electronics

MDPI

Article

Hybrid Intrusion Detection System Based on the Stacking Ensemble of C5 Decision Tree Classifier and One Class Support Vector Machine

Ansam Khraisat *, Iqbal Gondal, Peter Vamplew, Joarder Kamruzzaman and Ammar Alazab

Internet Commerce Security Laboratory, Federation University Australia, Mount Helen 3350, Australia; iqbal.gondal@federation.edu.au (I.G.); p.vamplew@federation.edu.au (P.V.); joarder.kamruzzaman@federation.edu.au (J.K.); aalazab@mit.edu.au (A.A.)

* Correspondence: a.khraisat@federation.edu.au

Received: 10 January 2020; Accepted: 14 January 2020; Published: 17 January 2020

Abstract: Cyberttacks are becoming increasingly sophisticated, necessitating the efficient intrusion detection mechanisms to monitor computer resources and generate reports on anomalous or suspicious activities. Many Intrusion Detection Systems (IDSs) use a single classifier for identifying intrusions. Single classifier IDSs are unable to achieve high accuracy and low false alarm rates due to polymorphic, metamorphic, and zero-day behaviors of malware. In this paper, a Hybrid IDS (HIDS) is proposed by combining the C5 decision tree classifier and One Class Support Vector Machine (OC-SVM). HIDS combines the strengths of SIDS) and Anomaly-based Intrusion Detection System (AIDS). The SIDS was developed based on the C5.0 Decision tree classifier and AIDS was developed based on the one-class Support Vector Machine (SVM). This framework aims to identify both the well-known intrusions and zero-day attacks with high detection accuracy and low false-alarm rates. The proposed HIDS is evaluated using the benchmark datasets, namely, Network Security Laboratory-Knowledge Discovery in Databases (NSL-KDD) and Australian Defence Force Academy (ADFA) datasets. Studies show that the performance of HIDS is enhanced, compared to SIDS and AIDS in terms of detection rate and low false-alarm rates.

Keywords: anomaly detection; hybrid approach; C5.0 Decision tree; Cyber analytics; data mining; machine learning; Zero-day malware; Intrusion; Intrusion Detection System

1. Introduction

Zero-day intrusion detection is a serious challenge as hundreds of thousands of new intrusions are detected every day and the damage caused by these intrusions is becoming increasingly harmful [1,2] and could result in compromising business continuity. Computer attacks are becoming more complicated and lead to challenges in detecting the intrusion correctly [3].

Intrusion detection systems (IDS) detect suspicious activities and known threats and generate alerts. Intrusions could be identified as any activity that causes damage to an information system [4]. IDS could be software or hardware systems capable of identifying any such malicious activities in computer systems. The goal of intrusion detection systems is to monitor the computer system to detect abnormal behavior, which could not be detected by a conventional packet filter. It is very vital to achieve a high degree of cyber resilience against the malicious activities and to identify unauthorised access to a computer system by analysing the network packets for signs of malicious activity.

IDSs use two broad methodologies for intrusion detection: The Signature-based Intrusion Detection System (SIDS) and Anomaly-based Intrusion Detection System (AIDS). SIDS, also called Knowledge-Based Detection or Misuse Detection, is a process where a signature identifier is determined

about a known malware so that malware can be detected in the future [5]. If that specific signature is identified again, the traffic can be identified as being malware. SIDS usually gives an excellent detection accuracy, particularly for previously known intrusions.

Since SIDS can be as effective as the update of the signature database, three issues arise. Firstly, it is easy to trick signature-based systems through the polymorphic behavior of malware. This method fails the similarity test as it does not match with any signature stored in the IDS database, giving the attacker a chance to gain access to the computer system. Secondly, the higher the number of signatures in the database, the longer it takes to analyses and process the huge volume of data. Thirdly and most importantly, SIDS has difficulty in detecting zero-day malware as the signature is not stored in the database [6].

AIDS systems have overcome the limitation of SIDS and are being used to identify malicious attacks on computer systems. The assumption for this technique is that the profile of a malicious activity differs from typical user behavior activities [7]. AIDS creates a statistical model describing the normal user activity and any abnormal activity that deviates from the normal model is detected. The design idea behind AIDS is to profile and represent the normal and expected standard behavior profile through monitoring activities and then defininining anomalous activities by their degree of deviation from the normal profile. AIDS uses features such as the number of emails sent by a user, the number of failed logins tries for a user, and the degree of processor use for a host in a given timeframe in learning the normal behaviors. Anomaly detection techniques have the ability of strong generalizability and to detect new attacks, while its drawbacks could be in the form of large false alarm rates due to the changing cyber-attack landscape.

The behaviors of alien users are deemed different to the standard activities and are categorized as intrusions. AIDS includes two phases: Thee training phase and testing phase. In the training phase, the normal traffic profile is learned from the data that represent normal behavior, and then the testing is done on a data set that is not seen by the model during the training phase.

AIDS could be classified into several sub-classes based on the learning methods, for instance, statistical based, knowledge based, and machine learning based [8].

The key advantage of AIDS is its ability to identify zero-day attacks as it does not have to rely on the signature database to detect an attack. AIDS triggers an alert signal when the examined behavior differs from the usual activity [9]. Furthermore, AIDS has various benefits: Firstly, it has the capability to discover internal malicious activities. If an intruder starts transacting on a stolen account, which could be misidentified as the normal user's activity, then it generates an alarm. Secondly, it is very difficult for a cybercriminal to learn what a normal user's behavior is without producing alerts, as the system is constructed from customized profiles [4].

Traditional IDSs have limitations: Inability to differentiate new malicious attacks, the need to be updated, low accuracy, and high false alarms. AIDS also has shortcomings such as a high number of false alarms [10]. To overcome those limitations, an innovative IDS model is proposed with the integration of SIDS and AIDS in order to achieve accuracy and to reduce the false alarm. Well known intrusions could be detected by SIDS and new attacks could be detected by AIDS.

In this paper, the signature intrusion detection system is built, based on the C5.0 Decision tree classifier, and the Anomaly intrusion detection system is built, based on one-class Support Vector Machine (SVM). The target is to understand how the intrusion detection system accuracy can be enhanced by utilising the ensemble stacking technique. While the traditional intrusion detection system concentrates on how the performance of a one classifier can be enhanced, this research studies how diverse machine learning techniques can be integrated to enhance intrusion detection accuracy. In this research, two machine learning techniques, namely the decision tree c5 and one class SVM classifier, were chosen to build the intrusion detection system. The decision tree c5 and one class SVM classifier have been evaluated independently and in combination by using stacking ensemble, which is tested as well.

Our paper makes the following contributions:

- Proposes a novel intelligent framework to identify well-known intrusions and zero-day attacks with high detection accuracy and low false-alarm rates.
- Develops a stacked hybrid IDS based on SIDS and AIDS to harness their respective strengths for better detection.
- Evaluates Hybrid Intrsuion Detection System (HIDS) on the Network Security Laboratory-Knowledge Discovery in Databases (NSL-KDD) and Australian Defence Force Academy (ADFA) datasets benchmark datasets in terms of accuracy and F-measure, and studies and validates its superior performance.

The rest of the paper is organized as follows: Section 2 reviews a few related works. Section 3 describes the detailed description of the proposed hybrid intrusion detection. Experimental results are presented in Section 4. Section 5 concludes the paper with a brief discussion and summary.

2. Related Work

There are many IDSs in the literature to identify abnormal activities, but most of these IDSs produce a large number of false positives and low detection accuracy. However, many hybrid IDSs have been proposed to overcome the drawbacks of SIDS and AIDS. Sumaiya et al. proposed an intrusion detection model by applying the chi square attribute extraction and multiclass support vector machine [11]. Syarif et al. used boosting, stacking, and bagging ensemble techniques for IDS to enhance the intrusion detection rate and to reduce false alarms [12]. They used four diverse machine learning techniques: Naïve Bayes, artificial neural networks, decision tree, genetic algorithms, rule induction, and k-nearest neighbor as the foundation classifiers for the ensemble methods. They highlighted that their proposed method has an accuracy of 99% in finding known attacks, but could only identify zero-day attacks at around 60% accuracy rates.

Kim et al. [13] presented a HIDS technique that hierarchically combines SIDS and AIDS models. Signature detection is developed based on the J48 decision tree classifier. Then, various one-class support vector machines models are built for the split subsets, which can reduce the profiling capability.

Muniyandi et al. [14] proposed an anomaly detection method that employs "K-Means + C4.5", for classifying anomalous and normal activities in a computer system. The K-Means clustering method is employed to divide the training data into k number of clusters by using the Euclidean distance similarity. The Hybrid method beats the individual method in terms of accuracy, but it causes high false alarms.

Al-Yaseena et al. [15] proposed a multi-level hybrid IDS which employs SVM and extreme learning machine to enhance the detection efficiency for known and unknown attacks. The system performed well on the KDD Cup 1999 dataset in terms of the false alarm rate, which was 1.87%.

Koc et al. proposed the Hidden Naïve Bayes (HNB) model for IDS issues such as dimensionality, correlated features, and big data stream [16]. The results showed that this method achieved an overall performance that was comparable to the Naïve Bayes model. Sivatha et al. proposed a lightweight IDS to classify abnormal activities in the network using wrapper based feature selection techniques that create good intrusion detection rates by adding the neural ensemble decision tree classifier [17].

M. Alazab proposed a methodology to extract features statically and dynamically from malware such as the Windows Application Programming Interface (API) calls and create a malware behavior profile by extracting malware API calls throughout execution [18].

Table 1 shows the IDS techniques and datasets covered by different intrusion detection system survey papers. Khraisat et al. presented a survey of current intrusion detection systems, which was a wide-ranging review of ID technqiues, and the datasets usually employed for evaluation purposes [4]. Several intrusion detection system techniques that have been developed to improve the detection rate. However, such technques may have difficulty increating and updating the signarture of new malware and in yielding high false alarms or poor detection rates.

Table 1. Comparison of intrusion detection systems (IDS) techniques and datasets issue covered (✓: Topic is covered, ✕ the topic is not covered).

IDS Methods	Intrusion Detection System Techniques						Dataset Issue
	SIDS	AIDS				Hybrid IDS	
		Supervised Learning	Unsupervised	Semi-Supervised Learning	Ensemble Methods		
Lunt [19]	✓	✕	✕	✕	✕	✕	✕
Axelsson [20]	✓	✓	✕	✕	✕	✕	✕
Liao, et al. [21]	✓	✓	✓	✕	✕	✓	✕
Agrawal and Agrawal [22]	✓	✓	✓	✓	✓	✓	✕
Buczak and Guven [23]	✓	✓	✓	✕	✓	✓	✓
Ahmed, et al. [24]	✕	✓	✓	✕	✕	✕	✓
Khraisat, et al. [4]	✓	✓	✓	✓	✓	✓	✓

Ghanem et al. proposed a hybrid detection approach for large datasets using detectors generated based on different machine learning techniques. Anomaly detectors were developed based on self and non-self-training data to obtain self-detectors [25]. K-means that clustering is used to decrease the volume of the training dataset by removing the redundant detector. The key role of AIDS is to create normal profiles of attacks. If the patterns are general in nature, then it is unable to identify several intrusions, which results in a poor detection rate. If the profiles are very specific, then it can identify different intrusions, but several normal behaviors could be classified as attacks. However, none of the previous works have attempted to strike a balance between accuracy and false positives. Recent studies focus on decreasing the false positive rate of AIDS by proposing a hybrid IDS. While earlier studies only integrate the results of both detection models, but in the proposed technique, the intrusion detection systems are hierarchically combined to improve accuracy. This allows the AIDS to improve its normal profiling capability with the use of the signature detection model. The details of the proposed Hybrid IDS are described in Section 3.

3. Hybrid Intrusion Detection System

Hybrid IDS is developed to overcome the disadvantages of SIDS and AIDS as it integrates SIDS and AIDS to detect both unknown and known attacks. In our approach, we used AIDS to identify unseen intrusions, while SIDS is used to identify well-known attacks. Our system is based on three stages process:

- Stage 1: SIDS based C5
- Stage 2: AIDS based One-Class SVM
- Stage 3: HIDS based Stacking Ensemble of C5 and One-Class SVM

Our hybrid IDS (HIDS) ultimately combines the C5 classifier (in the first stage) and One Class Support Vector Machine (in the second stage). The central idea of this novel approach was to combine the advantages of both SIDS and AIDS to build an efficient IDS. The Hybrid IDS has two phases; the SIDS phase and AIDS phase are shown in Figure 1.

Our intrusion detection techniques included online and offline, which means detection intrusion later and online stages, which means detection intrusions in real-time. In the offline stage, the C5 classifier learning method was used to update the signature database. This stage deals with the stored signature and passes it through some processes to decide if it is an attack or not. In the online stage, the initial detection model was created using a one-class SVM. The online IDS deals with the network in real-time. This stage analyses the network traffic to decide if it is an intrusion or not.

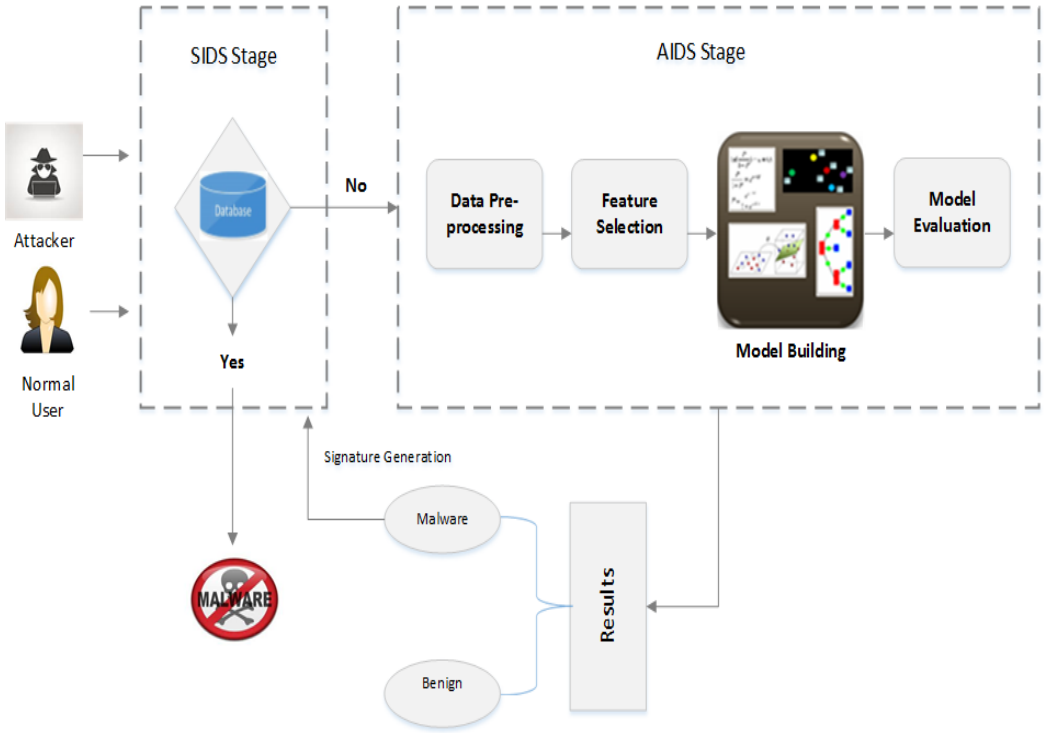

Figure 1. Hybrid intrusion detection system.

AIDS profiles normal user activities and raises a malicious alarm when the difference between a given observation at an instant and the recorded value of the profile exceeds a predefined threshold. User profiles are created from the records generated from user activities and are marked as benign. AIDS feedbacks malicious records to SIDS to be saved in the signature database. The principle reason for storing the intrusion data in the signature database is to mitigate against known intrusions. The performance of the proposed HIDS and its components (SIDS and AIDS) are evaluated by conducting experiments separately. In the following, the two phases of the proposed detection system are elaborated.

3.1. Stage 1: SIDS Based C5

SIDS is used in the first stage as it provides high accuracy in detecting well known intrusions and generates low false alarms. As a result, false positives are low as all known signatures for malicious samples are kept in the database. Therefore, attacks can be detected with high accuracy while reducing false positives.

In SIDS, the C5 classifier is used in this stage to detect well-known intrusions. In our previous work, C5 was analyzed and contrasted with other machine learning techniques [26]. The results revealed that C5 performs very well in terms of the detection rate and false alarm rate. The C5 algorithm is an improved version of the commonly used C4.5 classifier which was developed by Quinlan [27], based on decision tree [26]. It incorporates variable misclassification costs, handles missing data, can handle large numbers of input fields, and builds the model very fast. It takes a set of known data as the input and builds a decision tree from that data. In C5, the decision tree is built in a top-down fashion. The first attribute and its values are at the top of the tree and the next branch leads to either an attribute or outcome. C5 decision trees are created in view of a number of features and a set of training stages, and then the tree could be categorized by using a subsequent set to distinguish other samples.

We have used the C5 classifier for SIDS, as shown in Figure 2. Unknown samples are handled through pattern matching in order to determine whether they represent normal or abnormal activities. If the unknown sample is found in the signature database, then it triggers an alarm that it is a malware.

If no match is found, it will go to AIDS, which is the second stage of the framework, as shown in Figure 2.

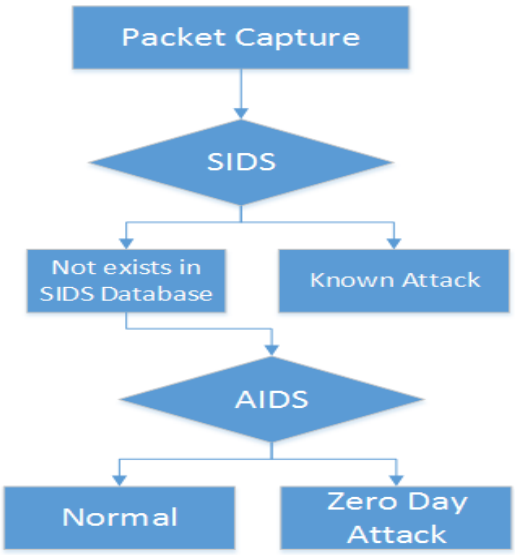

Figure 2. Hybrid intrusion detection system flowchart.

The first stage of the proposed framework results in one of two possible outcomes: Known attacks and unknown samples. At the second stage, the unknown samples (branching out at 'Not Exists' in the figure) are then presented to AIDS for further training and analysis to overcome the shortcoming of SIDS.

3.2. Stage 2: AIDS Based One-Class SVM

To successfully identify new intrusions, the result of the SIDS phase should be used as input data in the AIDS stage to detect new intrusions. AIDS should be built based on the normal activity of a user, which could have been used during the training stage. Then, intrusions are detected based on the measured state of the user profile, which is compared to the normal profile (determined based on the model), if it varies more than the described threshold, then it is marked as a malicious. The one class SVM learning model is used to identify normal behavior, as the One-class SVM (OCSVM), which learns the attributes of benign samples without using any information from the other class. One-class SVM was suggested by Schölkopf et al. [28] to predict the support of a high-dimensional distribution by modifying the SVM method to the one-class problem. It involves the first feature processing through a kernel and then employs relaxation parameters to separate the test point of a class from the rest of the datasets or origin [29], as illustrated in Figure 3. Relaxation parameters techniques are iterative approaches for solving large sparse linear systems. It is also used to solve linear least-squares and nonlinear equations problems. Relaxation parameters help SVM to control the compromise between the reaching of a low detection rate on the training stage and a low detection rate on the testing stage, which is the capacity to identify and classify unknown malwares.

The OCSVM classifier transforms instances into a large dimensional attribute space (via a kernel) and locates the suitable location of the boundary hyperplane, which splits the training data. The OCSVM is a normal binary-class SVM where all the training data are based on the first class. Thus, we consider those profiles to be abnormal, which are close to the origin of coordinates in a feature space. The establishment of the hyperplane needs to follow the categorization rule:

$$f(x) = (w, x) + b \tag{1}$$

where w is the normal vector and b is a bias term. The OCSVM adjusts the hyperplane to find a linear classifier by optimising the rule f. This classification rule can be used to assign a label to a test example x. x is classified as an intrusion if the $f(x)$ result is less than zero, or else it is classified as normal. As presented in Figure 3, the result of $f(x)$ can clarify the classification condition: Positive is considered to in the normal class, negative is in the intrusion class.

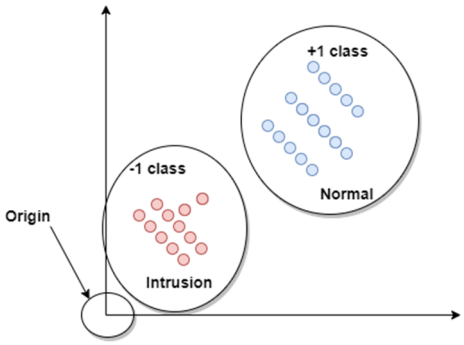

Figure 3. One-class SVM (Support Vector Machine) classifier based on relaxation parameters.

The one class SVM intrusion detection system can be expressed as mapping the data into a feature vector H using a suitable kernel function, and then attempts to isolate the mapped vectors from the origin with a determined margin (see Figure 3).

$$f(x) = \begin{cases} +1, & if\ x \in Normal \\ -1, & if\ x \in Intrusion \end{cases} \tag{2}$$

In stage 2 of the one class SVM, let $x1,\ x2,\ \ldots,\ xl$ be the training examples belonging to one class X, where X is a compact subset of R^N. Let Φ: X $\rightarrow$ H be a kernel map that transforms the training examples to another space. Then, to separate the data set from the origin, one needs to solve the following quadratic programming problem:

$$min\frac{1}{2}\|w\|^2+\frac{1}{Vl}\sum\nolimits_{i=1}^{l}\xi_i - p \tag{3}$$

which is subject to

$$(w \times \Phi(xi)) \geq \rho - \xi i\ i = 1, 2, \ldots, l\ \xi i \geq 0 \tag{4}$$

If w and ρ are solved in this problem, then the decision function

$$f(x) = sign((w \times \Phi(x)) - \rho) \tag{5}$$

will be normal for most instances xi comprised in the training data set.

The basic idea of OCSVM training is to build the OCSVM intrusion detection system that is able to detect the intrusions. Initially, both the training set and the test set are pre-processed to acquire the vector sets based on their unlikely data types. The training vector set at that point is employed to train the OCSVM intrusion detection system and the OCSVM detector is then used on the test vector set. If the return value result for the intrusion system OCSVM function $f(x)$ is less than zero, an intrusion is detected, otherwise it is normal. This entire stage technique is shown in a flowchart in Figure 4.

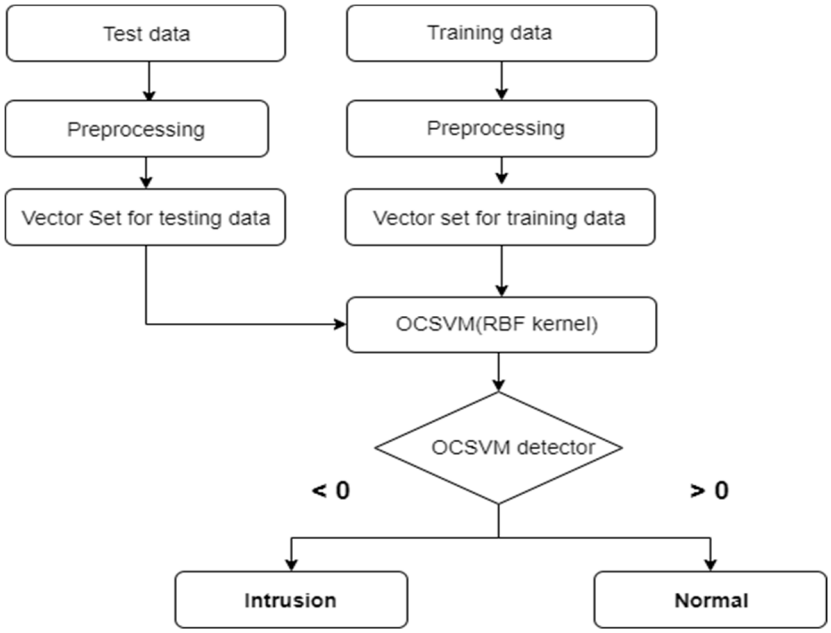

Figure 4. Anomaly-based Intrustion Detection System (AIDS) based one-class svm.

3.3. Stage 3: HIDS Based Stacking Ensemble of C5 and One-Class SVM

SIDS and AIDS have complementary strengths and weaknesses, so we developed a hybrid method using an ensemble of both techniques. In machine learning, ensemble techniques use many learning algorithms to accurately predict the outcome. In other words, different classifier models are trained on the same target and then their results are combined. In the first stage, a set of base level classifiers C1, C2, . . . , Cn are created. In the second stage, a meta-level classifier is built by uniting the base level classifier.

While many ensemble methods have been proposed in the literature, it is a difficult task to find a suitable ensemble configuration to detect zero-day attacks. There are three popular ensemble methods: Bootstrap aggregating, boosting, and stacking. Bootstrap aggregating, known as bagging, employs the simplest way of combining predictions that belong to the same class. For example, if we had four bagged decision trees that made the following class predictions for an input sample: Malware, normal, malware and malware, we would take the most frequent class and predict malware. Boosting steadily creates an ensemble via preparing each new model, utilizing the misclassified training instance that past models misclassified. An example of boosting is the AdaBoost algorithm, which uses a boosting technique. Stacking, also known as stacked generalization, is a technique that combines the other models' predictions.

In stacking, predictions of base learners (stage one) are used as input for the meta-learner (stage two). Stacking is a parallel integration of classifiers in which all the classifiers are implemen-ed parallel to eatchother and learning takes place at the meta-level. In this paper, two models, namely C5 and OCSVM, are built and then the predictions of the primary models are combined, as shown in Figure 5.

The focus is on how to enhance IDS accuracy by employing the stacking approach. Meanwhile, the current conventional data mining approaches focus on how to enhance the performance of a single model. Our work focuses on how different classifiers can be combined to improve the overall performance of IDS. It was also observed that this approach yields better accuracy in the area of intrusion detection, as illustrated in the following section.

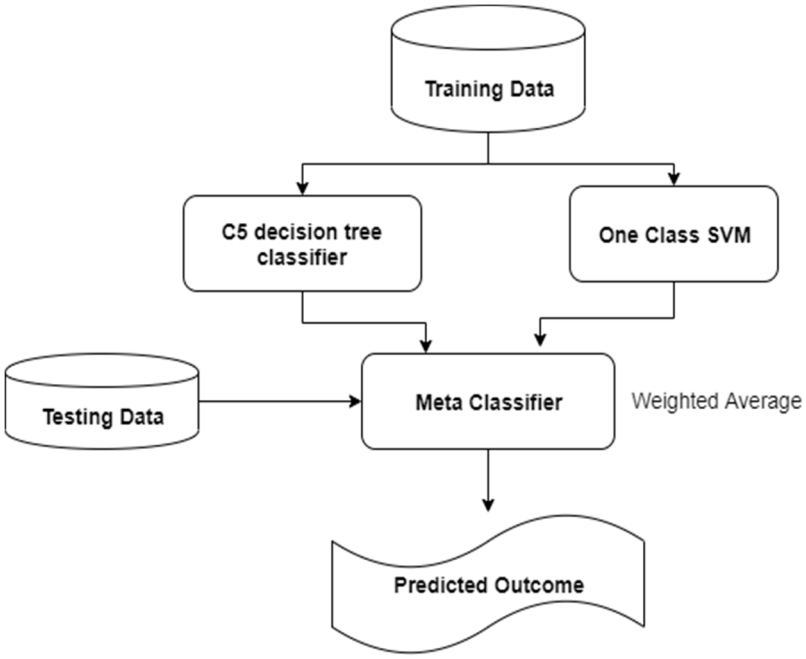

Figure 5. Hybrid Intrusion Detection System based on ensemble of C5 and One-Class SVM.

4. Model Evaluation

The NSL-KDD and ADFA datasets are used to evaluate the proposed hybrid IDS. The experiments have been performed using C5 and LIBSVM, library for Support Vector Machines, implementation of the support vector machine with default parameters. Details of the datasets are presented below.

4.1. Datasets

4.1.1. ADFA Dataset

Creech and Hu [29] built the ADFA Linux (ADFA-LD) cyber security benchmarks datasets for assessment of IDSs. Ubuntu Linux version 11.04 was used as the operating system to collect this dataset. Ubuntu Linux configuration offers several functions including the sharing of files, a database movement system, network settings, and a web server.

File transfer protocol, secure web server, secure shell protocol, and MySQL database are activated based on default ports. Personal Home Page (PHP) was used as a server scripting language and to make the Web pages dynamic and interactive. Apache was installed to enable web-based services. Apache acted as a middleman between the server and user computer.

The ADFA-LD is freely accessible on the Internet and can be found in Reference [29]. Table 2 shows ADFA-LD features and their types.

The ADFA Windows Dataset (ADFA-WD) offers a modern Windows dataset for HIDS evaluation. Table 3 presents various system calls in AFDA-LD and AFDA-WD. Table 4 defines each attack in details in the ADFA-LD dataset.

Table 2. ADFA-LD (ADFA Linux) dataset features.

Seq.	Name of the Category	Description
1	Flow features	It represents the v features communication between the machine to the computer server or server to-client such as source of IP address, source port number and destination IP address.
2	Basic features	It contains the features that describe the communications of protocols such as the connection duration, source to destination transaction bytes and packet count
3	Content features	based on data from packet contents.
4	Time features	It comprises the attribute of time such as the total time taken to send the first packet to the destination as well as the time taken to get the response packet.
5	Generated features	Related with protocols service.
6	Connection features	Built based on the chronological order of the last time feature
7	Labelled Features	It could be normal or intrusion

Table 3. Number of system calls traces in various categories of AFDA-LD and AFDA-WD datasets.

	ADFA-LD		ADFA-WD	
Dataset	**Traces**	**System Calls**	**Traces**	**System Calls**
Training data	833	308,077	355	13,504,419
Validation data	4372	2,122,085	1827	117,918,735
Attack data	746	317,388	5542	74,202,804
Total	5951	2,747,550	7724	205,625,958

Table 4. ADFA-LD attack classes.

Attack	Payload	Description
Hydra-FTP	Password brute force	This type of attack comprises of an attacker trying several passwords or passphrases with the hope of eventually guessing File Transfer Protocol (FTP) password correctly
Hydra-SSH	Password brute force	For guessing Secure Shell (SHH) password.
Add user	Add new super user	Add user command creates a new user
Java Meterpreter	Java based Meterpreter	This is an attack payload that offers a communication shell to the cybercriminal from which they can explore the target computer and execute malicious activities.
Meterpreter	Linux Meterpreter Payload	Client-side poisoned executable
Web shell	C100 Web shell	Web shell is a script running on a server that allows remote access and provides a set of functions to execute or a command-line interfaces on the system that hosts the Web server for the use of cybercriminal.

4.1.2. NSL-KDD Dataset

The KDD 1999 data set has been examined by Tavallaee et al. [30] and found a number of weaknesses. A few problems were noted, relating to synthesizing the network and attack data (after sampling the actual traffic) because of privacy issue, an unidentified packet loss caused by network traffic, and unclear attack definitions. Tavallaee et al. also completed statistical evaluations and revealed a high number of redundant records resulting in bias in the dataset. Hence, high bias can cause IDS to be inaccurate in terms of high false alarms. Therefore, machine learning techniques have

restricted flexibility to learn the true behavior of normal and abnormal activities from the dataset. They proposed a new data set, NSL-KDD, which contains marked records of the overall KDD data set and does not encounter the previously mentioned inadequacies. Table 5 shows the list of attacks presented in NSL-KDD dataset. This dataset has been widely used as a public dataset for the validation of IDS.

Table 5. List of attacks presented in NSL-KDD dataset.

Attack Name	Description
Denial of Service (DoS) attack	Make a computer service unavailable to its legitimate users by overwhelming the computer system. Attacker could send high volume of packet to target computer so normal traffic cannot be handled.
Buffer overflow	Occurs when more data is put into a fixed-length buffer than the buffer can handle.
FTP writes	Add files to ftp directory and eventually gain access to the computer system
Guessing password	Aims to find the correct password by trying systematic guessing of passwords techniques such as dictionary password or brute force attack
IMAP	Internet Message Access Protocol (IMPAP) permits cybercriminal to mount brute force attacks without being locked out or triggering an alert by intrusion detection system
IP Sweep	Occurs when cybercriminal sends Internet Control Message Protocol (ICMP) echo requests (pings) to several computer addresses. If a computer receiver host response, the response discloses the victim's IP address to the cybercriminal.
LAND (Local Area Network Denial)	Occurs when cybercriminal submits TCP SYN spoofed datagram where sender and receiver IPs and ports are be configured the same. When the victim computer attempts to reply, it enters into a loop, repetitively sending responses to itself which ultimately sources the victim machine to damage.
load module	Attacker aims to load two dynamically loadable kernel drivers into operating system to gain root access on the target computer.
Multi hop routing	Occurs when cybercriminal change routing messages to cause wrong routing updates which can ultimately lead to network failure.
Neptune (SYN flood)	Occurs when cybercriminal generates a SYN Flood attack against a network host by sending session establishment packets via a fake source IP address.
Nmap (port scanning)	Occurs when cybercriminal sends packets to target computer to discover what services are running on the target computer
Perl scripting	Occurs when cybercriminal inject malicious code in a target computer
phf script	A script named "phf" which is installed by default in the web server directory could be used to send illegitimate packets to the web server.
Ping of Death (PoD)	Cybercriminal tries to freeze the victim's computer by sending abnormal or large packets using a simple ping command.
Port sweep	In Ports weep attack, various hosts are scanned on a particular listening port. For instance, if the cybercriminal would like to detect all the web servers which are using ports 80 and 443.
Rootkit	This attack is used to obtain root or administrator access
Satan	Satan is a tool designed to probe a target computer system
Smurf	It is a distributed denial-of-service attack in which huge amounts of packets with the intended computer target are tricked as source IP to broadcast to a victim computer network.
Spy	This attack enables to collect data about a target computer without their knowledge
Teardrop	A *teardrop attack* is DoS *attack* which sends fragmented packets to a target computer
Warezmaster	Cybercriminal access on a computer server using guest account. The cybercriminal creates a hidden file and uploads "warez" (malicious file) onto webserver. Victim user can then later download these files.
Warezclient	Warezclient attack could be executed by victim during an FTP connection after warezmaster attack.
Normal	Not attack
Unknown	Unknown attack

4.2. Evaluation Metrics

Table 6 shows the confusion matrix for a two-class classifier that would commonly be used in an IDS. Each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class.

Table 6. Confusion matrix of an IDS for evaluation purpose.

Actual Class	**Predicted Class**		
		Normal	**Intrusion**
	Normal	True negative (TN)	False Positive (FP)
	Intrusion	False Negative (FN)	True positive (TP)

Usually, the evaluation of the IDS is assessed based on the Confusion matrix measurement as follows:

- True Positive Rate (TPR): It measures the quantitative relation between the attacks and the overall attacks number. *TPR* is 1 when all intrusions are correctly identified, and that is extremely rare for an IDS. *TPR* is also called the Detection Rate (DR) and is defined as:

$$TPR = \frac{TP}{TP + FN} \tag{6}$$

- False Positive Rate (FPR): It measures the quantitative relation between the normal cases that are detected as attacks and the overall number of normal cases. *FPR* is calculated as:

$$FPR = \frac{FP}{FP + TN} \tag{7}$$

- False Negative Rate (FNR): FNR shows that the intrusion detection system could not classify the intrusion and has classified it as normal. The *FNR* is calculated as:

$$FNR = \frac{FN}{FN + TP} \tag{8}$$

- Classification rate (CR) or Accuracy: The *CR* is the total accuracy of the IDS in classifying both normal and intrusion attacks and is calculated as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

- Precision (P): *P* is the percentage of total true positives (*TP*) instances divided by total number of true positives (*TP*) and false positives (*FP*) instances:

$$Precision = \frac{TP}{TP + FP} \times 100\% \tag{10}$$

- Recall (R): Refers to the percentage of total relevant results correctly classified, true positives (*TP*), divided by the total true positives and false negatives (*FN*) instances:

$$Recall = \frac{TP}{TP + FN} \times 100\% \tag{11}$$

- F-measure (FM): The *FM* is the mean of the precision and recall. F-Measure is favored when only one accuracy metric is needed as an evaluation measurement:

$$\text{F} - \text{measure} = \frac{2 * \text{Re}call * \text{Pr}ecision}{\text{Re}call + \text{Pr}ecision} \tag{12}$$

4.3. Experimental Results

The effectiveness of our proposed model is evaluated with other machine learning techniques that use the same datasets mentioned earlier.

In the first instance, for selected classification techniques, the dataset is divided into training and testing subsets for assessment purposes. For the NSL-KDD dataset, the process of training and testing the different stages is outlined in Table 7.

Table 7. Steps followed for training and testing the NSL-KDD dataset.

Steps	Details
Step 1	For NSL-KDD, we divided the data for training and testing. KDDTrain+, which is generated from KDD train set, is used as training set and KDDTest+ is used as testing set. The generated datasets, KDDTrain+ and KDDTest+, contain 125,973 and 22,544 records, respectively.
Step 2	Trained the SIDS using KDDTrain+; and then tested it using KDDTest+.
Step 3	Finally, tested the SIDS using KDDTest+. If SIDS classifies a sample as malware, then labeled it as malware. But, if SIDS classifies as normal, then those samples are passed to AIDS for further analysis
Step 4	Trained AIDS with KDDTrain++ and tested the samples labelled as normal by SIDS

With the ADFA dataset, we used the widely adopted 10-fold cross-validation scheme for training and testing purpose. In a 10-fold cross-validation, the dataset is split into 10 approximately equal sized non-overlapping subsets. Nine subsets are used for building the classifier in the training stage, while the remaining one subset is used to test the model. The test set is employed to estimate the IDS accuracy. This process is repeated 10 times, each time using a separate fold for testing. In this way, the whole dataset goes through the testing phase in turns, with each sample being tested once. The overall accuracy estimate is the mean of 10 rounds.

The proposed IDS accuracy has been evaluated for all stages; four statistics evaluation measurement have been computed: True positive rate, F-measure, false positive rate, and accuracy.

4.3.1. Stage One: SIDS Results

This experiment was conducted by using NSL-KDD Test+ dataset and ADFA dataset. To evaluate the performance of the proposed technique, the Confusion matrix was used. The Confusion matrix results for the C5 classifier in stage one is shown in Table 8 for both NSL-KDD Test+ and ADFA.

Table 8. Confusion matrix results with the use C5 classifier on KDDTest+ and ADFA dataset.

Dataset	**NSL_KDD**		**ADFA**	
Class	**Normal**	**Malware**	**Normal**	**Malware**
Normal	9488	263	36,065	935
Malware	3900	8933	1250	44,082

The detailed accuracy for C5 classifier with the use of NSL-KDD Test+ is shown in Table 9 and on ADFA dataset shown in Table 10.

Table 9. Detailed accuracy of C5 decision tree with KDDTest+.

Class	TP Rate	FP Rate	F-Measure
Normal	0.972	0.311	0.815
Malware	0.689	0.028	0.805

Table 10. Detailed accuracy of C5 decision tree on ADFA dataset.

Class	TP Rate	FP Rate	F-Measure
Normal	0.975	0.028	0.971
Malware	0.972	0.025	0.976

4.3.2. Stage Two: AIDs Results

One-class SVM with an RBF kernel was applied using LIBSVM. Confusion matrix results are shown in Table 11 for both datasets: NSL-KDD Test+ and ADFA.

Table 11. Performance of the One-Class Support Vector Machine.

Dataset	NSL-KDD		ADFA	
Class	**Normal**	**Malware**	**Normal**	**Malware**
Normal	9500	211	33,079	3921
Malware	6064	6769	15,554	29,778

The detailed analyses of the accuracy of the One-Class SVM classifier on NSL-KDD Test+ and ADFA datasets are highlighted in Tables 12 and 13, respectively. For AIDS, the detection accuracy is 72.17% with the use of NSL-KDD Test+ dataset and 76.4% for the ADFA dataset.

Table 12. Performance of the One-class SVM on the NSL-KDD Test+.

Class	TP Rate	FP Rate	F-Measure
Normal	0.978	0.473	0.752
Malware	0.527	0.022	0.683

Table 13. Performance of the One-class SVM on the ADFA dataset.

Class	TP Rate	FP Rate	F-Measure
Normal	0.894	0.343	0.773
Malware	0.657	0.106	0.754

4.3.3. Stage Three: Combination of the Two Stages

The Confusion matrix of the mixture of both classifiers in Stage three is shown in Table 14 for both NSL-KDD Test+ and ADFA.

Table 14. Confusion matrix results for Stage 3.

Dataset	NSL-KDD		ADFA	
Class	**Normal**	**Malware**	**Normal**	**Malware**
Normal	9500	211	36,114	886
Malware	6064	6769	1305	44,027

The accuracy of Stage 3 with the use of NSL-KDD Test+ and ADFA datasets is shown in Tables 15 and 16, respectively.

Table 15. Detailed accuracy at Stage 3 on the KDDTest+.

Class	TP Rate	FP Rate	F-Measure
Normal	0.972	0.273	0.833
Malware	0.727	0.028	0.832

Table 16. Detailed accuracy at Stage 3 on the ADFA dataset.

Class	TP Rate	FP Rate	F-Measure
Normal	0.976	0.029	0.971
Malware	0.971	0.024	0.976

As revealed in Figure 6, the detection accuracy of the intrusion is 81.5% with the use of NSL-KDD Test+ dataset and 97.3% for the ADFA dataset at Stage one. Meanwhile, the detection accuracy of the intrusion is 72.2% with NSL-KDD Test+ dataset and 76.4% for ADFA dataset at Stage two. At Stage 3, the accuracy rates are improved to 83.2% and 97.4%, respectively, for the both datasets. Results show that our suggested framework yields a superior detection rate and a lower false alarm rate, as compared with the single stage method.

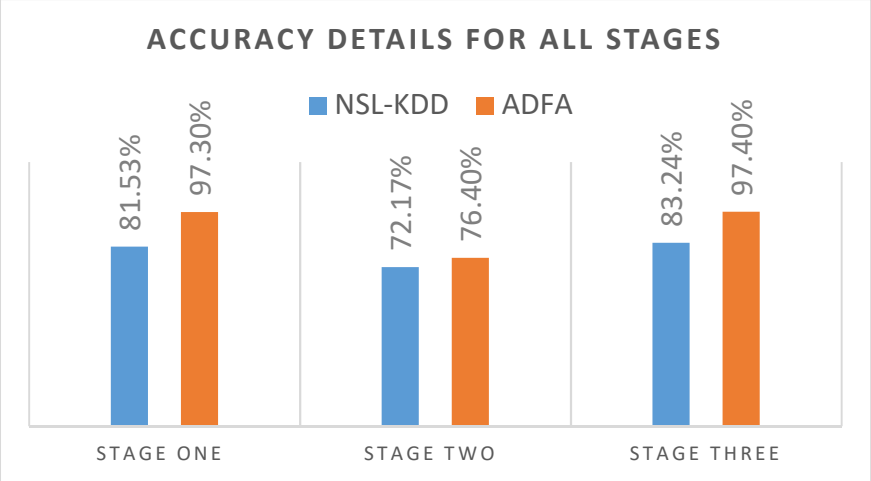

Figure 6. Accuracy details for all stages.

To further analyze performance, our approach is compared with different approaches reported in the literature in terms of the overall accuracy. Table 17 shows this comparison for the NSL-KDD dataset. Results show that our proposed model, which combines two stages, outperforms other approaches.

Table 17. Accuracy comparison of the proposed model on NSL-KDD (Test+ dataset).

Research	Accuracy Rate on KDDTest+
Abbasi, et al. [31]	77.38%
Panda, et al. [32]	81.47%
Abbasi, et al. [31]	79.66%
Proposed Technique	83.24%

According to the results shown in Figure 6, the accuracy obtained by our proposed algorithm on the KDDTest+ and ADFA datasets are supreme, as compared to the accuracy obtained by different

classifiers to obtain the accuracy. Table 18 shows the accuracy rates for different machine learning techniques, specifically C4.5, Naïve Bayes, Random Forest, multi-layer perception, SVM, CART, and KNN on the NSL-KDD dataset. The results show that our proposed technique, which combined the two stages, achieved the best performance, reaching an accuracy of 83.24%.

Table 18. Performance comparison between different classifiers and proposed algorithm on KDDTest+.

Machine Learning Techniques	NSL-KDD Accuracy
C4.5 [26]	81%
Naïve Bayes	76.56%
Random Forest	80.67%
Multi-layer perception	77.41%
SVM	69.52%
CART	80.3%
KNN	79.4%
Proposed Technique	83.24%

5. Conclusions

To create attacks in high volume, cybercriminals began using new techniques, like polymorphism, to change the signature each time and to generate new attacks. Efficient IDSs should be able to detect known and zero-day attacks reliably. In this paper, a novel framework is developed to build an intelligent IDS that overcomes the weaknesses of current IDSs, which means including detection methods for both known and unknown threats. The main contribution of our framework is the integration of the signature and anomaly intrusion detection systems, which takes advantage of the respective strengths of SIDS and AIDS. In the proposed IDS, signature-based IDS is applied to identify previously known intrusions, while an anomaly-based IDS is applied to detect unknown zero-day intrusions. We have effectively created signatures from anomaly IDSs to identify different intrusions to add in signature databases. Additionally, the advantage of the proposed IDS is not only the higher detection rate, but also the enhanced scalability, such as when new intrusions are stored to the signature intrusion database. We used the C5 classifiers to create an intrusion signature, which is capable of generating a rule pattern more rapidly and can detect the intrusions with fewer numbers of signatures. We have shown that an ensemble of the C5 classifier (signature) and one-class SVM (anomaly) can result in a better detection rate when compared with other machine learning techniques in terms of the detection rate, false alarms, true negative, false positive, false negative, recall, precision, specificity, sensitivity, and F-Measure. Compared to the single algorithms, combining multiple algorithms has given much better results. Our Hybrid IDS has shown superior results, as compared to existing techniques. Our future research will be focused on the way in which to apply this technique in order to improve the accuracy of IDSs in detecting different attacks.

Author Contributions: A.K. is the main author of the current paper. A.K. contributed to the development of the ideas, design of the study, theory, result analysis, and article writing. A.K. also designed the experiments and then performed the experiments. I.G., P.V., J.K. and A.A. undertook the revision works of the paper. All authors have read and agreed to the published version of the manuscript.

Acknowledgments: This work was done in the Internet Commerce Security Lab (ICSL) FedUni. Westpac Bank, ACSC, Victorian Government and IBM are partners in the ICSL.

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

References

1. Sun, X.; Dai, J.; Liu, P.; Singhal, A.; Yen, J. Using Bayesian Networks for Probabilistic Identification of Zero-Day Attack Paths. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 2506–2521. [CrossRef]
2. Alazab, M.; Tang, M. *Deep Learning Applications for Cyber Security*; Springer: Berlin/Heidelberg, Germany, 2019.

3. Alazab, A.; Hobbs, M.; Abawajy, J.; Khraisat, A. Malware detection and prevention system based on multi-stage rules. *Int. J. Inf. Secur. Priv.* **2013**, *7*, 29–43. [CrossRef]
4. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity* **2019**, *2*, 20. [CrossRef]
5. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J.; Alazab, A. A Novel Ensemble of Hybrid Intrusion Detection System for Detecting Internet of Things Attacks. *Electronics* **2019**, *8*, 1210. [CrossRef]
6. Alazab, A.; Hobbs, M.; Abawajy, J.; Khraisat, A.; Alazab, M. Using response action with intelligent intrusion detection and prevention system against web application malware. *Inf. Manag. Comput. Secur.* **2014**, *22*, 431–449. [CrossRef]
7. Alazab, A.; Hobbs, M.; Abawajy, J.; Alazab, M. Using feature selection for intrusion detection system. In Proceedings of the 2012 International Symposium on Communications and Information Technologies (ISCIT), Gold Cost, Australia, 2–5 October 2012; pp. 296–301.
8. García-Teodoro, P.; Díaz-Verdejo, J.; Maciá-Fernández, G.; Vázquez, E. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Comput. Secur.* **2009**, *28*, 18–28. [CrossRef]
9. Huda, S.; Abawajy, J.; Alazab, M.; Abdollalihian, M.; Islam, R.; Yearwood, J. Hybrids of support vector machine wrapper and filter based framework for malware detection. *Future Gener. Comput. Syst.* **2016**, *55*, 376–390. [CrossRef]
10. Alazab, A.; Abawajy, J.; Hobbs, M.; Khraisat, A. Crime toolkits: The current threats to web applications. *J. Inf. Priv. Secur.* **2013**, *9*, 21–39. [CrossRef]
11. Sumaiya Thaseen, I.; Aswani Kumar, C. Intrusion detection model using fusion of chi-square feature selection and multi class SVM. *J. King Saud Univ. Comput. Inf. Sci.* **2017**, *29*, 462–472. [CrossRef]
12. Syarif, I.; Zaluska, E.; Prugel-Bennett, A.; Wills, G. *Application of Bagging, Boosting and Stacking to Intrusion Detection*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 593–602.
13. Kim, G.; Lee, S.; Kim, S. A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. *Expert Syst. Appl.* **2014**, *41*, 1690–1700. [CrossRef]
14. Muniyandi, A.P.; Rajeswari, R.; Rajaram, R. Network anomaly detection by cascading k-Means clustering and C4. 5 decision tree algorithm. *Procedia Eng.* **2012**, *30*, 174–182. [CrossRef]
15. Al-Yaseen, W.L.; Othman, Z.A.; Nazri, M.Z.A. Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system. *Expert Syst. Appl.* **2017**, *67*, 296–303. [CrossRef]
16. Koc, L.; Mazzuchi, T.A.; Sarkani, S. A network intrusion detection system based on a Hidden Naïve Bayes multiclass classifier. *Expert Syst. Appl.* **2012**, *39*, 13492–13500. [CrossRef]
17. Sivatha Sindhu, S.S.; Geetha, S.; Kannan, A. Decision tree based light weight intrusion detection using a wrapper approach. *Expert Syst. Appl.* **2012**, *39*, 129–141. [CrossRef]
18. Alazab, M. Profiling and classifying the behavior of malicious codes. *J. Syst. Softw.* **2015**, *100*, 91–102. [CrossRef]
19. Lunt, T.F. Automated audit trail analysis and intrusion detection: A survey. In Proceedings of the 11th National Computer Security Conference, Baltimore, MD, USA, 17–20 October 1988.
20. Axelsson, S. *Intrusion Detection Systems: A Survey and Taxonomy*; Technical Report; Department of Computer Engineering, Chalmers University of Technology: Gothenburg, Sweden, 2000.
21. Liao, H.-J.; Lin, C.-H.R.; Lin, Y.-C.; Tung, K.-Y. Intrusion detection system: A comprehensive review. *J. Netw. Comput. Appl.* **2013**, *36*, 16–24. [CrossRef]
22. Agrawal, S.; Agrawal, J. Survey on anomaly detection using data mining techniques. *Procedia Comput. Sci.* **2015**, *60*, 708–713. [CrossRef]
23. Buczak, A.L.; Guven, E. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 1153–1176. [CrossRef]
24. Ahmed, M.; Naser Mahmood, A.; Hu, J. A survey of network anomaly detection techniques. *J. Netw. Comput. Appl.* **2016**, *60*, 19–31. [CrossRef]
25. Ghanem, T.F.; Elkilani, W.S.; Abdul-kader, H.M. A hybrid approach for efficient anomaly detection using metaheuristic methods. *J. Adv. Res.* **2015**, *6*, 609–619. [CrossRef]
26. Khraisat, A.; Gondal, I.; Vamplew, P. An Anomaly Intrusion Detection System Using C5 Decision Tree Classifier. In Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, Melbourne, Australia, 3–6 June 2018; pp. 149–155.

27. Quinlan, J.R. *C4. 5: Programs for Machine Learning*; Elsevier: Amsterdam, The Netherlands, 2014.
28. Schölkopf, B.; Platt, J.C.; Shawe-Taylor, J.; Smola, A.J.; Williamson, R.C. Estimating the support of a high-dimensional distribution. *Neural Comput.* **2001**, *13*, 1443–1471. [CrossRef] [PubMed]
29. Creech, G.; Hu, J. A Semantic Approach to Host-Based Intrusion Detection Systems Using Contiguousand Discontiguous System Call Patterns. *IEEE Trans. Comput.* **2014**, *63*, 807–819. [CrossRef]
30. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009; pp. 1–6.
31. Abbasi, A.; Wetzels, J.; Bokslag, W.; Zambon, E.; Etalle, S. On Emulation-Based Network Intrusion Detection Systems. In Proceedings of the Research in Attacks, Intrusions and Defenses: 17th International Symposium, RAID 2014, Gothenburg, Sweden, 17–19 September 2014; Stavrou, A., Bos, H., Portokalidis, G., Eds.; Springer: Cham, Switzerland, 2014; pp. 384–404. [CrossRef]
32. Panda, M.; Abraham, A.; Patra, M.R. Discriminative multinomial Naïve Bayes for network intrusion detection. In Proceedings of the 2010 Sixth International Conference on Information Assurance and Security, Baltimore, MD, USA, 23–25 August 2010; pp. 5–10.