mathematics

MDPI

Article

An Efficient Network Intrusion Detection and Classification System

Iftikhar Ahmad [1,*], Qazi Emad Ul Haq [2], Muhammad Imran [3,*], Madini O. Alassafi [1] and Rayed A. AlGhamdi [1]

1 Department of Information Technology, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia; malasafi@kau.edu.sa (M.O.A.); raalghamdi8@kau.edu.sa (R.A.A.)
2 Center of Excellence in Cybercrime and Digital Forensics (CoECDF), Naif Arab University for Security Sciences (NAUSS), Riyadh 14812, Saudi Arabia; qabdulrab@nauss.edu.sa
3 School of Engineering, Information Technology and Physical Science, Federation University, Brisbane 4000, Australia
* Correspondence: iakhan@kau.edu.sa (I.A.); m.imran@federation.edu.au (M.I.)

Abstract: Intrusion detection in computer networks is of great importance because of its effects on the different communication and security domains. The detection of network intrusion is a challenge. Moreover, network intrusion detection remains a challenging task as a massive amount of data is required to train the state-of-the-art machine learning models to detect network intrusion threats. Many approaches have already been proposed recently on network intrusion detection. However, they face critical challenges owing to the continuous increase in new threats that current systems do not understand. This paper compares multiple techniques to develop a network intrusion detection system. Optimum features are selected from the dataset based on the correlation between the features. Furthermore, we propose an AdaBoost-based approach for network intrusion detection based on these selected features and present its detailed functionality and performance. Unlike most previous studies, which employ the KDD99 dataset, we used a recent and comprehensive UNSW-NB 15 dataset for network anomaly detection. This dataset is a collection of network packets exchanged between hosts. It comprises 49 attributes, including nine types of threats such as DoS, Fuzzers, Exploit, Worm, shellcode, reconnaissance, generic, and analysis Backdoor. In this study, we employ SVM and MLP for comparison. Finally, we propose AdaBoost based on the decision tree classifier to classify normal activity and possible threats. We monitored the network traffic and classified it into either threats or non-threats. The experimental findings showed that our proposed method effectively detects different forms of network intrusions on computer networks and achieves an accuracy of 99.3% on the UNSW-NB15 dataset. The proposed system will be helpful in network security applications and research domains.

Keywords: AdaBoost; network intrusion; decision tree; SVM; MLP; UNSW-NB15

check for updates

Citation: Ahmad, I.; Ul Haq, Q.E.; Imran, M.; Alassafi, M.O.; AlGhamdi, R.A. An Efficient Network Intrusion Detection and Classification System. *Mathematics* **2022**, *10*, 530. https://doi.org/10.3390/math10030530

Academic Editor: Alfonso Mateos Caballero

Received: 2 November 2021
Accepted: 26 January 2022
Published: 8 February 2022

1. Introduction

Intrusion detection is a method for observing and tracking events on a computer system. It is used to identify signs of security issues and activities are monitored using event-based techniques and security information. With the exponential increase in internet-based facilities, the number of devices for computing and consumers connected to data networks and computer networks has increased significantly. These devices provide and serve online services to users and public/private sector organizations. In parallel, there has been an infinite amount of unauthorized access to online services. Typical services include protocol-specific attacks (ARP, IP, TCP, UDP, and ICMP), traffic flooding, worms, and DoS [1]. In protocol-specific attacks, the attacker exploits a specific protocol feature installed on the target machine. Prevalent protocol attacks include SMURF, SYP, and authentication server attacks. The ever-increasing attacks demand an effective intrusion detection system that detects documented forms and learns to detect new forms. The intrusion detection system is one of the methods used to address the network security issue. The imperfectness of existing systems has allowed data mining to make several significant contributions in the field of intrusion detection.

There are three main types of intrusion detection systems [2–4]:

i Host intrusion detection system (HIDS);
ii Network intrusion detection system (NIDS);
iii Network node intrusion detection system (NNIDS).

The HIDS runs on all the machines in the network and other parts of the enterprise network. NIDS is deployed and intended to manage those places only where chances of vulnerability are high. NNIDS is like NIDS [3,4], but it only applies to one host at a time. Three main approaches are used for intrusion detection:

i Signature-based IDS;
ii Anomaly-based IDS;
iii Hybrid of signature and anomaly-based IDS.

Signature-based IDS focuses on identifying signatures and patterns and matching those patterns with the well-known signature of misuses. Anomaly-based IDS searches for forms of unknown signature attacks-based IDS that are hard to detect. Owing to the exponential growth in malware and attack styles, anomaly-based IDS uses machine learning approaches to equate trustworthy activity patterns with new behaviors.

Several supervised learning techniques such as decision trees, SVM, and ANNs [5–7] have been used, but they have a high false-positive ratio for infrequently occurring attacks. This study demonstrates an AdaBoost-based network intrusion detection system (NIDS) that takes statistical network flow features to recognize malicious activities to tackle network attacks. A set of features are extracted from network flow after potential analysis of HTTP, MQTT, and DNS network packets. For this purpose, we used the source files of the UNSW-NB15 dataset [8,9].

In the proposed system, firstly, it performs the feature selection from the entire dataset. Next, we plot a correlation matrix and select features based on their correlation with one another, and remove some highly correlated features. In this study, we used AdaBoost on a decision tree classifier to classify the normal traffic and possible threats and obtained an accuracy of 99.3%.

The key contributions in this study are presented as follows:

i. A feature selection method is proposed based on the correlation matrix calculated between the features. Features that were highly correlated with one another were removed as they added no significant difference and only increased the complexity of the model.
ii. Optimal features are selected from the UNSW NB 15 dataset.
iii. The AdaBoost-based model is proposed, using a decision tree to classify normal network traffic and network threats.

The rest of the paper is organized as follows. Section 2 describes the background and literature review. The description of the proposed system is presented in Section 3. The performance metrics and evaluation protocol are discussed in Section 4. The experimental results and discussion are presented in Section 5. Section 6 presents the comparison and Section 7 concludes the paper.

2. Background and Related Work

An intrusion detection system is a technique for tracking network activities between different entities by estimating their integrity and availability principles [10,11]. Classic systems consist of data source, preprocessing, and a decision-making method to identify vulnerability. The first step involves raw data gathered from host traces or network traffic. The second step covers the construction of features passed to the decision-making method to identify possible threats [12]. Several studies have been conducted in the past to address this challenge. The previous research studies have used KNN, SVM, and ANN models for network anomaly detection [13–15].

Awais et al. [16] proposed an ANN-based method for detecting network intrusions with an accuracy of 89.4%. The proposed structure uses a neural network for partitioning the training data. Each training data sample is used for training a network and a boosting algorithm to learn an optimized set of weights. This system was evaluated and tested on KDD99 [17] and UNSW-NB15 datasets. The UNSW-NB15 dataset contains a mixture of 49 input features, with nine attack types and normal network traffic. The details of the attack types are shown in Table 1, where they are divided into two different categories. The first category (shown at sr. # 1, Table 1) is for normal network traffic and the second category (shown at sr. # 2 to 10, Table 1) describes the attack types, which include Analysis, Backdoor, DoS, Exploits, Fuzzers, Generic, Reconnaissance, Shellcode, and Worms. The data types and their respective descriptions are shown in Table 2.

Table 1. UNSW-NB 15 dataset categories.

Sr. No.	Type	Description
1	Normal	Natural transaction data.
2	Analysis	An attack targets web applications through emails, ports, or web scripts.
3	Backdoor	Using backdoor to secure remote access.
4	DoS	Attacks computer memory
5	Exploits	An instruction that takes advantage of bugs/errors caused by unintentional behavior on the network.
6	Fuzzers	An attack to crash the system by inputting a lot of random data.
7	Generic	A technique to clash the block-cipher configuration by using hash functions.
8	Reconnaissance	A probe to evade network security controls by collecting relevant information.
9	Shellcode	Code is used to exploit software vulnerabilities.
10	Worms	A set of virus codes can add to a computer system or other programs.

Table 2. Training instances.

Class	Normal	Intrusion
Training Instances	37,000	45,332
Label	0	1

The categories of the UNSW-NB15 dataset and their descriptions are shown in Table 1, where one is the normal network traffic and the rest are different types of threats. The system could face these threats while connected to the internet. In another study [7], the authors applied a deep convolutional neural network (DCNN) on the NSL-KDD dataset for the binary classification of anomaly detection. The proposed approach was optimized by randomized grid search, hyper-parameters' optimization, and fine-tuning. They obtained the highest accuracy of 85.22% using DCNN, whereas 82.02% was obtained using the random tree classifier.

The methodology of network intrusion is either hybrid-based or intrusion-based. Many researchers have used ensemble-based intrusion detection techniques to propose enhanced systems for optimizing NIDS. Mustafa et al. [18] proposed a beta mixture model for anomaly detection. The author computed a normal network profile, calculated the deviations from that network profile, and considered that as an anomaly. The UNSW-NB15 dataset was used for performance evaluation using external evaluation metrics. This technique slightly improved the accuracy of the UNSW-NB15 dataset. In another study [19], the authors use ANNs for signature-based intrusion detection in IoT devices. They tested the approach on various malicious and heterogeneous data, achieving an average accuracy of 84% with an average false-positive rate of 8%.

Benjamin et al. [20] proposed an ensemble-based intrusion detection technique against application layer protocols (MQTT, HTTP, and DNS), as shown in Figure 1. The authors generated the new statistical flow features from the protocols based on their potential properties. The experimental analysis showed that statistically generated features are related to normal and suspected activities. Ensemble-based AdaBoost was developed using naïve Bayes, decision trees, and artificial neural network (ANN). NIMS botnet was used to detect and evaluate ensemble learning.

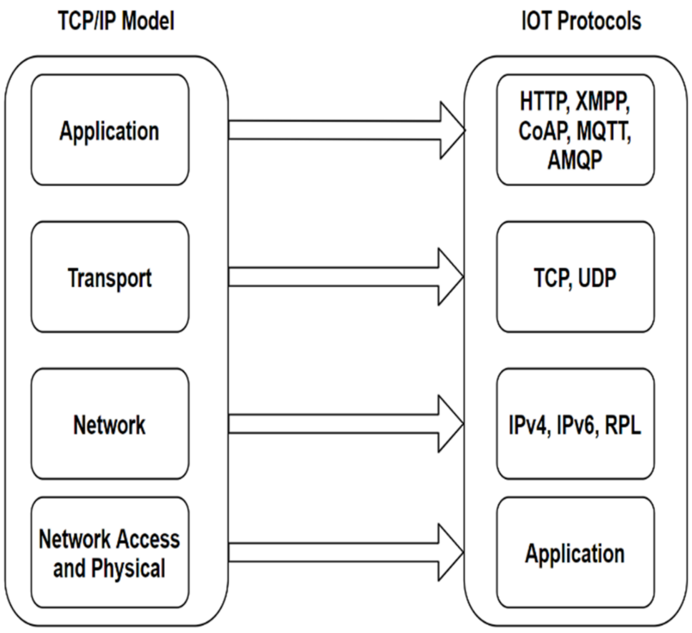

Figure 1. TCP/IP model and IoT protocols.

Wei et al. [21] propose an Adaboost-based network intrusion detection system based on the principle of weak learners. They use naïve Bayes as weak learners and boost the performance utilizing the weight update ability of the Adaboost. Yuan et al. [22] proposed a semi-supervised tri-Adaboost-based methodology for network intrusion detection. They suggest using three distinct weak Adaboost-based classifiers for training their model. The dimension of the features is reduced using the chi-square methodology.

Sheraz et al. [23] presented an anomaly system based on deep learning. Convolution neural networks (CNNs), recurrent neural networks (RNNs), and autoencoders structures are proposed. They also test the conventional IDS based on machine learning and present an evaluation of their findings. Truong et al. [24] presented an empirical study on the use of generative adversarial networks (GANs) for network anomaly detection. The authors consider using multiple datasets and perform extensive experiments to test the robustness of GANs. A traffic aggregation technique was also developed to extract statistical features. Dutta et al. [25] propose using DNN consisting of multiple stacked fully connected layers that present a flow-based system for multiclass anomaly detection. Lecun uniform initialization is used for the initialization of weights. Michal et al. [26] propose a two-stage hybrid network anomaly detection system. Classical autoencoder (CAE) and DNN are used for feature engineering and classification. They evaluate the performance of their model using the UNSW-NB-15 dataset and achieve an accuracy of 91.29%.

Yuan et al. [27] propose an anomaly detection architecture to detect short interval intrusions using SVM. They train the model using cross-correlation and Kullback–Leibler (KL) divergence calculated by the data planes and control traffic. Furthermore, they evaluate the performance of their proposed model based on a realistic internet traffic dataset. Giacinto and Roli [28] developed a network intrusion system using an automated design of classifier systems. Intrusions were detected based on the voting rule method. The experimental results showed that hybrid classification techniques are effective if each classifier performs well. Redundant features have little contribution to NIDS. Srilatha and Johnson [29] focused on examining and finding potential data features for intrusion detection systems. They aimed to design an optimized and computationally effective NIDS. Srilatha proposed a hybrid technique involving ensemble-based classifiers. PCA was used to reduce input dimensions, but has not achieved the expected results in the NIDS domain. MIT Lincoln laboratory prepared the dataset for the evaluation of the proposed solution.

Arif et al. [30] proposed using Adaboost for an efficient intrusion detection system. The synthetic minority oversampling technique (SMOTE) is considered for handling the class imbalance problem, whereas principal component analysis (PCA) and ensemble feature selection (EFS) are selected for feature selection. Through this technique, they obtain an accuracy of 81.83% on the CIC IDS 2017 database (University of New Brunswick: Fredericton, NB, Canada).

From the above literature review, we observed that that previous research uses Adaboost mainly for the weight update property and does not consider the importance of discriminative features to distinguish the threats from normal network activity. Thus, we believe in using a complex dataset (UNSW NB 15), (UNSW Canberra: Canberra, Australia) selecting the most discriminative features, and utilizing AdaBoost as a weight update model. At the same time, the decision tree is a primary classifier. Decision trees perform exceptionally well on network data. The weight update in the algorithm is done by Adaboost, which shows promising results when used in combination with the decision tree.

3. Proposed Method

This study proposes an automatic and efficient network intrusion detection system based on the Adaboost technique. Firstly, we select the features from the pool of features from the UNSW-NB15 dataset. For this purpose, we perform a correlation analysis. From the result of the investigation, we remove those highly correlated features with each other, as shown in Figure 2. Furthermore, this study presents the techniques to classify network traffic as normal traffic or threats.

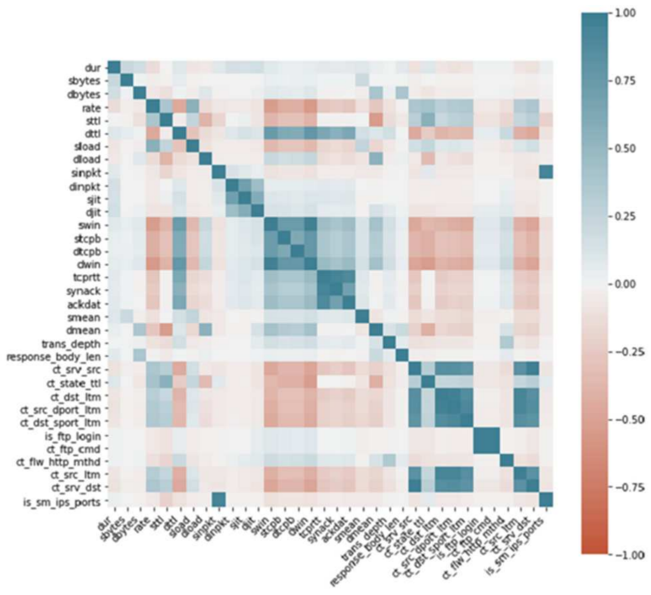

Figure 2. Correlation matrix.

We briefly review ANN, SVM, and Adaboost-based decision trees in our proposed system. ANN and SVM were used for comparison. Furthermore, the techniques are described below.

i. Artificial neural network (ANN): The artificial neural network technique is used to match targets by transforming respective inputs into outputs of each class. The result depends on the weights learned for each neuron and the activation function to get the output. Training a successful model requires normal and irregular network data to train the weights more effectively.

$$f(I) = \tau(\sum_j W_j \cdot I_j) \tag{1}$$

The model uses an activation function in which Equation (1) represents the input data, where τ represents the sigmoid activation and w_j is the weight corresponding to each input.

ii. Support vector machine (SVM): Support vector machine or SVM is a widely known classifier used for regression and classification purposes. This study uses SVM for network intrusion on the UNSW-NB15 data set [14]. SVM provides various kernel functions used to map low dimensional to high dimensional space in an SVM model. This study used the RBF kernel to obtain the results, as discussed in the next section [31]. Equation (2) represents the formula for this approach as shown below:

$$\mathrm{K}\ (x_i\ ,\ x_j\ ) = \exp\left(\gamma||x_i\ - x_j\ ||^2\right) \tag{2}$$

iii. Adaboost: In this study, we proposed the use of a decision tree's-based Adaboost model, which uses the basic principles of decision trees as a primary classifier, but, on top of that, employs Adaboost for the weight updates, hence achieving outstanding results. Section 3.4 discusses the algorithm for our proposed approach.

3.1. System Architecture

The system diagram of the proposed system is shown in Figure 3. The proposed system architecture is based on different components used to perform network intrusion detection.

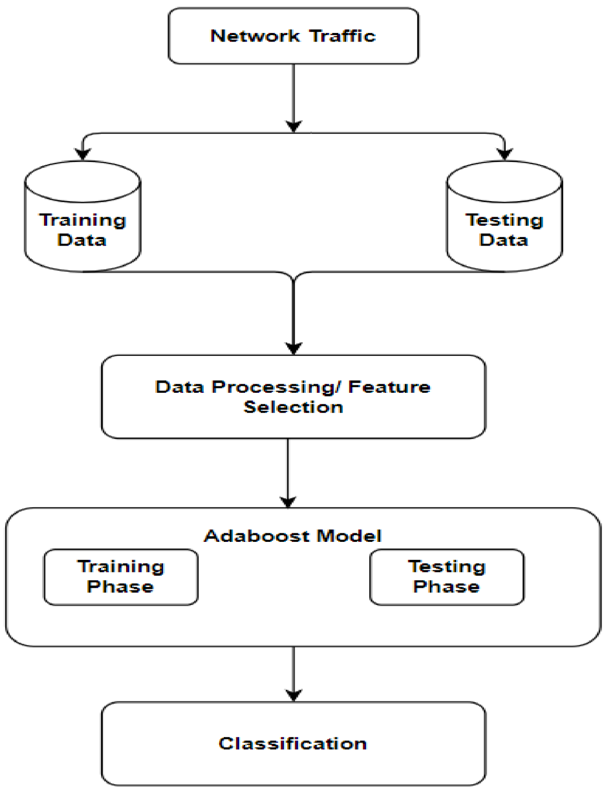

Figure 3. Overall proposed system.

The UNSW-NB15 dataset is a comprehensive dataset that contains 49 features utilized in detecting network intrusions. We used a subset of the entire available data. Its training and testing instances specification is mentioned in Tables 2 and 3, respectively. Data were divided into training and testing sets. Then, feature selection was applied based on the correlation matrix. In the next step, we trained our Adaboost model, which used a decision tree as a classifier using maximum depth = 2 from the pool {2,4,6,8} as the complexity of the model increased when the depth increased to 4 and the tree tends to overfit for higher depth values. The algorithm selected was 'SAMME.R' because, when tested, the SAMME.R algorithm converged faster than others, hence utilizing fewer boosting iterations and lower error rates for the proposed model.

Table 3. Testing instances.

Class	Normal	Intrusion
Testing Instances	56,000	119,341
Label	0	1

There were 82,332 input samples in the training set and 175,341 samples in the test set. Decision tree was used as the classifying model, while Adaboost performed the weight updates. Section 3.4 discusses the algorithm for our proposed approach (Algorithm 1).

3.2. Dataset Description

In this study, we used the UNSW-NB15 dataset for evaluating the performance of the proposed network intrusion system. Nour et al. [32] examined the complexity of the UNSW-NB-15 dataset in three aspects and found that the dataset is more complex. Furthermore, the authors presented that KDD CUP 99, NSLKDD, and KDD 98 do not reflect modern footprint attacks, whereas the UNSW-NB-15 dataset comprises current synthesized suspicious network activities. The Australian Center for Cyber Security (ACCS) used IXIA Perfect Storm to create the raw network packets. Using Argus, Bro-IDS tools, this dataset contains 49 input features and class labels representing nine different attack types [1,33]. The input features include packet-based features, time features, connection features, and content features. Packet-based features are based on the payload exchanged between the hosts. Time features assist the examination of source jitter, destination jitter, source, and destination inter-packet arrival times. The content features provide the content-related information, e.g., sequence numbers, mean packet size, and content size of data from the server. The connection-based features offer general information about the connection, e.g., number of links calling the same service, number of connections from source, and number of references to the destination address. The available features contain general HTTP methods for sending the information and FTP login details.

The attack types are DoS, Fuzzers, Exploit, Worm, shellcode, reconnaissance, generic, analysis, and backdoor (see description of attacks in Table 1). We computed the values of the newly created column as 0 for all the normal activities and 1 for the rest of the nine attack types. CSV files are classified into two network activities as discussed below:

i Normal
ii Intrusions

Network activity is classified as normal if there is no network intrusion. In contrast, activity is classified as a network attack when someone breaches an internet-based application via the port, bypassing the network authentication, accessing the resources with unauthorized access, and targeting security loopholes. We used a label encoder to convert the 'activity' column into numerical data. The data are divided into two sets, i.e., training and testing sets. The training set contains 82,332 instances, whereas the testing set consists of 175,341 records. Previous work on this dataset acknowledges that the data are normally distributed in the sense that the different types of network traffic are evenly distributed between training and testing sets [9]. Tables 2 and 3 describe the class-based distributions for training and testing data.

Furthermore, feature selection has been applied to select optimum features, as explained in the next section.

3.3. Feature Selection

Feature selection is a process of selecting discriminative feature variables that contribute most to the target variable. The objective of the feature selection process is to reduce the computational cost of the model by removing the variables that have no contribution to the determination of the target variable.

We used *p*-value and correlation measures for examining the relationship between input variables and output labels. The *p*-value is the probability of observation under the observation. This probability is then used to accept or reject the hypothesis. Equation (3) is used for computing the *p*-value as follows:

$$z = \frac{\hat{P} - P0}{\sqrt{\frac{P0\ (1-P0)}{n}}} \tag{3}$$

Here, $\hat{p}$ is the sample proportion, $P0$ is the assumed population under the null hypothesis, and n is the sample size. Then, the *p*-value can be computed by finding the corresponding value from z value obtained. We have not achieved the expected results using the *p*-value, so we used correlation measures for feature selection.

The correlation coefficient is the statistical measure of finding the strength of the relationship between two variables using their relative moments. Its value ranges from +1 to -1.

The correlation coefficient variable of value +1 shows a strong positive correlation between two variables, whereas -1 represents a strong negative relationship between two variables. There is no relationship between the two attributes if the value of the correlation coefficient is zero.

$$r = \frac{\sum(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum(x_i - \overline{x})^2 \sum(y_i - \overline{y})^2}} \tag{4}$$

In Equation (4), x_i are the values of X variable in the sample, x is the mean value of X variable, yi are the values of Y variables in the sample, and y is the mean value of Y variable.

We computed the correlation matrix of all the variables and eliminated those variables that do not correlate with the target variable. The correlation matrix for all the input variables is shown in Figure 2. As shown in Figure 2, there are some input variables where either they have no correlation with the target variable or they have a strong correlation with each other. We have removed those variables as well. After analyzing the correlation matrix, we are removing attack_cat, proto, service, state, id, spkts, dpkts, sloss, dloss, and ct_dst_src_ltm. The potential features that we are using in the proposed NIDS are shown in Table 4.

Table 4. Input features for NIDS.

Name	Type	Description
dur	Float	Record total duration
sbytes	Integer	Source to Destination transaction bytes
dbytes	Integer	Destination to source transaction bytes
sttl	Integer	Source to destination time to live value
dttl	Integer	Destination to source time to live value
sload	Float	Source bits per second
dload	Float	Destination bits per second

Table 4. *Cont.*

Name	Type	Description
sinpkt	Float	Source interpacket arrival time (mSec)
dinpkt	Float	Destination interpacket arrival time
sjit	Float	Source jitter (mSec)
djit	Float	Destination jitter (mSec)
swin	Integer	Source TCP window advertisement value
stcpb	Integer	Source TCP base sequence number
dtcpb	Integer	Destination TCP base sequence number
dwin	Integer	Destination TCP window advertisement value
tcprtt	Float	TCP connection setup round-trip time
synack	Float	TCP connection setup time, the time between the SYN and the SYN_ACK packets.
ackdat	Float	TCP connection setup time, the time between the SYN_ACK and the ACK packets.
smean	Integer	Mean of the packet size transmitted by the src
dmean	Integer	Mean of the packet size transmitted by the DST
trans_depth	Integer	Represents the pipelined depth into the connection of HTTP request/response transaction
response_body_len	Integer	Actual uncompressed content size of the data transferred from the server's HTTP service.
ct_srv_src	Integer	No. of connections that contain the same service (14) and source address (1) in 100
st_state_sttl	Integer	No. for each state
ct_dst_ltm	Integer	No. of connections of the same destination address
ct_src_dport_ltm	Integer	No connections of the same source address
ct_src_sport_ltm	Integer	No of connections of the same destination address
is_ftp_login	Binary	Session is accessed by user, then 1, else 0
ct_ftp_cmd	Integer	No of flows that have a command in the FTP session.
ct_flw_http_mthd	Integer	No. of flows with methods such as Get and Post an HTTP service.
st_src_ltm	Integer	No. of connections of the same source address
ct_srv_dst	Integer	No. of connections of the same destination address
is_sm_ips_ports	Integer	If the source and destination IP addresses are equal, then the value 1, else 0

3.4. Adaboost Algorithm Pseudocode

The Adaboost algorithm is an iterative procedure that usually comprises many classifiers [34]. The Adaboost algorithm works in that it first classifies the training input data and produces the output labels. It then compares the results with the actual output and boosts the weight if it is wrongly classified. Again, misclassified data are organized with the boosted weights, the same process is repeated, and the weights are constantly updated [34].

Algorithm 1 AdaBoost Pseudocode

```
Require : n $\geq$ 0
$w_i \leftarrow n$
for m $\leftarrow$ 1 to M do
        $Fit\ a\ classifier\ x.(T^m)\ to\ the\ training\ data\ using\ weights\ w_i.$
        $err^m \leftarrow \Sigma_1^n w_i II(c_i \neq T^m(x_i)) \backslash \Sigma_1^n w_i$
        $e^m \leftarrow log(1 - err^m / err^m)$
        $w_i = \omega_i.exp(e^m \cdot II(c_i \neq T^m(x_i)))$
        Renormalize $w_i$.

        $C(x) \leftarrow arg\big(max_k\, \Sigma_1^{m\ m} \cdot II(T^m(x) = k)\big)$
```

4. Performance Metrics and Evaluation Protocol

In this section, we analyze the performance of our method for two classification problems (normal traffic vs. threats) using the UNSW-NB15 dataset with a 10-fold cross-validation technique. The detail about the classes is given in the previous section of our proposed method.

This section explains the performance evaluation and evaluation matrix used in the experiments. Then, we discuss the impact of fewer potential features. Finally, the comparison is shown between SVM, ANN, and Adaboost classification algorithms.

We conducted several experiments to find the effectiveness of the proposed system. We used TensorFlow with python for the implementation of our proposed model. All of our assessments were performed on a 64-bit Windows system running GPU-enabled TensorFlow with a Core i7-10750H (10th Gen), 32 GB RAM, and an NVIDIA GeForce RTX 2070 Super (8 GB) GPU.

We used the UNSW-NB15 dataset (described in Section 3.2) to conduct our assessments. In NIDS, this dataset is treated as a benchmark. In addition, the use of this dataset helps draw comparisons with current approaches and studies.

We used the following performance metrics in this study to evaluate the system:

i True Positive (TP)—Attack data correctly classified as an attack.
ii False Positive (FP)—Normal data incorrectly classified as an attack.
iii True Negative (TN)—Normal data correctly classified as normal.
iv False Negative (FN)—Attack data incorrectly classified as normal.

We used different performance metrics to measure the performance of our method. Accuracy computes the number of accurate classifications out of total samples—Equation (5).

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total Samples}} \tag{5}$$

The precision computes the number of correct classifications penalized by the number of incorrect classifications—Equation (6).

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{6}$$

The recall computes the number of correct classifications penalized by the missed entries—Equation (7).

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{7}$$

The F-score measures the harmonic mean of precision and recall, which is a derived effectiveness measurement—Equation (8).

$$\text{F} - \text{Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{8}$$

We trained the models on a massive dataset with no duplication of records for evaluating the performance and to overcome the overfitting and underfitting problems.

We tested our models by setting various parameters and, based on the results, we came to the following parameters that give us the best results, as mentioned below:

i SVM technique was adjusted by kernel = 'RBF, degree = 3, gamma = 'scale', and shrinking = 'true'; these parameters were selected as they showed the best results. The rest of the parameters were set to default.
ii Four hidden layers tuned ANN from a combination of {4,6,8} because increasing the number of hidden layers had no effect on the results, and their sizes were 256, 128, 64, and 32, respectively. We used 'relu' as the activation function and 'Adam' as a solver for weight optimization as they gave optimal results.

iii The Adaboost classifier was adopted on top of the decision tree classifier with maximum depth = 2, algorithm = 'SAMME.R'. Max depth was kept at two as increasing causes the model to overfit, and the 'SAMME.R' algorithm was selected owing to its faster convergence.

5. Experimental Results and Discussion

In this section, we analyze the performance of our proposed system for the classification problem using the UNSW-NB15 dataset. We evaluate the results of SVM, ANN, and Adaboost on top of the decision tree classifier for a binary classification problem of network intrusion detection on the UNSW-NB15 dataset. Most of the work done revolves around the traditional datasets. We focus on the more complex network dataset using reduced features and apply three algorithms to compare the performance. This research aims to accurately classify normal network traffic and network threats from suspicious network activities.

In this paper, we compared various techniques for classifying network data into the form of threats or non-threats. We tested multiple models, including the use of ANN, SVM, and finally our proposed technique of Adaboost based on the decision tree classifier. While ANN uses an activation function to generate outputs, it performs weight updates at each neuron to improve the results. We achieve an accuracy of 89.54% when we set the input parameters to four hidden layers, with 256, 128, 64, and 32 hidden sizes; 'relu' as the activation function; and 'Adam' as a solver for weight optimization.

In this study, we also used support vector machine (SVM) to classify network intrusions. In SVM, we plot each sample point in an n-dimensional hyperplane, and then we find the maximum distance between the hyperplanes, which differentiate the classes quite well. We use input parameters, kernel 'RBF' with degree = 3, gamma = 'scale', and shrinking = 'true', and we achieve an accuracy of 94.7%.

The proposed approach we present in this paper is Adaboost-based decision tree classification. The proposed method uses decision tree as a classifier, while Adaboost is used for weight updates. A correlation matrix was computed to find the correlation of the input features. Depending on the results, we performed feature selection, where we removed some features on the following bases:

i They are highly correlated with one another.
ii They do not affect the performance of our model.

The parameters we used in our Adaboost model were maximum depth = 2 and algorithm = 'SAMME.R'. We achieved an accuracy of 99.3%. Furthermore, the comparison of the results of the three models is shown in Table 5.

Table 5. Performance (%) of the proposed system using the UNSW-NB15 dataset.

	ANN	**SVM**	**Adaboost**
Accuracy	89.54	94.7	99.3
Precision	91.12	97.6	99.7
Recall	93.4	92.07	98.5
F-Score	90.6	95.8	99.95

Figure 4 presents the performance evaluation matrix between the accuracy, precision, recall, and F-score among the SVM, ANN, and Adaboost-based decision tree classifier models.

Table 5 displays the results of our proposed models for the NIDS problem using the UNSW-NB15 dataset. When we observe the outcomes of our model, we can see that the achieved results are overall higher than those of other existing approaches. Our proposed model achieves the highest accuracy of 99.3%.

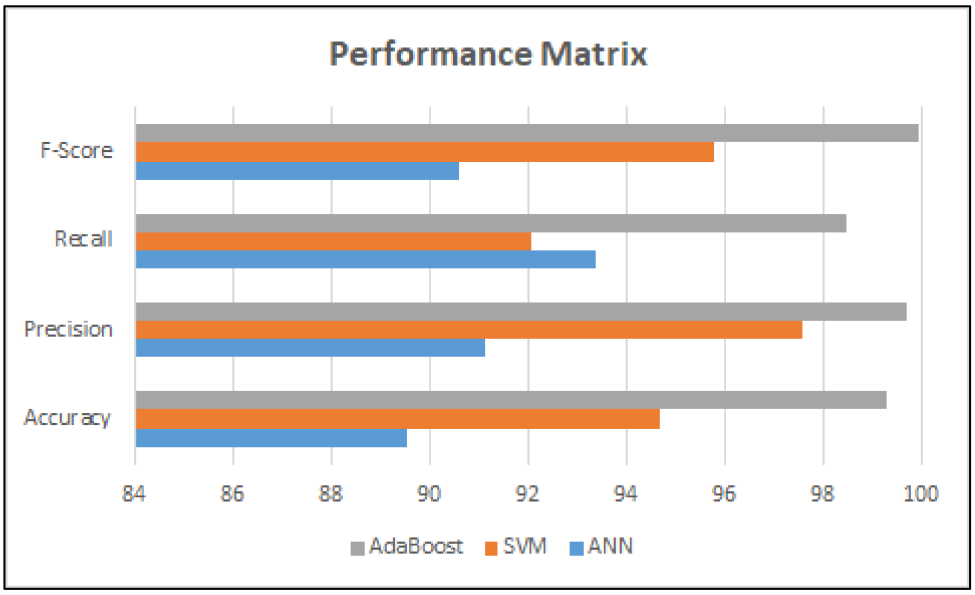

Figure 4. Performance matrix based on different classifiers.

6. Comparison

This section compares the different approaches used to detect network intrusion anomalies and their datasets. Table 6 presents the detailed overview of some of the techniques used to detect network anomalies using the UNSW-NB15 dataset. Previously, various datasets were used for network intrusion detection, but none of them were as comprehensive as UNSW-NB15, as it contains nine different types of attack categories. Although we are classifying threats and normal traffic in our approach, we obtain a variety of input network features to train the model. ANN (artificial neural network) [22] has been used on the UNSW-NB15 dataset as a whole and obtained an accuracy of 84%. Adaboost-based neural network learning [5] has been used where NN (neural network) serves as a base model and Adaboost is used to achieve an 86.40% classification accuracy. Wei et al. [35] used an Adaboost algorithm for network intrusion detection. They present a four-module system: Feature extraction, data labeling, weak classifiers design, and robust classifier construction. Furthermore, an improved objective function and weight initialization method is presented to adjust the false positive ratio (FPR) and detection rate (DR). The authors test their approach on the KDD CUP 99 dataset and obtain a maximum accuracy of 90.88%. Wei et al. [36] also propose an improved Adaboost algorithm that uses decision stumps as weak classifiers. They offer to combine two weak classifiers for continuous and categorical features into a robust classifier. They show that their algorithm has low computational complexity and error rates by experimentation. Their algorithm provides detection rates between 90.4 and 90.88%. Naseer et al. [7] proposed a deep convolution neural network (DCNN), which is fine-tuned using randomized search over configuration space. Furthermore, they test their approach using the NSLKDD dataset. Mabu et al. [37] presented a novel fuzzy class association rule mining approach using genetic network programming (GNP) to detect network intrusions. Decision trees and negative selection algorithms have also been used to detect network intrusions, as illustrated in Table 6. The highest accuracy of 99.3% is achieved by our proposed model using the Adaboost-based decision tree classifier models owing to the optimal features selected from the dataset. These comparisons indicate that our model's results are excellent compared with other methods based on Adaboost-based decision tree classifiers.

Table 6. Comparison with existing work.

No	References	Technique	Dataset	Accuracy
1	[19]	ANN	UNSW-NB15	84%
2	[16]	AdaBoost-based neural network learning	UNSW-NB15	86.40%
3	[37]	GNP	KDD cup99	90.26%
4	[38]	Negative selection algorithm	Constructed a set of receptors	80%
5	[7]	DCNN	NSLKDD	85.22%
6	[35]	AdaBoost	KDD cup99	90.4–90.88% DR
7	[39]	Decision tree	UNSW-NB15	95%
8	[21]	Naïve Bayes with Adaboost	KDD cup99	84.32% DR
9	[22]	Semi-supervised tri-Adaboost	KDD cup99	89.05% DR
10	[30]	Adaboost	CIC IDS 2017	81.83
11	[36]	Adaboost	KDD CUP	90.4–90.88% DR
12	Our Proposed Method	Adaboost-based decision tree classifier	UNSW-NB15	99.3%

7. Conclusions and Future Work

In this study, we proposed a feature selection method based on the correlation matrix among all the features of the UNSW-NB 15 dataset. We also propose an approach for network intrusion detection based on the selected features using the Adaboost-based decision tree classifier. Furthermore, we also discussed the issues faced by the current NIDS (network intrusion detection system) techniques. The proposed method firstly involves feature selection based on a correlation matrix. We discarded the features that were highly correlated with other input features and had little effect on the output label. The technique is based on AdaBoost on top of the decision tree classifier. The proposed method for network intrusion detection has very high accuracy with the UNSW-NB15 dataset as the model was trained and tested on the best discriminating features. In comparison with the previous works, we assessed the abilities of our model based on the dataset used and demonstrated consistent accuracy in the classification. We have evaluated our proposed system using different performance metrics, and performed the comparison with the state-of-the-art techniques to show that the proposed NIDS system performs better than the existing systems. The proposed NIDS will be helpful in network security applications and research domains.

Furthermore, this work can be extended to the real world to identify the dynamic intrusions on live network traffic. An analysis of suitable classifiers can also be performed to detect and analyze the performance of the classifiers. We intend to perform multiclass analysis to detect and classify different types of threats based on their categories.

Author Contributions: Conceptualization, I.A., Q.E.U.H., and M.I.; data curation, Q.E.U.H. and M.I.; formal analysis, I.A. and Q.E.U.H.; funding acquisition, I.A.; methodology I.A., Q.E.U.H. and M.I.; project administration, I.A.; resources, M.I.; software, Q.E.U.H.; supervision, I.A. and M.I.; validation, M.O.A.; visualization, R.A.A.; writing—original draft, Q.E.U.H.; writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research work was funded by Institutional Fund Projects under grant no. (IFPHI-144-611-2020). Therefore, the authors gratefully acknowledge technical and financial support from the Ministry of Education and King Abdulaziz University, DSR, Jeddah, Saudi Arabia.

Acknowledgments: This research work was funded by Institutional Fund Projects under grant no. (IFPHI-144-611-2020). Therefore, the authors gratefully acknowledge technical and financial support from the Ministry of Education and King Abdulaziz University, DSR, Jeddah, Saudi Arabia.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Dubrawsky, I. (Ed.) Chapter 2—General Security Concepts: Attacks. In *How to Cheat at Securing Your Network*; Syngress: Maryland Heights, MO, USA, 2007; pp. 35–64. ISBN 9781597492317. [CrossRef]
2. Jee, K.; Zhichun, L.I.; Jiang, G.; Korts-Parn, L.; Wu, Z.; Sun, Y.; Rhee, J. Host Level Detect Mechanism for Malicious DNS Activities. U.S. Patent Appl. 15 644 018, 11 January 2018.

3. Soniya, S.S.; Vigila, S.M.C. Intrusion detection system: Classification and techniques. In Proceedings of the 2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT), Nagercoil, India, 18–19 March 2016.
4. Spafford, E.; Zamboni, D. *Data Collection Mechanisms for Intrusion Detection Systems*; CERIAS Technical Report; Center for Education and Research in Information Assurance and Security: West Lafayette, IN, USA, 2000; 47907-1315.
5. Bouzida, Y.; Cuppens, F. Neural networks vs. decision trees for intrusion detection. In Proceedings of the IEEE/IST Workshop on Monitoring, Attack Detection and Mitigation (MonAM), Tuebingen, Germany, 28–29 September 2006; Volume 28.
6. Habeeb, R.A.A.; Nasaruddin, F.; Gani, A.; Hashem, I.A.T.; Ahmed, E.; Imran, M. Real-time big data processing for anomaly detection: A Survey. *Int. J. Inf. Manag.* **2019**, *45*, 289–307. [CrossRef]
7. Naseer, S.; Saleem, Y. Enhanced Network Intrusion Detection using Deep Convolutional Neural Networks. *KSII Trans. Internet Inf. Syst.* **2018**, *12*, 5159–5178. [CrossRef]
8. The-UNSW-NB15-Dataset. Available online: https://paperswithcode.com/dataset/unsw-nb15 (accessed on 6 August 2021).
9. Moustafa, N.; Slay, J. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, ACT, Australia, 10–12 November 2015.
10. Pontarelli, S.; Bianchi, G.; Teoli, S. Tra c-aware design of a high-speed fpga network intrusion detection system. *IEEE Trans. Comput.* **2013**, *62*, 23222334. [CrossRef]
11. Tan, Z.; Jamdagni, A.; He, X.; Nanda, P.; Liu, R.P. A system for denial-of-service attack detection based on multivariate correlation analysis. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *25*, 447456.
12. Moustafa, N.; Slay, J. A hybrid feature selection for network intrusion detection systems: Central points. *arXiv* **2017**, arXiv:1707.05505.
13. Li, W.; Liu, Z. A method of SVM with normalization in intrusion detection. *Procedia Environ. Sci.* **2011**, *11*, 256–262. [CrossRef]
14. Liao, Y.; Vemuri, V.R. Use of k-nearest neighbor classifier for intrusion detection. *Comput. Secur.* **2002**, *21*, 439–448. [CrossRef]
15. Choudhary, S.; Kesswani, N. Analysis of KDD-Cup'99, NSL-KDD and UNSW-NB15 Datasets using Deep Learning in IoT. *Procedia Comput. Sci.* **2020**, *167*, 1561–1573. [CrossRef]
16. Baig, M.M.; Awais, M.M.; El-Alfy, E.S.M. A multiclass cascade of Artificial neural network for network intrusion detection. *J. Intell. Fuzzy Syst.* **2017**, *32*, 2875–2883. [CrossRef]
17. KDD Cup 1999 Dataset for Network-Based Intrusion Detection Systems. Available online: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (accessed on 13 July 2021).
18. Moustafa, N.; Creech, G.; Slay, J. Anomaly Detection System using Beta Mixture Models and Outlier Detection. In *Progress in Computing, Analytics and Networking*; Springer: Singapore, 2018; pp. 125–135.
19. Zhang, J.; Zulkernine, M. Anomaly based network intrusion detection with unsupervised outlier detection. In Proceedings of the 2006 IEEE International Conference on Communications, Istanbul, Turkey, 11–15 June 2006; Volume 5, pp. 2388–2393.
20. Moustafa, N.; Turnball, B.; Kwang, K. An Ensemble Intrusion Detection Technique Based on Proposed Statistical Flow Features for Protecting Network Traffic of Internet of Things. *IEEE Internet Things J.* **2018**, *6*, 4815–4830. [CrossRef]
21. Li, W.; Li, Q. Using Naive Bayes with AdaBoost to Enhance Network Anomaly Intrusion Detection. In Proceedings of the 2010 Third International Conference on Intelligent Networks and Intelligent Systems, Shenyang, China, 1–3 November 2010; pp. 486–489. [CrossRef]
22. Yuan, Y.; Huo, L.; Yuan, Y.; Wang, Z. Semi-supervised tri-Adaboost algorithm for network intrusion detection. *Int. J. Distrib. Sens. Netw.* **2019**, *15*, 1550147719846052. [CrossRef]
23. Naseer, S.; Saleem, Y.; Khalid, S.; Bashir, M.K.; Han, J.; Iqbal, M.M.; Han, K. Enhanced Network Anomaly Detection Based on Deep Neural Networks. *IEEE Access* **2018**, *6*, 48231–48246. [CrossRef]
24. Truong-Huu, T.; Dheenadhayalan, N.; Pratim Kundu, P.; Ramnath, V.; Liao, J.; Teo, S.G.; Praveen Kadiyala, S. An Empirical Study on Unsupervised Network Anomaly Detection using Generative Adversarial Networks. In Proceedings of the 1st ACM Workshop on Security and Privacy on Artificial Intelligence (SPAI '20), Taipei, Taiwan, 6 October 2020; pp. 20–29. [CrossRef]
25. Toupas, P.; Chamou, D.; Giannoutakis, K.M.; Drosou, A.; Tzovaras, D. An Intrusion Detection System for Multi-class Classification Based on Deep Neural Networks. In Proceedings of the 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), Boca Raton, FL, USA, 16–19 December 2019; pp. 1253–1258. [CrossRef]
26. Dutta, V.; Choraś, M.; Kozik, R.; Pawlicki, M. Hybrid Model for Improving the Classification Effectiveness of Network Intrusion Detection. In *Advances in Intelligent Systems and Computing*; Herrero, Á., Cambra, C., Urda, D., Sedano, J., Quintián, H., Corchado, E., Eds.; Springer: Cham, Swizterland, 2021; Volume 1267. [CrossRef]
27. Zhang, Y.; Yang, Q.; Lambotharan, S.; Kyriakopoulos, K.; Ghafir, I.; AsSadhan, B. Anomaly-Based Network Intrusion Detection Using SVM. In Proceedings of the 2019 11th International Conference on Wireless Communications and Signal Processing (WCSP), Xi'an, China, 23–25 October 2019; pp. 1–6. [CrossRef]
28. Giacinto, G.; Roli, F. An approach to the automatic design of multiple classifier systems. *Pattern Recognit. Lett.* **2001**, *22*, 25–33. [CrossRef]
29. Chebrolu, S.; Abraham, A.; Thomas, J.P. Feature deduction and ensemble design of intrusion detection systems. *Comput. Secur.* **2005**, *24*, 295–307. [CrossRef]
30. Yulianto, A.; Sukarno, P.; Suwastika, N.A. Improving AdaBoost-based Intrusion Detection System (IDS) Performance on CIC IDS 2017 Dataset. *J. Phys. Conf. Ser.* **2019**, *1192*, 012018. [CrossRef]

31. Osuna, E.; Freund, R.; Girosi, F. *Support Vector Machines: Training and Applications*; Tech. Rep. A.I. Memo No. 1602; Massachusetts Institute of Technology: Cambridge, MA, USA, 1997.
32. Moustafa, N.; Jill, S. The evaluation of network anomaly detection systems: Statistical analysis of the unswnb15 data set and the comparison with the kdd99 data set. *Inf. Secur. J. Glob. Perspect.* **2016**, *25*, 18–31. [CrossRef]
33. The-Bro-IDS-Tool. Available online: https://www.bro.org/ (accessed on 21 July 2021).
34. Zhu, J.; Rosset, S.; Zou, H.; Hastie, T. Multiclass AdaBoost. *Stat. Its Interface* **2009**, *2*, 349–360.
35. Hu, W.; Hu, W. Network-based intrusion detection using adaboost algorithm. In Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05), Compiegne, France, 19–22 September 2005; pp. 712–717.
36. Hu, W.; Hu, W.; Maybank, S. AdaBoost-Based Algorithm for Network Intrusion Detection. *IEEE Trans. Syst. Man Cybern. Part B* **2008**, *38*, 577–583. [CrossRef]
37. Mabu, S.; Chen, C.; Lu, N.; Shimada, K.; Hirasawa, K. An Intrusion-Detection Model Based on Fuzzy Class-Association-Rule Mining Using Genetic Network Programming. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **2011**, *41*, 130–139. [CrossRef]
38. Widulinski, P.; Wawryn, K. A Human Immunity Inspired Intrusion Detection System to Search for Infections in an Operating System. In Proceedings of the 27th International Conference on Mixed Design of Integrated Circuits and System (MIXDES), Wroclaw, Poland, 25–27 June 2020; pp. 187–191.
39. Khammassi, C.; Krichen, S. A ga-lr wrapper approach for feature selection in network intrusion detection. *Comput. Secur.* **2017**, *70*, 255–277. [CrossRef]