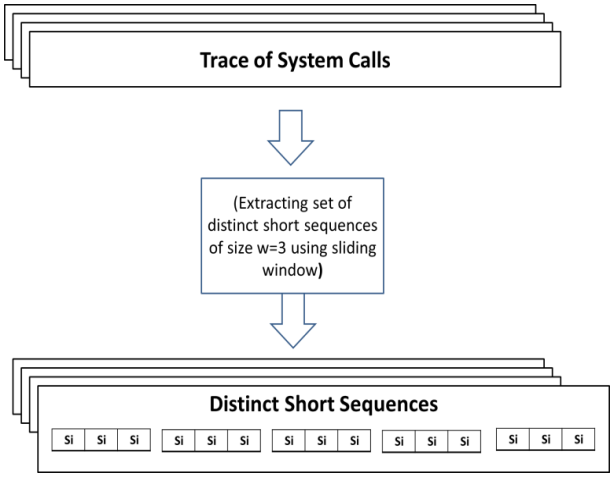

Fig. 3. Architecture of feature extraction component for short-sequence-based technique.

Algorithm 1. Short-Sequence-Based Training.

```
Learn(algorithm,options,train_traces)

For 0 ≤ i < train_traces.length do
  short_sequences[i]= distinctEX(train_traces[i],w)
End for
detection_model=  algorithm.buildModel(short_sequences,
options)
Return detection_model
```

Algorithm 2. Short-Sequence-Based Classification (Abstract).

```
Classify(detection_model,test_trace)

test_sequences= distinctEX(test_trace,w)
Class= detection_model.classify(test_sequences)
Return Class
```

Algorithm 3. Short-Sequence-Based Classification.

```
Classify(detection_model,test_trace)

test_sequences= distinctEX(test_trace,w)
train_sequences= detection_model.train_sequences

For 0 ≤ i < detection_model.train_traces.length do
 For 0 ≤ j < train_sequences[i].length do
   For 0 ≤ k < test_sequences.length do
    if(test_sequences[k]== train_sequences[j]
      similarity[i]++;
   End for
 End for
End for

if(sum(similarity)> detection_model.threeshold)
  Class = 'normal'
else
  class = 'abnormal'
Return Class
```

D. Frequency-Based Feature Extraction

As stated previously, a system call trace consists of system calls issued by a running process. Several kinds of features can be extracted from such data. A simple method is to count calls in the trace which means a trace is expressed by just a single feature. Such simple feature is not informative enough to detect abnormality since the count of calls in normal and abnormal traces is almost similar. In general, learning and classifying in frequency-based feature extraction are done very fast. However, it suffers from low accuracy to detect anomalies. To overcome this issue, a novel feature extraction technique is proposed to get higher detection rate and even faster processing by using limited number of dimensions. In this technique, distinct system call sequences of varying length are counted. The length of system call sequence is set from one to five which results in a vector of five elements as seen in Fig. 4. For example, say a trace is expressed by the call sequence {68, 68, 10, 68, 10, 100}, there are three distinct system call sequences of length one. These are 68, 10 and 100 without repeating 68 or 10. Extracting these distinct sequences is done through a sliding window of size one. Taking two as window size, the distinct sequences for the same trace is expressed by {(68, 68), (68,10), (10,68), (10,100)} which equals to four distinct sequences. The feature vector in the suggested extraction technique consists of five elements resulted by changing window size from one to five. Max window size is five and it is chosen empirically because ineffective results were observed after exceeding this limit. Consequently, any trace is transformed into feature vector of five elements. To better understand how the frequency-based feature extraction process is done, following definitions are considered. Let:

1. U is set of all distinct system calls in Linux kernel 2.6.38 which is almost 369 system calls based on the unistd.h file from Linux which contains system call numbers.

2. S denotes a system call, $S \in U$.

3. V is set of all system call traces in ADFA-LD.

4. T is trace of system calls $T \in V$. Particularly:

$T = \{Si: i= 1, 2, \ldots, |T|\}$; where $|T|$ is the number of system calls in the trace.

5. C is any subsequence in a trace T; $C \in T$.

6. From any trace Ti the extracted feature vector is Fi: {Fi (w), w=1, …, 5}

7. Distinct (T, w): a function that counts subsequences of size w without repeating.

Based on the previous definitions, equation (2) shows how the five elements are constructed. Combining them, a trace is transformed into fixed size feature vector. The feature vector is used by the classification algorithm to learn a normal profile. Also, any test trace is converted into feature vector before going to detection model to detect if an abnormality exists.

$$F_i(w) = \text{distinct}(T_i, w) \quad (2)$$

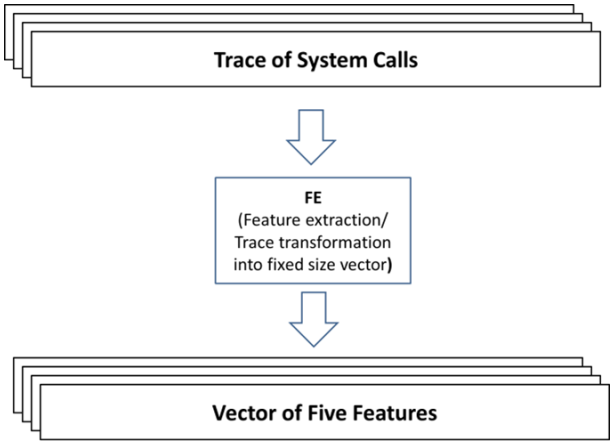

Fig. 4. Architecture of feature extraction component for frequency-based technique.

E. Frequency-Based Classification Algorithms

Referring to Fig. 2, the classification algorithm would construct the normal profile using training data available. Any test trace is then evaluated through measuring the deviation from normal profile. Normally, a threshold needs to be specified to decide if the deviation is significant or not. In the proposed HIDS, the best threshold is specified empirically. To build the normal profile in HIDS, several machine learning algorithms are utilized. The proposed frequency-based HIDS applies one-class support vector machine (SVM) [46] with various kernel functions. Moreover, the widely used KNN in literature [47-49] is utilized which measures distances between features using several distance functions. K-furthest neighbors which is inspired from [50] is also applied to construct a normal profile. There are two steps in the used classification algorithm: to learn normal profile selecting one or more features reside in feature vector and then to classify the transformed test trace as normal or abnormal. Steps of the training and classification are shown in Algorithm 4 and Algorithm 5, respectively. In training algorithm, ADFA-LD training set is used, called here: train_traces. A mutually exclusive set of ADFA-LD is used in classification algorithm as a testing set; called here: test_traces. Algorithm 5 will be called for every trace in the test set. After extracting the features from the training set, a machine learning algorithm is used to build the model. This model will be used for classification. Several algorithms will be considered and compared in this research, namely, SVM, KNN, and KFN. These algorithms have various parameters for configuration; called here: options.

Algorithm 4 Frequency-Based Training.

```
Learn(algorithm,options,train_traces)

For 0 ≤ i < train_traces.length do
  short_sequences[i]= distinctEX(train_traces[i],w)
End for
detection_model=  algorithm.buildModel(short_sequences, options)
Return detection_model
```

Algorithm 5 Frequency-Based Classification.

```
Classify(detection_model,test_trace)

For 1 ≤ w ≤ 5 do
  Feature_Vector[w]= distinct(test_trace,w)
End for
Class= detection_model.classify(Feature_Vector)
Return Class
```

In KNN and KFN options, various distance metrics can be used. Chosen metrics in the experiments are Euclidean, Cosine and Manhattan. Table 4 shows the method of calculating each metric where xi represents the ith element of extracted feature vector from normal trace and yi is the ith element of extracted feature vector from test trace. The feature vector contains 5 features. However, we shall consider experimentally various features combinations to decide the feature

combination that provides best obtainable results. In Table 4, "n" represents the number of considered features ranging from 1 to 5.

Table 4. Distance metrics for "n" features.

Euclidean distance	$\sqrt{\sum_{i=1}^{i=n}(x_i - y_i)^2}$
Cosine distance	$1 - \frac{\sum_{i=1}^{i=n} x_i y_i}{\sqrt{\sum_{i=1}^{i=n} x_i^2} \times \sqrt{\sum_{i=1}^{i=n} y_i^2}}$
Manhattan distance	$\sum_{i=1}^{i=n} \lvert x_i - y_i \rvert$

5. Experiments and Results

We have conducted three experiments to evaluate the performance of the proposed HIDS techniques. The developed prototypes are implemented using Java programming language, Weka [51] with Libsvm [52] extension and MySQL. Experiments are conducted using a computer that has two CPU cores running at 2.9 GHz and twelve GB of RAM. Figure 5 to Fig. 9 compare the obtained results with the published results of six top HIDS systems that were mentioned in Section 3. The comparisons are shown in Table 5 and numbered for convenience. The detection rate (DR), false alarm rate (FAR) and learning time are used for evaluation according to [44]. These measures are computed as follows:

- DR is the number of correctly detected abnormal traces divided by the total number of abnormal traces contained in ADFA-LD attack set.
- FAR is the average error rate obtained from false positive rate (FPR) and false negative rate (FNR), where:
  - FPR is the total number of normal traces detected as attack incorrectly divided by the total number of normal traces contained in ADFA-LD validation set.
  - FNR is the complement of DR, namely (1 – DR).
- Learning time is the time required to extract and learn the normal baseline by the classification algorithm.

Figures 5 and 6 compare the obtained results from frequency-based techniques in terms of DR and FAR respectively with the published results that use similar category of techniques. While Fig. 7 and 8 compare the obtained results from short-sequence-based techniques in terms of DR and FAR respectively with the published results that use similar category of techniques. Figure 9 compares the learning time of the proposed novel algorithm with the best published algorithm. In the following we compare our results shown in cases 1, 2 and 3 to the results of the six top HIDS systems shown in Table 5. Although Case 9 outperformed all frequency-based methods, yet it is not compared because it included randomization step which makes it unreliable and would generate different profiles each time the extraction is done on the same training data.

A. Case 1

The results obtained in this case are the best of our three cases and are competent to the best seen results in the literature up to our knowledge. In this experiment, we used a novel short-sequence-based extraction and classification algorithm of system call traces. We studied three variations of this algorithm as explained in Subsection 4.3. The variation A of the proposed classification algorithm performed best compared to the other two variations. In this classification algorithm, the summation of matching hits is taken as the similarity score. In other words, similarity of a test trace to the whole normal profile of the operating system is calculated. Setting the similarity score threshold empirically to 9000 as the least acceptable score gave a detection

rate (DR) of 90.48%, and false alarm rate (FAR) of 22.5%. Compared to all other results of the six top HIDS systems shown in Table 5, this experiment achieved best results in terms of DR and learning time. Compared to case 4 of Table 5 which has the best achieved results in the literature, the peak detection rates are almost the same. However, the proposed algorithm outperformed case 4 in terms of learning time by taking only seconds instead of weeks as shown in Fig. 9. Creech *et al.* [42] who developed the semantic approach stated that the significant burden occurred by the long time required to extract semantic features in learning phase limits HIDS to quickly respond to changes in the normal profile or baseline. Consequently, an HIDS based on semantic approach will be inflexible to eventual environmental changes like installing new services, changing the roles and adding new user accounts. On the other hand, the proposed novel short-sequence-based algorithm does not suffer from any of these limitations and it can respond to changes in the baseline in a matter of seconds. Additionally, it can learn incrementally without the need to rebuild the whole baseline from scratch. However, the quick learning time is at the expense of FAR which is higher in the proposed algorithm in this paper compared to semantic approach that has lower FAR as seen in Fig. 8.

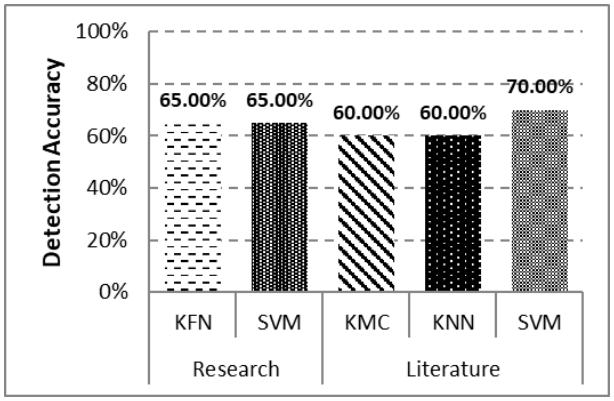

Fig. 5. Comparison of detection rates between frequency-based techniques.

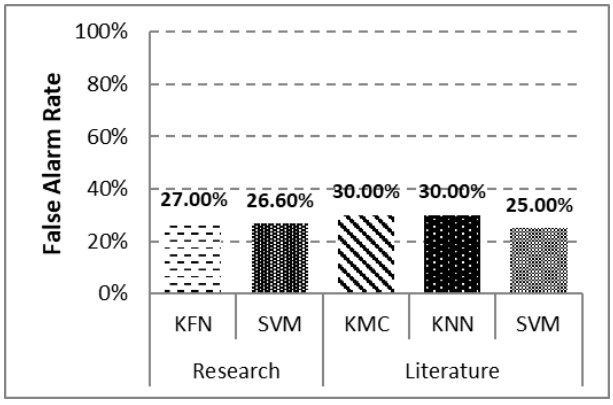

Fig. 6. Comparison of false alarm rates between frequency-based techniques.

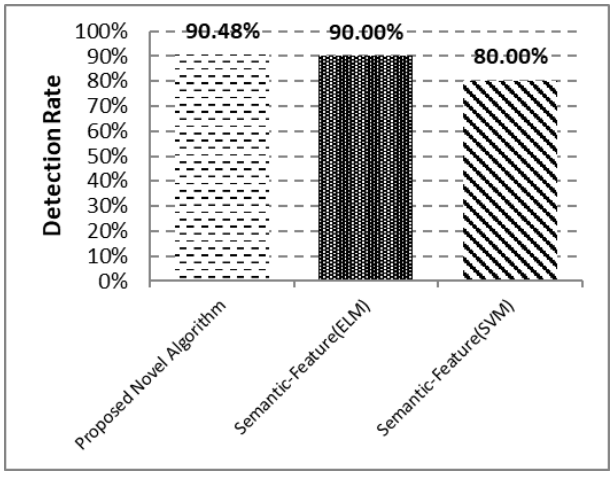

Fig. 7. Comparison of detection rates between short-based techniques.

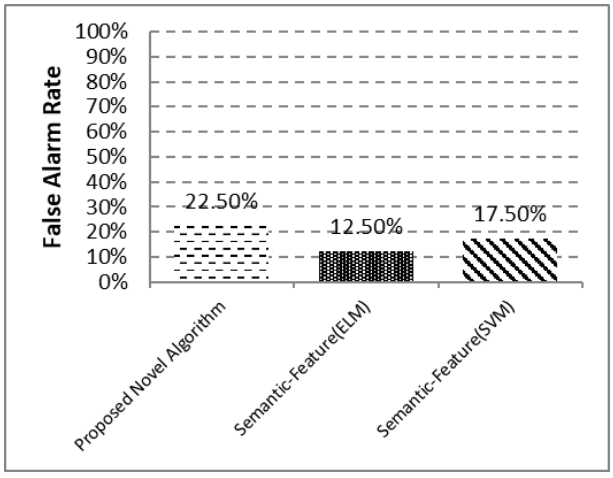

Fig. 8. Comparison of false alarm rates between short-based techniques.

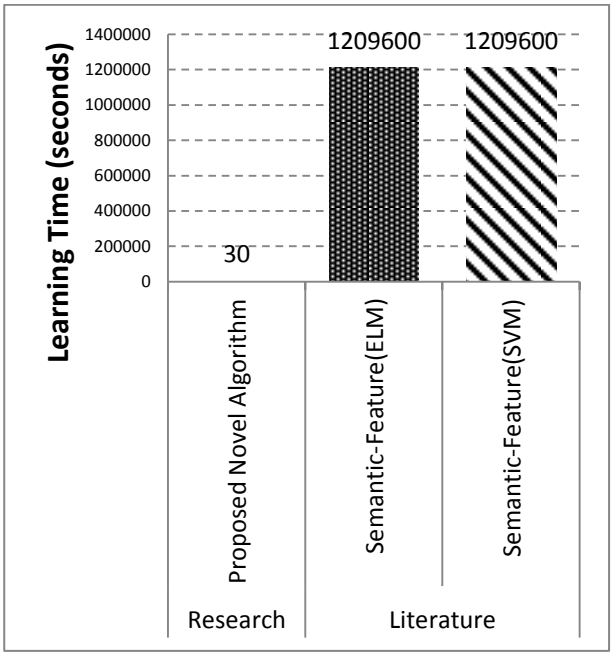

Fig. 9. Comparison of learning times between short-sequence-based techniques. Note that two weeks are assumed for literature to represent "few weeks".

B. Case 2

In this experiment, every trace is transformed into a set of features using frequency-based feature extraction technique. We used one-class SVM as frequency-based classification algorithm and conducted an exhaustive search of all possible combinations of up to five features to evaluate and select the best combination. The best results are obtained by using only a single feature which is the count of unique system calls in a trace. In one-class SVM, linear kernel function is chosen with nu parameter set to 0.2 empirically to control the number of support vectors. The results achieved were 65% DR and 26.6% FAR. Compared to cases 7 and 8 of Table 5 which used KNN and KMC respectively with several frequency-based features, the results of this experiment outperformed in terms of DR and FAR as seen in Fig. 5 and 6. However, case 6 that used SVM with hybrid extraction technique performed better.

The most important finding of this experiment is that the count of unique system calls is a highly informative feature and may give good results if used with other combinations of extracted features in the literature.

Table 5. Comparison of proposed HIDS with HIDS in the Literature Review.

Case	Ref No.	Feature Extraction	Classification Algorithm	DR	FAR	Learn time
1	Proposed	Short-Sequence-Based	Proposed novel algorithm	90.48%	22.5%	s
2	Proposed	Frequency-Based	SVM (Linear function)	65%	26.6%	ms
3	Proposed	Frequency-Based	KFN	65%	27%	ms
4	[12] [42]	Short-Sequence-Based	ELM	90%	12.5%	week
5	[12] [42]	Short-Sequence-Based	SVM	80%	17.5%	week
6	[43]	Frequency-Based with Short Sequences	SVM	70%	25%	s
7	[17] [15]	Frequency-Based	KNN	60%	30%	s
8	[17]	Frequency-Based	KMC	60%	30%	s
9	[44]	Statistical Features	KNN	78%	21%	s

C. Case 3

In this experiment, we used KNN and KFN as frequency-based classification algorithms. KFN performed better than KNN by setting k=1, threshold r=47 and selecting the efficient Manhattan distance metric. As in case 2, the best selection of feature combination included only the count of unique system calls in a trace. Compared to cases 7 and 8 of Table 5, which used KNN and KMC with several frequency-based features, we obtained slightly better DR with lower FAR and rapid learning speed using only a single feature instead of several features. However, the hybrid

extraction technique in case 6 brought better results using SVM.

We can conclude that KFN can bring better results than KNN using frequency-based feature extraction. This needs to be further investigated using extraction techniques from the literature.

6. Conclusions and Future Work

In the present work, we developed a novel host-based IDS that can detect zero-day attacks based on anomaly detection. The main contribution is an anomaly detection algorithm using distinct short-sequences extraction from system call traces. This algorithm utilized a novel method to extract distinct short sequences of system calls per normal trace to create a normal profile. Then, a companion classification algorithm is used to evaluate the IDS. A prototype is developed using Java and MySQL and tested using the ADFA-LD dataset. The best results obtained were detection rate of 90.48% and false alarm rate of 22.5% with learning time of about 30 seconds. To the best of our knowledge, the obtained detection rate is much higher than almost all compared systems and is very close to the highest result. Moreover, the proposed system provides the best combination of high detection rate and very small learning time. This provides high capability to detect zero-day attacks and also makes it flexible to cope with any environmental changes since it can learn quickly and incrementally without the need to rebuild the whole classifier from scratch. Nevertheless, there are still some limitations in the proposed HIDS that need to be considered as future work. The false alarm rate needs to be decreased by improving the extraction and the classification algorithms. Moreover, the abnormality threshold value has to be determined automatically.

Acknowledgment

My thanks and appreciation to Dr. Mohamed Ashraf Madkour for persevering with me as my advisor throughout the time it took me to complete this research paper. As my advisor, Dr. Mohamed provided detailed guidance and encouragement throughout the course of preparing for and conducting the research. I am grateful as well to Dr. Seyed Mohamed Buhari for coordinating and overseeing the administrative concerns. He provided a lot of materials, tools and guidelines used by the research community.

References

[1] **Latif, R., Abbas, H., Assar, S.** and **Ali, Q.** (2014) "Cloud Computing Risk Assessment: A Systematic Literature Review," in: *Future Information Technology*, J. J. (Jong H. Park, I. Stojmenovic, M. Choi, and F. Xhafa, Eds. Springer Berlin Heidelberg, pp: 285-295.

[2] "*Amazon EC2 attack prompts customer support changes*," SearchAWS. [Online]. Available: http://searchaws.techtarget.com/news/1371090/Amazon-EC2-attack-prompts-customer-support-changes. [Accessed: 09-May-2015].

[3] "*Hackers Stole Account Details for Over 60 Million Dropbox Users*," Motherboard. [Online]. Available: http://motherboard.vice.com/read/hackers-stole-over-60-million-dropbox-accounts. [Accessed: 10-Nov-2016].

[4] "*Yahoo data breach affects at least 500 million users, company says*," PCWorld, 22-Sep-2016. [Online]. Available: http://www.pcworld.com/article/3123426/security/yahoo-data-breach-affects-at-least-500-million-users.html. [Accessed: 10-Nov-2016].

[5] "*LinkedIn Lost 167 Million Account Credentials in Data Breach*," Fortune, 18-May-2016. .

[6] **Patcha, A.** and **Park, J.-M.** (2007) "An overview of anomaly detection techniques: Existing solutions and latest technological trends," Comput. Netw., vol. 51(12) Aug.: 3448-3470,

[7] **Forrest, S., Hofmeyr, S.** and **Somayaji, A.** (2008) "The Evolution of System-Call Monitoring," in: *Computer Security Applications Conference, 2008. ACSAC 2008. Annual*, pp. 418-430.

[8] **Hofmeyr, S. A., Forrest, S.** and **Somayaji, A.** (1998) "Intrusion Detection Using Sequences of System Calls," *J. Comput Secur*, **6** (3) Aug.: 151-180,

[9] **Eskin, E.** and **Stolfo, S. J.** (2007) "*System and methods for intrusion detection with dynamic window sizes*," US7162741 B2, 09-Jan-2007.

[10] **Hoang, X. A.** and **Hu, J.** (2004) "An efficient hidden Markov model training scheme for anomaly intrusion detection of server applications based on system calls", **2**: 470-474.

[11] **Hoang, X. D., Hu, J.** and **Bertok, P.** (2009) "A Program-based Anomaly Intrusion Detection Scheme Using Multiple Detection Engines and Fuzzy Inference," *J Netw Comput Appl*, **32**(6) Nov.:1219–1228.

[12] **Creech, G.** and **Hu, J.** (2014) "A Semantic Approach to Host-Based Intrusion Detection Systems Using Contiguous and Discontiguous System Call Patterns", *IEEE Trans. Comput.*, **63**(4) Apr.: 807-819.

[13] **Ghosh,A. K., Wanken, J.** and **Charron, F.** (1998) "Detecting anomalous and unknown intrusions against programs", in: *Computer Security Applications Conference,. Proceedings. 14th Annual, 1998*, pp: 259-267.

[14] **Ghosh, A. K., Schwartzbard, A.** and **Schatz, M.** (1999) "Learning Program Behavior Profiles for Intrusion Detection", in: *Proceedings of the 1st Conference on Workshop on Intrusion Detection and Network Monitoring - Volume 1, Berkeley, CA, USA*, pp: 6-6.

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

[16] **Liao, Y.** and **Vemuri, V. R.** (2002) "Use of K-Nearest Neighbor Classifier for Intrusion Detection", *Comput Secur*, **21**(5) Oct.: 439-448.

[17] **Xie, M., Hu, J., Yu, X.** and **Chang, E.** (2014) "Evaluating Host-Based Anomaly Detection Systems: Application of the Frequency-Based Algorithms to ADFA-LD," in: *Network and System Security*, M. H. Au, B. Carminati, and C.-C. J. Kuo, Eds. Springer International Publishing, pp: 542–549.

[18] **Chen, W.-H., Hsu, S.-H.** and **Shen, H.-P.** (2005) "Application of SVM and ANN for intrusion detection", *Comput. Oper. Res.*, **32**(10) Oct.: 2617–2634.

[19] "*MIT Lincoln Laboratory: DARPA Intrusion Detection Evaluation*." [Online]. Available: http://www.ll.mit.edu/ideval/data/. [Accessed: 14-Sep-2016].

[20] **Forrest, S. Hofmeyr, S. A., Somayaji, A.** and **Longstaff, T. A.** (1996) "A Sense of Self for Unix Processes," in: *Proceedings of the 1996 IEEE Symposium on Security and Privacy, Washington, DC, USA*, p. 120.

[21] **Forrest, S., Hofmeyr, S. A.** and **Somayaji, A.** (1997) "Computer Immunology," *Commun ACM*, **40** (10) Oct.: 88-96.

[22] **Warrender, C., Forrest, S.** and **Pearlmutter, B.** (1999) "Detecting intrusion using system calls: alternative data models," in: *Proceedings of the IEEE Symposium on Security and Privacy*.

[23] **Brown, C., Cowperthwaite, A., Hijazi, A.** and **Somayaji, A.** (2009) "Analysis of the 1999 DARPA/Lincoln Laboratory IDS Evaluation Data with NetADHICT," in: *Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications, Piscataway, NJ, USA*, pp: 67–73.

[24] **Owezarski, P.** (2010) "A Database of Anomalous Traffic for Assessing Profile Based IDS," in: *Traffic Monitoring and Analysis*, F. Ricciato, M. Mellia, and E. Biersack, Eds. Springer Berlin Heidelberg, pp: 59-72.

[25] **McHugh, J.** (2000) "Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations As Performed by Lincoln Laboratory", *ACM Trans Inf Syst Secur*, **3**(4) Nov.: 262-294.

[26] **Vaughan-Nichols, S. J.** (2016) "*Amazon EC2 cloud is made up of almost half-a-million Linux servers*," ZDNet. [Online]. Available: http://www.zdnet.com/article/amazon-ec2-cloud-is-made-up-of-almost-half-a-million-linux-servers/. [Accessed: 18-Sep-2016].

[27] **Engen, V., Vincent, J.** and **Phalp, K.** (2011) "Exploring Discrepancies in Findings Obtained with the KDD Cup '99 Data Set", *Intell Data Anal*, **15**(2) Apr.: 251–276.

[28] "*Computer Immune Systems - Data Sets and Software*" [Online]. Available: http://www.cs.unm.edu/~immsec/systemcalls.htm. [Accessed: 18-Sep-2016].

[29] **Creech, G.** and **Hu, J.** (2013) "Generation of a new IDS test dataset: Time to retire the KDD collection," in: *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, pp: 4487-4492.

[30] "*OpenStack | Cloud | Ubuntu.*" [Online]. Available: http://www.ubuntu.com/cloud/openstack. [Accessed: 15-Sep-2016].

[31] "*Welcome to The Apache Software Foundation!*" [Online]. Available: https://www.apache.org/. [Accessed: 19-Sep-2016].

[32] "*PHP: Hypertext Preprocessor*." [Online]. Available: https://secure.php.net/. [Accessed: 19-Sep-2016].

[33] "*MySQL*." [Online]. Available: https://www.mysql.com/. [Accessed: 19-Sep-2016].

[34] **Community, T.** (2016) "Tiki Wiki CMS Groupware," Tiki Wiki CMS Groupware :: Community. [Online].

Available: https://tiki.org/HomePage. [Accessed: 19-Sep-2016].

[35] **EgiX** (2016) "*snarf_ajax.php Remote PHP Code Injection*." [Online]. Available: https://www.exploit-db.com/exploits/18265/. [Accessed: 19-Sep-2016].

[36] "*Penetration Testing Software, Pen Testing Security*," Metasploit. [Online]. Available: https://www.metasploit.com/. [Accessed: 20-Sep-2016].

[37] "*THC-HYDRA - fast and flexible network login hacker*." [Online]. Available: https://www.thc.org/thc-hydra/. [Accessed: 20-Sep-2016].

[38] "*StopTheHacker.com | Experts Explain: Hidden Backdoors*," StopTheHacker.com. [Online]. Available: http://www.stopthehacker.com/2012/02/07/experts-explain-hidden-backdoors/. [Accessed: 11-Mar-2017].

[39] **Ye, N., Emran, S. M., Chen, Q.** and **Vilbert, S.** (2002) "Multivariate Statistical Analysis of Audit Trails for Host-Based Intrusion Detection," *IEEE Trans Comput*, **51**(7) Jul.: 810-820.

[40] **Yeung, D.-Y.** and **Ding, Y.** (2003) "Host-based intrusion detection using dynamic and static behavioral models", Pattern Recognit., **36**(1) Jan.: 229-243.

[41] **Sharma, A. Pujari, A. K.** and **Paliwal, K. K.** (2007) "Intrusion detection using text processing techniques with a kernel based similarity measure," *Comput. Secur.*, **26** (7–8) Dec.:488–495,

[42] **Creech, G.** (2014) "*Developing a high-accuracy cross platform Host-Based Intrusion Detection System capable of reliably detecting zero-day attacks*", Awarded by: University of New South Wales Engineering & Information Technology,.

[43] **Xie, M., Hu, J.** and **Slay, J.** (2014) "Evaluating host-based anomaly detection systems: Application of the one-class SVM algorithm to ADFA-LD," in: *11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), 2014*, pp: 978-982.

[44] **Haider, W. Hu, J.** and **Xie, M.** (2015) "Towards reliable data feature retrieval and decision engine in host-based anomaly detection systems," in: *IEEE 10th Conference on Industrial Electronics and Applications (ICIEA), 2015*, pp: 513-517.

[45] **Wagner, D.** and **Soto, P.** (2002) "Mimicry Attacks on Host-based Intrusion Detection Systems," in: *Proceedings of the 9th ACM Conference on Computer and Communications Security, New York, NY, USA*, 2002, pp: 255-264.

[46] **Schölkopf, B. Smola, A. J. Williamson, R. C.** and **Bartlett, P. L.** (2000) "New Support Vector Algorithms*", Neural Comput,* **12** (5) May.: 1207-1245,

[47] **Xie, M. Hu, J. Han, S.** and **Chen, H. H.** (2013) "Scalable Hypergrid k-NN-Based Online Anomaly Detection in Wireless Sensor Networks," *IEEE Trans. Parallel Distrib. Syst.*, **24**(8) Aug.: 1661-1670.

[48] **Xie, M., Hu, J.** and **Tian, B.** (2012) "Histogram-Based Online Anomaly Detection in Hierarchical Wireless Sensor Networks," in: *Proceedings of the 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications, Washington, DC, USA*, pp: 751-759.

[49] **Xie, M., Han, S.** and **Tian, B.** (2011) "Highly Efficient Distance-Based Anomaly Detection through Univariate with PCA in Wireless Sensor Networks," in: *2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, pp: 564-571.

[50] **Said, A., Fields, B., Jain, B. J.** and **Albayrak, S.** (2013) "User-centric Evaluation of a K-furthest Neighbor Collaborative Filtering Recommender Algorithm," in: *Proceedings of the 2013 Conference on Computer Supported Cooperative Work, New York, NY, USA*, pp: 1399-1408.

[51] **Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P**. and **Witten, I. H.** (2009) "The WEKA Data Mining Software: An Update," *SIGKDD Explor Newsl*, **11**(1) Nov.: 10-18,.

[52] **Chang, C.-C.** and **Lin, C.-J.** (2011) "LIBSVM: A Library for Support Vector Machines," *ACM Trans Intell Syst Technol*, **2**(3) May.: 27:1-27:27.

نظام كشف التسلل المحسن بتتبع آثار استدعاءات الوظائف الأساسية في نظام التشغيل

يعقوب سيد إكرام سيد و **محمد أشرف إسماعيل مدكور**

كلية الحاسبات وتقنية المعلومات، جامعة الملك عبد العزيز، جدة، المملكة العربية السعودية

Jacob.sayid@hotmail.com

المستخلص. لكشف الهجمات الإلكترونية غير المعروفة مسبقًا، والتي تستهدف الأنظمة الحديثة، تم اقتراح عدة أنظمة مبنية على أساس المضيف للكشف عن المتسللين، وذلك باستخدام مجموعة بيانات ADFA-LD التي تم تجميعها حديثا. تقوم هذه الأنظمة المقترحة بكشف الانحياز عن السلوك الطبيعي للنظام باستخدام تقنيات تعتمد على تتبع آثار استدعاءات الوظائف الأساسية في نظام التشغيل من قبل العمليات في الذاكرة، والتي تم تجميعها في مجموعة البيانات ADFA-LD. بشكل عام، يوجد في الأنظمة المقترحة قصور من عدة جوانب، يتمثل في انخفاض دقة كشف المتسللين، وارتفاع نسبة الخطأ فيما تم اكتشافه، والاستهلاك العالي جداً لموارد النظام، وطول مدة التعلم على السلوك الطبيعي للنظام، مع فقدان المرونة الكافية للاستجابة على التغيرات التي تطرأ على السلوك الطبيعي للنظام بشكل مستمر. وللتغلب على جميع هذه السلبيات وتحقيق أفضل مزيج من الدقة العالية، ونسبة الخطأ المنخفضة، والتعلم السريع للسلوك الطبيعي للنظام، تم اقتراح نظامين لكشف المتسللين مبنية على أساس المضيف. النظام الأول ينتفع من خوارزمية مستحدثة لاستخراج السلاسل القصيرة والفريدة فقط من استدعاءات الوظائف الأساسية في نظام التشغيل، وذلك لتكوين اللمحة الخاصة بالسلوك الطبيعي للنظام. بعد ذلك، يتم استخدام خوارزمية مصاحبة لتصنيف سلوك العمليات واكتشاف أي انحياز عن السلوك الطبيعي. النظام الآخر يقوم باستخراج خصائص فريدة مبنية على التكرار والتردد من آثار استدعاءات الوظائف الأساسية في نظام التشغيل، وذلك لتمثيل السلوك الطبيعي للنظام. بعد ذلك، يتم استخدام تقنيات كشف الانحياز عن السلوك الطبيعي وشبه الخاضعة للإشراف مثل support vector machines و k-nearest neighbors و k-furthest neighbors. قمنا بإنشاء نموذجين للمقترحين باستخدام لغة البرمجة جافا، بناءً على مجموعة البيانات ADFA-LD وذلك لمقارنة أداء النظامين. النتائج التجريبية أظهرت أن النظام الأول قد تفوق على النظام الثاني. إلى حد علمنا، فإن ما توصلنا إليه من نتائج قد تفوق على جميع التقنيات المنشورة حديثا من ناحية فترة التعلم على السلوك الطبيعي للنظام وكمية استهلاك الموارد. كما فاقت دقة الكشف في النظام المقترح من قبلنا تقريبًا جميع الأنظمة المقترحة مؤخرًا بالمقارنة، وكانت الدقة شبه مساوية

لأفضل ما تم نشره حتى وقت كتابة الورقة العلمية. وبشكل خاص، فإن نظام كشف التسلل من خلال كشف الانحياز عن السلوك الطبيعي للنظام، والمبني على خوارزمية استخراج السلاسل القصيرة والفريدة فقط من استدعاءات الوظائف الأساسية لنظام التشغيل قد جمع بين مزيج من عدة مزايا فضلى، كارتفاع دقة كشف المتسللين وانخفاض فترة التعلم. لقد حقق النموذج الذي تم تطويره نسبة من الدقة العالية تساوي ٩٠.٤٨٪ ونسبة خطأ تساوي ٢٢.٥٪ مع فترة تعلم على السلوك الطبيعي تساوي تقريبًا ٣٠ ثانية فقط. إن في هذا المزيج من المزايا ما يمكّنه من اكتشاف معظم الهجمات الإلكترونية غير المعروفة مسبقا، ويجعل النظام ذا مرونة ليتواءم ويتماشى مع أية تعديلات في البيئة، نظراً لأنه قابل لأن يتعلم السلوك الطبيعي الجديد بسرعة، وبشكل تكاملي من دون الحاجة إلى بناء كامل خوارزمية التصنيف من الصفر.

الكلمات المفتاحية: كشف الانحياز عن السلوك الطبيعي، الهجمات الإلكترونية الحديثة، تعلم الآلة، أنظمة كشف التسلل، خوارزميات التصنيف، والتنبؤ.