Host-based intrusion detection system using Support Vector Machine

Saifullah Memon[1,*], Asghar Ali[2], Waqas Ali[3], Muhammad Awais Rajput[4], Abbas Ali Ghoto[5], Muhammad Aamir Bhutto[6],

[1]State Key Laboratory of Networking and Switching Technology, BUPT, Beijing, China
[2]School of Engineering and Information Technology, UNSW, Australia
[3]School of Information Engineering, Yangzhou University, Jiangsu, China
[4]Department of Artificial Intelligence, QUEST, Nawabshah, Pakistan
[5]Department of Mathematics & Statistics, QUEST, Nawabshah, Pakistan
[6]Department of Computer System Engineering, QUEST, Nawabshah, Pakistan
[*]Corresponding author: memonsaifullah@bupt.edu.cn

Abstract

In line with the communication industry's use of recent advancements in network technology to link remote areas of the world, attackers or intruders have stepped up their attacks on networking infrastructure. System administrators might deploy intrusion detection tools and systems to thwart such efforts. In recent years, the use of machine learning (ML) techniques in intrusion detection systems (IDSs) has increased. One of the most popular machine learning (ML) techniques for intrusion detection is the Support Vector Machine (SVM) due to its excellent generalization and capacity to escape the dimensionality curse. Recent studies have shown that the number of dimensions still impacts how well SVM-based intrusion detection systems work. The fact that SVM assesses all data characteristics equally has also caused some concerns. Actual intrusion detection datasets include a lot of redundant or superfluous characteristics. It would be ideal to consider feature weights while training an SVM. Knowledge Discovery in Databases (KDD) intrusion detection dataset offers labeled data for the scientists and researchers; choosing the essential features or patterns from the input dataset makes the problem more straightforward and faster and acquires much more accuracy towards threat detection. Our work demonstrates the efficiency of recognizing the essential input patterns to design a more efficient Intrusion Detection System (IDS). Consequently, removing irrelevant or unimportant inputs makes the problem of detecting a threat simpler, faster, and more accurate. It has been an essential issue in intrusion detection that features selection and ranking must be made accordingly; it is the only way to detect intrusion accurately and efficiently. We implement the procedure to remove one feature at a time to run experiments on a Support Vector Machine (SVM) to grade the significance of the features for the KDD dataset. It has been observed that SVM-based IDSs utilizing fewer features could improve and efficiently perform.

Keywords—Intrusion Detection System, Support Vector Machine, DoS, Knowledge Discovery in Databases, Hybrid intelligent system, Decision trees

1 Introduction

Over the past several years, as information technology has grown in popularity and reach, there have been substantial trade-offs among its benefits. As a result of the continual danger of black hats, there has been a greater emphasis on network security. There has been a massive surge in network assaults over the previous decade. These assaults were very intricate and deadly. Dozens of thousands of computer hackers probe and assault various computer networks on a daily basis. Ping sweeps, which are completely innocuous, may range all the way up to more complex methods of exploiting security vulnerabilities [1]. Many computer security approaches, such as encryption, firewalls, and intrusion detection systems (IDS), have been explored in the recent decade to guard against various cyber assaults and computer vi-ruses. Intrusion detection has shown to be the most promising of these strategies for combating complicated and dynamic intrusion behaviors [2]. Any action that attempts to undermine the integrity, confidentiality, or availability of the resource is considered interference.

ISSN: 2523-0379 (Online), ISSN: 1605-8607 (Print)
DOI: https://doi.org/10.52584/QRJ.2001.07

The Intrusion Detection System (IDS), like other security tools (antiviruses, firewalls, etc.), is designed to improve the security of information and communication systems. To put it another way, an intrusion detection system, or IDS, is a tool, often a computer system, that keeps an eye on the goings-on in order to identify potentially dangerous or unsettling alerts. IDS functions in a manner quite similar to that of a spam filter in that it generates an alert whenever certain events take place[3]. Several intrusion detection algorithms have recently been introduced, with statistical methods, knowledge-based methods, data mining methods, and machine learning-based methods being the most popular. Statistical-based techniques are used to record network traffic activity, and a profile is developed that describes its random behavior. Due to the absence of an intelligent learning model, statistical anomaly detection might result in an excessive number of false alarms or an inability to identify potential dangers effectively. Prior understanding of user behavior is essential for knowledge-based approaches. However, Knowledge-based intrusion detection systems use if-then logic to encapsulate an expert's Knowledge of known attack patterns and system weaknesses. The process of getting these rules is time-consuming and error-prone. There are a lot of problems with statistical and cognitive methods, which is why there has been a lot of interest in employing techniques from machine learning to automate the process of pattern learning. Different academic institutions have conducted research on a number of different machine learning strategies, including neural networks, decision trees, and Bayesian net-works, in order to build IDS better. The performance of the Support Vector Machine for IDS is investigated in this research. Since networks are becoming increasingly vulnerable to potential cyber threats, firewalls, security policies, identification systems, and access controls might not be adequate to offer total security because all these security technologies are prone to various system errors. IDS are defense-line guarding computer systems and networks from abuse. Thus IDS are being used as dedicated assistants to the network security infrastructure. Detecting features of an attack is a crucial component of intrusion detection systems because it has to match those features against thousands of patterns in a short period. Thus IDS are used for identifying and correcting vulnerabilities. These are also used for tracking, tracing, and prosecution of intruders [4]–[7]

1.1 Problem Statement

Detection and identification of DoS threats requires efficient machine learning techniques, and machine learning-based training with a given dataset with a large number of features provides high classification accuracy; however, it is

- Time-consuming
- Computation-intensive
- More complex

Therefore, identification of a reduced and optimal feature set is required; however smaller feature set for machine learning considering the following objectives is needed.

- High accuracy
- Detecting a large number of DoS attacks

1.2 Research Objective

The main objective of our research is to secure the system by detecting unauthorized access and protecting the system from being exploited by attackers. To keep the system responsive to legitimate users under a DoS attack, we will be using machine learning techniques to learn our classifier's patterns of malicious activities.

1.3 Proposed Solution

Supervised learning would be implemented to detect a threat's pattern to provide safety and security. We will use the Support Vector Machine (SVM) based on super-vised learning to train a classifier with specific features. Experimenting with seven DoS attacks (extendable to a higher number of threats) for classification with SVM, the KDD (Knowledge Discovery and Data-mining) dataset would be utilized for offline training and cross-validation.

1.4 Unsupervised Learning

Machine learning is an area of artificial intelligence concerned with creating and developing algorithms that en-able computers to construct behaviors based on experimental data such as sensor or database data. Machine learning research aims to uncover complicated patterns and make informed judgments based on data. Search engines, medical diagnostics, handwriting and language recognition, picture verification, load forecasting, marketing, and sales diagnostics are just a few fields in which machine learning is applied. In 1994, ML was employed for the first time in intrusion detection to categorize Internet traffic [8]. It is the foundation for most ma-chine-learning-based Internet traffic classification techniques.

1.5 Machine Learning Strategies Employed in Intrusion Detection Systems

The expert system (ES) [9] is a rule-based method that was widely employed in the early stages of IDSs. Human experts' experience is represented in such systems as a set of rules. This offers management more practical understanding than human specialists in frequency, consistency, and breadth of identifying behaviors associated with well-established patterns of abuse and assault. ES, on the other hand, is less flexible and resilient. Rather than human specialists, data mining techniques derive association rules and repeating occurrences from available data sets. It creates prediction models using statistical approaches to find precise relationships between data elements. They made a data mining framework for intrusion detection based on the obtained criteria [7]. System usage activities are precisely monitored and evaluated to build rules for detecting misuse assaults. The disadvantage of such frameworks is that they tend to generate many rules, increasing the system's complexity. Decision trees are one of the most extensively utilized supervised learning algorithms in IDS because of their simplicity, high detection accuracy, and quick adaptability [10]. Artificial Neural Networks (ANNs) are another useful tool for representing linear and nonlinear processes. The resultant model can calculate the likelihood of provided data matching the traits it has been taught to identify. Later ANN-based IDS [11] identified complicated assaults effectively. Data clustering approaches may be used for unsupervised intrusion detection [12]. Because these approaches rely on calculating a distance between numeric characteristics, they cannot handle symbolic properties quickly, resulting in inaccuracy. IDS uses an-other well-known ML approach: the Naive Bayes classifier [10]. Because Naive Bayes specifies that data features be conditionally independent, which is not usually the case for intrusion detection, the related characteristics might diminish its efficacy. Support vector machines (SVMs) are a viable rival to intrusion detection systems [13], [14], such as conventional decision trees and ANN, since they can offer real-time detection capabilities and manage massive data dimensions. With SVM, training matrices are represented as high-dimensional feature spaces, with each matrices' class represented by a nonlinear mapping. The data is then categorized by creating a collection of support vectors from the training input, which specifies the hyperspace attribute. The following are the two contributions that we have made to this research. The first half of this article describes the state-of-the-art in intrusion detection using SVM and then looks at the resources that various re-searchers in this area have employed. Second, it proposes a novel approach to selecting the best features for detecting intrusion after multiple experiments with varying subsets of features out of 41. It has been determined that, with less time and computer power, we can categorize DoS attacks using just 25 characteristics as opposed to 41 features. So concentrating on fewer characteristics saves time and computational resources.

1.6 Support Vector Machine

The Support Vector Machine (SVM) is already well-known for being the most effective method for learning binary classification, and it uses supervised machine learning. The support vector machine (SVM), which was first developed as a pattern classifier based on a statistical learning technique for classification and regression using several kernel functions, is currently used for a wide variety of pattern recognition applications. It is a supervised machine learning algorithm, which means that in order for it to train well, it requires labeled data. SVM is limited to just being able to conduct binary-class classification, while intrusion detection requires multi-class classification[4], [15]. This is due to the intrinsic structural limitation of a binary classifier that is present in SVM. Despite modest gains, the number of dimensions still impacts the SVM-based classifier's performance[16]. SVM treats all data features equally. Many characteristics in genuine intrusion detection datasets are redundant or unimportant. Feature weights should be considered during SVM training [17]. For the IDS do-main, SVM training takes a long time and needs a lot of dataset storage. As a result, SVM is computationally cost-ly in resource-constrained ad hoc networks [18]. SVM also necessitates processing raw information for classification, which adds to the architecture's complexity and reduces the accuracy of intrusion detection [6] It has also been employed for information security and intrusion detection recently. Because of their outstanding generalization and ability to overcome the dimensionality curse, Support Vector Machines have become one of the most preferred algorithms for anomalous intrusion detection. Another advantage of SVM is that it aids in discovering a global minimum of real risk via structural risk reduction. It can generalize effectively using kernel methods even in high-dimensional areas with few training samples. The SVM can choose proper configuration settings because it does not rely on conventional empirical hazards like neural networks. One of the key benefits of utilizing

SVM for IDS is its speed since the capacity to identify intrusions in real-time is quite valuable. Because the classification difficulty is independent of the depth of the feature space, SVMs may learn a more extensive range of patterns and scale better. When a new pattern appears during classification, SVMs dynamically update the training patterns [19]. The Support Vector Machine (SVM) is a tool primarily used in machine learning and falls in the supervised learning class. It is also applicable to classification and regression and is based on statistical learning theory developed by Vladimir Vapnik at ATT Bell Laboratories in mid-1995. It depends on the Structural Risk Minimization Principle. It is classified by constructing an optimal splitting hyperplane that optimally divides two sorts of data by making two distinct classes. In the case of regression, it optimally accomplishes linear regression. There-fore, SVMs are learning systems used in high dimensional feature space using hypothesis space of linear functions. It is mainly used for problems that have two-group classification. A unique and significant feature of its approach is that the solution depends on data points at the margin. Those points are known as vectors. Linear SVM is extendable to nonlinear, using a group of nonlinear basis functions once the problem is trans-formed into a feature space. The data points could be segregated linearly in the feature space, which could be immensely high dimensional. This is the essential ad-vantage of SVM. It is not compulsory to implement this transformation to judge the splitting hyperplane in high dimensional feature space; instead, a kernel representation could be used, in which the solution is represented as a weighted sum of the values of a specific kernel function assessed at the support vectors, this is the principal benefit of SVM. Therefore, supervised learning as implemented by SVM methods produces output and input mapping functions through a group of labeled training data. To generate a classifier, SVM uses a hyper-linear separating plane.

2 Related Work

Because basic SVM cannot be employed for the IDS area because of the aforementioned flaws, numerous writers have proposed SVM framework variations to alleviate the issue. Here are some of the works that are linked. Peddabachigari et al. investigated the performance of SVM and Decision Tree as solo detectors and hybrids in an empirical study[20]. Both a hierarchical model (DT-SVM), with the DT as the first layer to produce node in-formation for the SVM in the second layer, and an ensemble model, which comprised the standalone techniques as well as the hierarchical hybrid, were investigated. The DT was used as the first layer of the hierarchical model. While using the ensemble method, the detection rate of each specific assault type during training is taken into consideration when assigning a weight to each individual technique. After that, when the system is put to the test, the method that is picked to output the classification is restricted to having the highest weight for the particular attack prediction. The strategies were evaluated for effectiveness on the KDD Cup '99 data set. Heba F. et al. [21] proposed an intrusion detection system that used Principal Component Analysis (PCA) and Support Vec-tor Machines (SVMs) to pick the best feature subset. Several tests on the NSL-KDD dataset were used to verify the efficacy and practicality of the proposed IDS system. T. Shon and Moon suggested a Machine Learning Model combining supervised and unsupervised learning ad-vantages that used a modified Support Vector Machine (SVM) [22]. Furthermore, a basic feature selection ap-proach utilizing GA is offered to pick more acceptable packet fields. In his article, KyawThetKhaing suggested an upgraded SVM Model using Recursive Feature Elimination (RFE) and the k-Nearest Neighbor (KNN) approach to accomplish a feature ranking and selection job for the new model[23]. In an ad hoc network, J.F Joseph et al. developed an in-dependent host-based ID for identifying sinking behavior [24], [25]. To increase detection accuracy, the suggest-ed detection system employs a multi-layer technique. SVM is used to train the detection model in order to improve detection accuracy even further. SVM, on the other hand, is computationally costly for ad hoc network nodes with low resources. As a result, the suggested IDS preprocesses the training data to decrease SVM's computational cost. Predefined association functions are used to reduce the number of features in the training data. Furthermore, the suggested IDS employs a linear classification approach known as Fischer Discriminant Analysis (FDA) to filter out data containing poor information (en-tropy). SVM is now possible in ad hoc network nodes because of the aforesaid data reduction methods. To identify intrusions, R. C. Chen et al. employed RST (Rough Set Theory) and SVM (Support Vector Ma-chine)[4], [5]. The data is first preprocessed, and the dimensions are reduced using RST. The RST selected features are then submitted to the SVM model for learning and testing. The technology substantially reduced data density in space. To solve the challenge of feature selection, many strategies have been used. Some approaches assess the "goodness" of a feature set based on the predicted accuracy of a classifier. Others, on the other hand, calculate the importance of a collection of

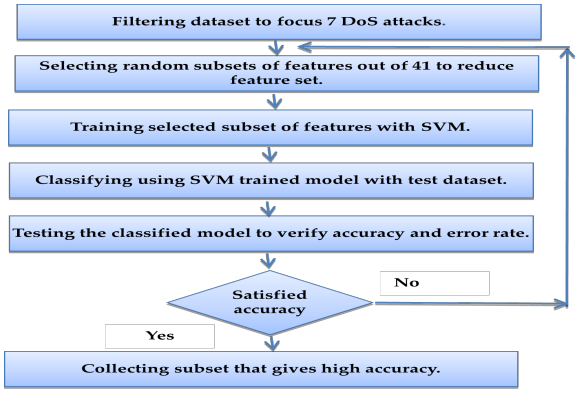

Fig. 1: Proposed System Architecture

attributes using metrics like information, consistency, or distance. These methods have a number of flaws. The first important problem is that providing the classifier with random characteristics might lead to biased results. There-fore, we cannot use the classifier's predicted accuracy as a metric to pick features. A second disadvantage is that given a collection of N features, testing all conceivable feature combinations (2N Combinations) to find the optimal combination to feed the classifier is not a viable strategy.

3 Proposed Optimization

This study gives a comprehensive approach for selecting the most optimal collection of features from the NSL-KDD dataset in order to accurately describe normal traffic and differentiate it from ab-normal traffic using a Support vector machine.

The NSL-KDD dataset is mined for its most relevant characteristics using the method that has been suggested. After that, the NSL-KDD dataset with its reduced features is used to train and develop the detection model for the SVM classifier. The following elements make up the framework of the model that is being proposed:

3.1 Filtering Dataset

Since the KDD dataset is an extensive database of connection vectors, each of these vectors represents a series of packets and signifies either a normal or an attack. The dataset contains 23 different attacks, but we focus on only 7 DoS attacks; we extract only those connections for training and classification on the SVM classifier.

3.2 Selecting Reduced Feature Set

Since the KDD dataset has an overall 41 features, which gives high accuracy while training and testing the SVM classifier, we are looking for a feature set simpler than 41, which may provide us with as much accuracy as 41. Keeping this hypothesis in mind, we are selecting a reduced subset of features randomly out of 41.

3.3 Training Classifier with a Selected Subset of Features

Once we have selected a reduced subset of features, it is divided into two equal parts, one for training and the other for classification. This partition is done through MATLAB's function named CVPARTITION. Using this function, a cross-validation partition for the data will be created. A CVPARTITION object specifies a random partition on a data collection of a certain size. The size of the data set must be given. When verifying a statistical mod-el via the use of cross-validation, this division may be used to determine the test set and the training set. Once it has been partitioned successfully for training and testing, we are ready to go for the training with MATLAB's SVMTRAIN, a built-in function for the Support Vector Machine classifier.

3.4 Classifying with Testing Dataset

Once the classifier has been trained or converged the model successfully, we are ready to classify it with the testing dataset. In the classification phase, we give a testing dataset to the SVM classifier to classify new instances for which it has not been trained so far to validate whether it is classifying accurately. In fact, this is the assessment of the classifier in which it detects new instances for which it has not been trained. For classification, we have used MATLAB's built-in function SVMCLASSIFY. This function classifies data using a Support Vector Ma-chine. It classifies each row in TEST using the SVM classifier structure SVMSTRUCT created using SVMTRAIN and returns the predicted class level GROUP. TEST is required to have the same column configuration as the data that was used in SVMTRAIN in order to train the classifier. The value of GROUP shows the group that each entry of the TEST table belongs to.

3.5 Testing to Verify Accuracy

Once the SVMCLASSIFY has classified the testing dataset for which it has not been trained. It is necessary to per-form a validation test to testify to the accuracy of the classification, that is, to what extent the new dataset is classified accurately against the trained model, trained by the SVMTRAIN function. This test is based on the number of accurately classified connection vectors and inaccurately classified

connection vectors for the test dataset against the SVM classifier. Hence, we get the ratio of accurately and inaccurately classified records to know how much accuracy the support vector machine gives for detecting attacks. We calculate the accuracy and error rate for all types of attacks separately and overall accuracy. Here is how we calculate accuracy and error rate. Accuracy = total number of accurately detected attacks / total number of attacks * 100 Error_rate = total number of inaccurately detected attacks / total number of attacks * 100 If the accuracy of a reduced subset is equal to or close to that of 41 feature-set, it signifies that we have achieved our goal of finding the reduced set and stopping with the reduced set in hand. If not, we will again start filtering the features out of 41 and repeat the process to find a reduced feature set. We propose two contributions to this research. The first half of this article describes the state-of-the-art in intrusion detection using SVM and then looks at the resources that various researchers in this area have employed. Second, it proposes a novel approach to selecting the best features for detecting intrusion after multiple experiments with varying subsets of features out of 41. It has been determined that, with less time and computer pow-er, we can categorize DoS attacks using just 25 characteristics instead of 41 features. So concentrating on fewer characteristics saves time and computational resources. The SVM classifier may be used in an intrusion detection system to accurately and effectively identify different real-time attacks, such as Denial of Service (DoS) attacks after it has been trained with a sufficient amount of training with high accuracy and low error rate. The smaller dataset will result in better performance and more accuracy for the SVM-based detection model. In addition, the fewer features will result in a reduction in the time need-ed for system training and testing.

3.6 NSL-KDD Dataset

The NSL-KDD [17], a unique dataset for assessing re-search in network intrusion detection systems, is going to be employed in this study, and it will serve as the dataset that is used. It is made up of particular records taken from the comprehensive KDD 99 dataset. The NSL-KDD dataset provides solutions to the problems present-ed by the KDD 99 benchmark, and each connection record has 41 attributes. Out of the total of 41 features, there are 34 numeric features and 7 symbolic or discrete features. There are a total of 22 different training assault types included in the NSL-KDD training set, with an additional 17 available in the testing set alone. Table 1 summarizes NSL-KDD Dataset Features.

The Knowledge Discovery and Data mining tools con-test that was organized in conjunction with KDD-99, the fifth international conference on Knowledge Discovery and Data mining, included the usage of the KDD as one of the contest's tools. The objective of the competition was to construct a network intrusion detector, which is essentially a forecasting model that is able to discern between a "bad connection," which refers to an attack or intrusion, and a "good connection," which refers to a regular connection. This Standard data set is capable of being audited, which includes simulating a diverse number of incursions in a network of military settings. This data set contains millions of connection records. A single connection is a series of packets opening and closing at clearly described times, between which data is exchanged from a sender's IP address and receiver's IP address using specific, clearly defined protocols. Following are the categories of attacks included in the KDD dataset. DoS: Denial of Service, such as Syn Flood. R2L: Unauthorized Access from a distant computer, such as Passcode Guessing. U2R: Unauthorized Access to local user's privileges, such as several 'Buffer-Overflow' attacks. Probing: Surveillance and other Probing, such as Scanning. This is a publicly available labeled dataset used for intrusion detection research domain. Our experiment contains 116869 connections, of which 57.62% are normal traffic, and 53.38% are divided into different types of attacks. It has 41 features whose description is given above.

4 Simulation and result discussion

The training, testing, and analysis have been done using the KDD dataset, which has been a source of attraction for researchers worldwide for many years in the field of intrusion detection. This dataset has a rich data-base of 5 different categories of attack types, including Denial of Service (DoS) attacks. It has 41 features that specify the attributes of malicious behavior during the network activity. Our focus has been only on seven types of DoS attacks. We have undertaken experiments through SVM with these seven types of attacks to analyze the accuracy and error rate of detecting an attack. In the very first phase, we trained and tested SVM with normal and a single attack type with 41 feature-set; we are progressively increasing the number of attack types and testing their accuracy and error separately as well as overall accuracy. In addition, we have also calculated the training and classification time taken by the SVM and performed a comparative analysis among several feature sets and classes. We are searching for a reduced

TABLE 1: Description of NSL-KDD Dataset Features [26]

S. No	Name of Features	Description
1	Duration	Length (no. of seconds) of the connection
2	protocol_type	Type of the protocol
3	Service	Network service on the destination
4	Flag	Status flag of the connection
5	src_bytes	No. of data bytes from source to destination
6	dst_bytes	No. of data bytes from destination to source
7	Land	1 if connection is from/to the same host/port; 0 otherwise
8	wrong_fragment	No. of wrong fragments
9	Urgent	No. of urgent packets
10	Hot	No. of ”hot” indicators
11	num_failed_logins	Logins no. of failed logins
12	logged_in	1 if successfully logged in; 0 otherwise
13	num_compromised	No. of ”compromised” conditions
14	root_shell	1 if root shell is obtained; 0 otherwise
15	su_attempted	1 if ”su root” command attempted; 0 otherwise
16	num_root	No. of ”root” accesses
17	num_file_creations	No. of file creation operations
18	num_shells	No. of shell prompts
19	num_access_files	No. of operations on access control files
20	num_outbound_cmds	No. of outbound commands in an ftp session
21	is_host_login	1 if the login belongs to the ”hot” list; 0 otherwise
22	is_guest_login	1 if the login is a ”guest” login; 0 otherwise
23	Count	No. of connections to the same host as the current connection in the past two seconds
24	srv_count	No. of connections to the same service as the current connection in the past two seconds
25	serror_rate	% of connections that have ”SYN” errors
26	srv_serror_rate	% of connections that have ”SYN” errors
27	rerror_rate	% of connections that have ”REJ” errors
28	srv_rerror_rate	% of connections that have ”REJ” errors
29	same_srv_rate	% of connections to the same service
30	diff_srv_rate	% of connections to different services
31	srv_diff_host_rate	% of connections to different hosts
32	dst_host_count	Count of connections having the same destination host
33	dst_host_srv_count	Count of connections having the same destination host and using the same service
34	dst_host_same_srv_rate	% of connections having the same destination host and using the same service
35	dst_host_diff_srv_rate	% of different services on the current host
36	dst_host_same_src_port_rate	rate % of connections to the current host having the same src port
37	dst_host_srv_diff_host_rate	% of connections to the same service coming from different hosts
38	dst_host_serror_rate	% of connections to the current host that have an S0 error
39	dst_host_srv_serror_rate	% of connections to the current host and specified service that have an S0 error
40	dst_host_rerror_rate	% of connections to the current host that have an RST
41	dst_host_srv_rerror_rate	% of connections to the current host and specified service that have an RST error

TABLE 2: Distribution of dataset for various attacks

Class	No. of observations	%age
normal	67343	57.62%
neptune	41214	35.27%
ipsweep	3599	3.08%
smurf	2646	2.26%
back	956	0.82%
teardrop	892	0.76%
pod	201	0.17%
land	23	0.02%

feature-set out of 41 that gives equal or closest accuracy and takes less time than the 41 feature-set. We randomly selected thousands of subsets to achieve this goal and then trained, classified, and tested them with an SVM classifier. Some of them gave inferior results in terms of accuracy and lesser time, but some gave better results. Eventually, we found a reduced subset that gave the closest accuracy and less time than that of the 41 feature-set. We present some results out of thousands whose accuracy is getting higher and time is getting progressively lesser in detecting the attack. We also compared and contrasted different outcomes of different feature sets.

This experiment has a bigger feature set of 21 features. The confusion matrix-1 and Fig no. 2 show that these features are better for detecting three types of attacks, Teardrop, Smurf, and Land, because their detection ac-curacy is 100%. Whereas Ipsweep and Normal suffer from some distortion, table 15.18 shows

TABLE 3: Confusion Matrix-1: 21 features with 8 classes

	normal	Neptune	teardrop	smurf	pod	back	land	ipsweep	total
normal	30952	2353	0	98	0	24	5	239	33671
Neptune	19	20588	0	0	0	0	0	0	20607
teardrop	0	0	446	0	0	0	0	0	446
smurf	0	0	0	1323	0	0	0	0	1323
pod	3	0	0	0	97	0	0	0	100
back	14	0	0	0	0	464	0	0	478
land	0	0	0	0	0	0	9	0	9
ipsweep	10	225	0	13	0	0	0	1552	1800
training time	00:07:19								
classifying time	00:16:25								

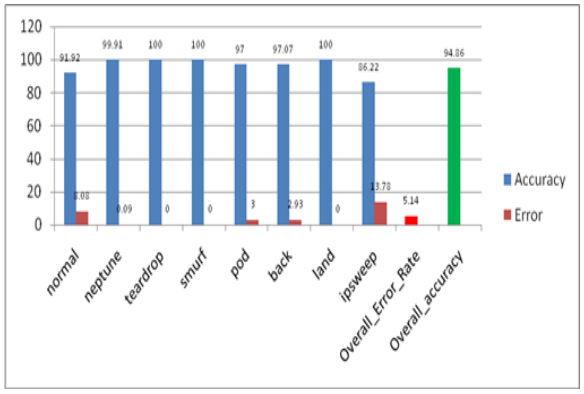

Fig. 2: Overall accuracy and error rate for 8 classes with 21 features

that 2353 connections of normal traffic are mistreated as Neptune threat, 239 as Ipsweep. Still, overall accuracy is 94.86% leaving the error rate just 5.14%.

Confusion Matrix-2 and Fig. 3 represent another feature set of 21, whose overall accuracy is better than the pre-vious one. It works better in the case of Smurf, Back, Land, and Ipsweep, and overall accuracy is 97.83% which is better than that of 16 features which gave over-all accuracy of 88.66. It has also performed better work detecting Normal traffic than the 16 feature set. However, it is not as better for the case of a Teardrop attack as we can see from the figure and confusion matrix above. It took 00:07:15 minutes for testing and 00:15:30 minutes for classification.

TABLE 4: Confusion Matrix-2: Another 21 features with 8 classes

	normal	Neptune	teardrop	smurf	pod	back	land	ipsweep	total
normal	33135	120	12	89	0	48	5	262	33671
Neptune	256	20299	0	0	0	0	0	52	20607
teardrop	45	0	401	0	0	0	0	0	446
smurf	0	0	0	1323	0	0	0	0	1323
pod	1	0	0	99	0	0	0	0	100
back	12	0	0	0	0	466	0	0	478
land	0	0	0	0	0	0	10	0	10
ipsweep	228	24	0	15	0	0	0	1532	1799
training time	00:07:15								
classifying time	00:15:30								

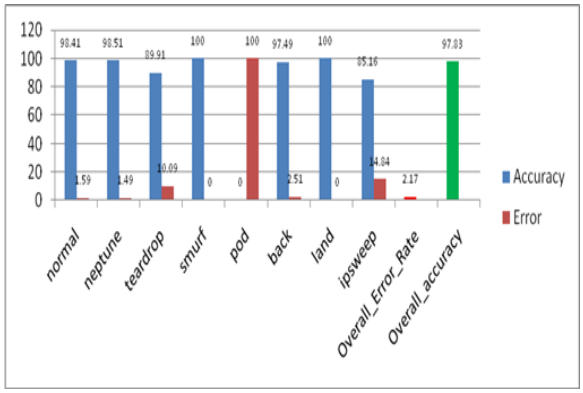

Fig. 3: Overall accuracy and error rate for 8 classes with 21 features

TABLE 5: Confusion Matrix-3: 25 features with 8 classes

	normal	Neptune	teardrop	smurf	pod	back	land	ipsweep	total
normal	33626	0	7	15		4	3	15	33672
Neptune	18	20589	0	0	0	0	0	0	20607
teardrop	0	0	446	0	0	0	0	0	446
smurf	3	0	0	1319	0	0	0	1	1323
pod	4	0	0	3	90	0	0	3	100
back	30	0	0	0	0	447	0	1	478
land	2	0	0	0	0	0	7	0	9
ipsweep	23	0	0	0	0	0	0	1776	1799
training time	00:10:21								
classifying time	00:18:35								

4.1 Reduced feature-set with high accuracy

The following experiment shows a reduced feature-set of 25 features which gives high accuracy, less training time, and less classifying time.

Confusion Matrix-3 and Fig. 4 represent the optimal and reduced subset of 25 features, which gives the closest accuracy separately for all attacks and overall aver-age accuracy, leaving the least overall error rate. It provides overall accuracy of 99.72% compared to 41 fea-tures with 99.78%. All attack types are more than 98% accurate except Pod, Back, and Land. It is for the sole reason that these attack types have

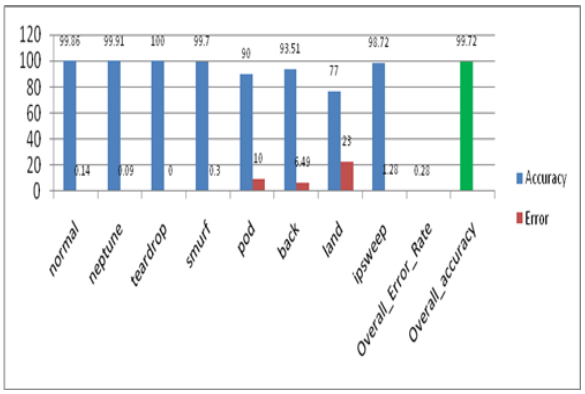

Fig. 4: Overall accuracy and error rate for 8 classes with 25 features

TABLE 6: Comparative accuracies of different feature sets with varying numbers of classes

		Classes						
Feature Sets	9	64.21	62.11	59.34	55.78	51.6	50.03	46.9
	10	95.78	95.33	93.21	94	94.12	94.99	94.95
	11	95.6	95.12	95.77	95.88	95.7	95.57	95.38
	15	55.89	55.43	51.56	50.43	47.99	47.89	47.1
	16	62.65	59.11	57.87	55.65	52.12	49.88	49.37
	16	95.67	96.89	96.67	96.47	96.55	96.5	96.47
	17	96.34	96.2	95.55	95.13	91.87	90.89	90.76
	18	93.7	93.75	93.22	92.1	93.99	93.89	93.05
	19	94.12	93.99	93.23	94.6	94.12	93.99	93.97
	21	97.99	97.96	97.93	97.83	97.45	97.88	94.83
	25	99.96	99.95	99.92	99.86	99.83	99.81	99.72
	28	95.98	95.82	94.46	93.89	94.11	94.87	94.88
	30	95.19	95.1	94.99	94	94.92	94.89	94.86
	35	92.1	92.12	91.4	89.99	89.43	88.89	88.66
	41	99.97	99.96	99.91	99.89	99.86	99.85	99.78

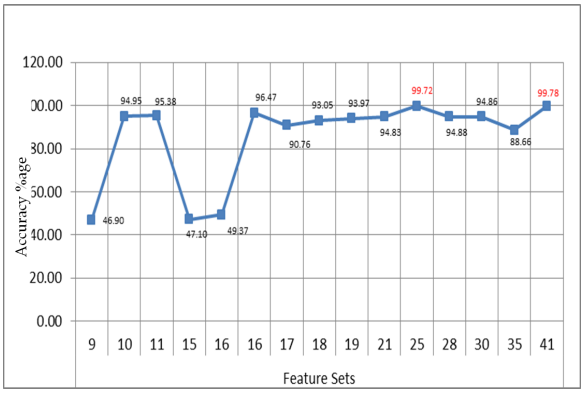

Fig. 5: An overall analysis of comparative accuracies of different feature sets with varying numbers of classes

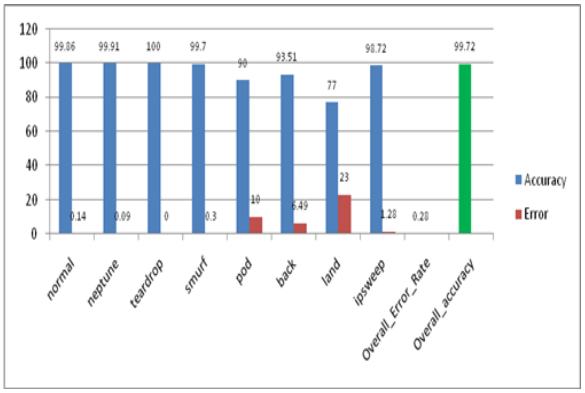

Fig. 6: Overall accuracy and error rate for 8 classes with 41 features

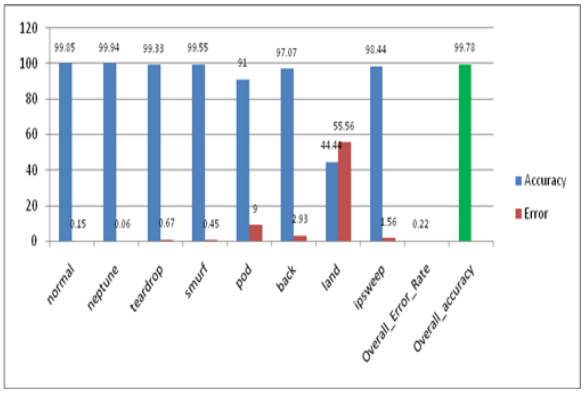

Fig. 7: Overall accuracy and error rate for 8 classes with 25 features

fewer observations in their training data set. Generally speaking, the lesser the training data, the more the error it produces for su-pervised learning, be it an SVM or any other machine learning technique. If we look at the time taken for train-ing and classification for these 25 features and the time taken by all 41 features, we found that the time for train-ing and classification for 25 features is 61% and 77%, respectively, lesser than that of 41 features. Comparative analysis for both feature sets is given ahead, showing that these 25 features are optimal over-all accuracy, training time, and classification time than 41 feature sets.

4.2 Overall analyses of different features-sets

Above Fig. 5 and Table 3 represents the overall analysis of several features as different features give different accuracy. For example, 9 features give overall accuracy of 46.90%, whereas 17 features give 90.76% accuracy. Another thing to notice here is that increasing the number of features does not mean that it will provide better accuracy; instead, it is the property of a specific feature that contributes to detecting a particular threat. This can be seen in the above figure that one set of 16 features gives 49.37% accuracy while the other set of 16 features gives 96.47% accuracy. Therefore just increasing the number of features does not lead you to higher accuracy; instead, it is based on careful selection of those features, which contributes to detecting a specific threat. It has also been observed that a subset of 10 features gives higher accuracy than many other larger subsets like 15, 16, 17, 21, and 28. Hence selecting an optimal subset has been a challenge for many researchers in the intrusion detection domain. Nonetheless, we have found an optimal subset of 25 features that give the closest accuracy and take less training and classification time than 41 features. We can see from the figure above that a feature set of 25 gives 99.72% accuracy, which is closest to 99.78% provided by all 41 features—further, the fewer the number of features, the lesser the training and classification time. We have also compared the time for training and classification of our proposed feature set and 41 features set in upcoming sections.

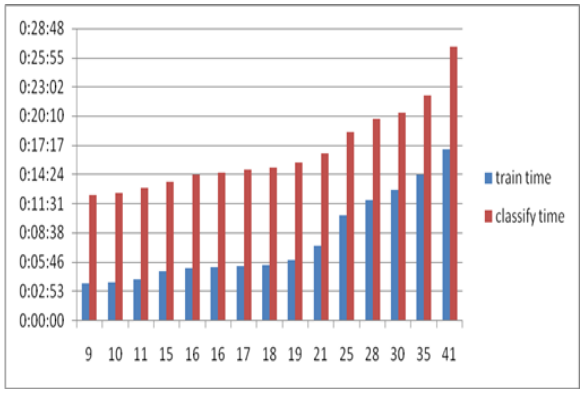

Fig. 8: Time taken by different subsets of feature sets

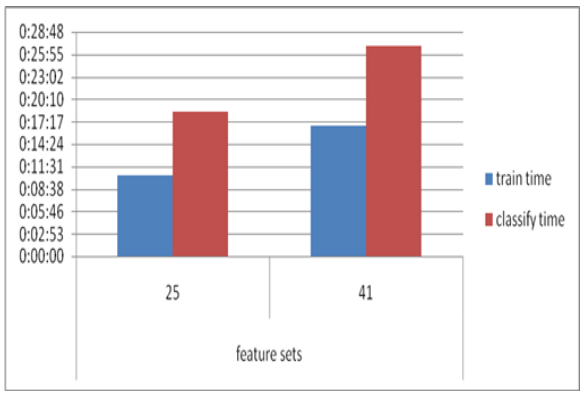

Fig. 9: Time taken by 41 features and 25 features

4.3 Comparison of 41 and 25 feature-sets

If we compare Fig. 6 and 7 in terms of the accuracy of all 7 types of DoS attacks, we can conclude that the individual accuracy of all attacks is very close to that of 41 features. Accuracy of Normal is 99.8% and 99.85% respectively, however for Neptune is 99.91% and 99.94%, for Teardrop is 100% and 99.33% , for Smurf is 99.7% and 99.55%, for Pod is 90% and 91%, for Back is 93.51% and 97.7%, for Land 77% and 44.44% for Ipsweep 98.72% and 98.44% respectively. For both feature sets, overall accuracy is 99.78% and 99.72%, respectively, and the overall error rate is 0.22% and 0.28, respectively. Hence it is concluded that the accuracy of these 25 features is highly close to that of 41 feature sets.

4.4 Comparing training and classifying time of different features-sets

In Fig. 8, we have pictured the training and classifying time of different subsets of features; we can observe that the bigger the feature-set, the higher the training and classification time is. The more the features, the more the training and classification time it takes. For every supervised learning, if we increase the input space, the time taken by the learning algorithm also increases. Here another thing to be noticed is that the classification time is always more significant than the training time. Hence, it is clear from Fig. 8 that the time taken to train and classify SVM with fewer features is less than the time taken with more features. In our case, we found the optimal set of 25 features, which takes less time for training and classifying than the time taken by 41 feature-set, as depicted in Fig.9.

5 Conclusion

Since attacks on networks have become more frequent and damaging over the past several years, the use of an intrusion detection system, often known as an IDS, has become an increasingly important component in safe-guarding networks. The research community is paying increasing attention to the serious open issue of enhanc-ing IDS performance as a result of the massive amounts of security audit data and the complex and dynamic aspects of intrusion behaviors. The Support Vector Ma-chine (SVM) is widely regarded as one of the most effec-tive machine learning algorithms for classifying anom-alous behavior, and it is only one of the many methods that may be used to identify and prevent intrusions. Support vector machines provide the foundation of a significant number of intrusion detection systems. They are, nevertheless, computationally intensive. Dimension reduction methods are used to extract crucial character-istics from a dataset in order to overcome this issue. The following are the two contributions to this study. First, this article discusses current developments in intrusion detection using SVM in the first section, followed by an examination of the tools used by several researchers in this field. Second, it proposes a novel approach to select-ing the best features for detecting intrusion after various experiments with varying subsets of features out of 41. It has been concluded that we can only use 25 features to classify DoS attacks instead of using 41 features with less time and computing resources. Hence focusing on a lesser number of features save time and computing re-sources. Once the SVM classifier has been trained with a sufficient amount of training with high accuracy and less error rate, we can implement this classifier in an Intrusion Detection System to detect various real-time attacks accurately and efficiently, including Denial of Service (DoS) attacks. The SVM-based detection model will have improved performance as well as increased accuracy as a result of the decreased dataset. In addition to this, the amount of time spent training and testing the system will be cut down as a result of the decreased number of features.

References

[1] Jackson, Timothy R., John G. Levine, Julian B. Grizzard, and Henry L. Owen. "An investigation of a compromised host on a honeynet being used to increase the security of a large enterprise network." In Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004., pp. 9-14. IEEE, 2004.

[2] Denning, Dorothy E. "An intrusion-detection model." IEEE Transactions on software engineering 2 (1987): 222-232.

[3] Frank, Jeremy. "Artificial intelligence and intrusion detection: Current and future directions." In Proceedings of the 17th national computer security conference, vol. 10, pp. 1-12. 1994.

[4] Chen, Rung-Ching, Kai-Fan Cheng, Ying-Hao Chen, and Chia-Fen Hsieh. "Using rough set and support vector machine for network intrusion detection system." In 2009 First Asian Conference on Intelligent Information and Database Systems, pp. 465-470. IEEE, 2009.

[5] Chen, Rung-Ching, Kai-Fan Cheng, Ying-Hao Chen, and Chia-Fen Hsieh. "Using rough set and support vector machine for network intrusion detection system." In 2009 First Asian Conference on Intelligent Information and Database Systems, pp. 465-470. IEEE, 2009.

[6] Bennett, Kristin P., and J. A. Blue. "A support vector machine approach to decision trees." In 1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98CH36227), vol. 3, pp. 2396-2401. IEEE, 1998.

[7] Lee, Wenke, Salvatore J. Stolfo, and Kui W. Mok. "A data mining framework for building intrusion detection models." In Proceedings of the 1999 IEEE Symposium on Security and Privacy (Cat. No. 99CB36344), pp. 120-132. IEEE, 1999.

[8] Frank, Jeremy. "Artificial intelligence and intrusion detection: Current and future directions." In Proceedings of the 17th national computer security conference, vol. 10, pp. 1-12. 1994.

[9] Villena-Román, Julio, Sonia Collada-Pérez, Sara Lana-Serrano, and José Carlos González-Cristóbal. "Hybrid approach combining machine learning and a rule-based expert system for text categorization." In Twenty-Fourth International FLAIRS Conference. 2011.

[10] Amor, Nahla Ben, Salem Benferhat, and Zied Elouedi. "Naive bayes vs decision trees in intrusion detection systems." In Proceedings of the 2004 ACM symposium on Applied computing, pp. 420-424. 2004.

[11] Mukkamala, Srinivas, Guadalupe Janoski, and Andrew Sung. "Intrusion detection using neural networks and support vector machines." In Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290), vol. 2, pp. 1702-1707. IEEE, 2002.

[12] Shah, Hiren, Jeffrey Undercoffer, and Anupam Joshi. "Fuzzy clustering for intrusion detection." In The 12th IEEE International Conference on Fuzzy Systems, 2003. FUZZ'03., vol. 2, pp. 1274-1278. IEEE, 2003.

[13] Ambwani, Tarun. "Multi class support vector machine implementation to intrusion detection." In Proceedings of the International Joint Conference on Neural Networks, 2003., vol. 3, pp. 2300-2305. IEEE, 2003.

[14] Anand, Ashish, and Ponnuthurai N. Suganthan. "Multiclass cancer classification by support vector machines with class-wise optimized genes and probability estimates." Journal of theoretical biology 259, no. 3 (2009): 533-540.

[15] Noble, William S. "What is a support vector machine?." Nature biotechnology 24, no. 12 (2006): 1565-1567.

[16] Suthaharan, Shan. "Machine learning models and algorithms for big data classification." Integr. Ser. Inf. Syst 36 (2016): 1-12.

[17] Vieira, S., R. Garcia-Dias, and W. Pinaya. "Machine Learning Methods and Applications to Brain Disorders, ch." A step-by-step tutorial on how to build a machine learning model (2019): 343-370.

[18] Sun, Aixin, Ee-Peng Lim, and Wee-Keong Ng. "Web classification using support vector machine." In Proceedings of the 4th international workshop on Web information and data management, pp. 96-99. 2002.

[19] Tong, Simon, and Daphne Koller. "Support vector machine active learning with applications to text classification." Journal of machine learning research 2, no. Nov (2001): 45-66.

[20] Peddabachigari, Sandhya, Ajith Abraham, Crina Grosan, and Johnson Thomas. "Modeling intrusion detection system using hybrid intelligent systems." Journal of network and computer applications 30, no. 1 (2007): 114-132.

[21] Heba, F. Eid, Ashraf Darwish, Aboul Ella Hassanien, and Ajith Abraham. "Principle components analysis and support vector machine based intrusion detection system." In 2010 10th international conference on intelligent systems design and applications, pp. 363-367. IEEE, 2010.

[22] Shon, Taeshik, Yongdae Kim, Cheolwon Lee, and Jongsub Moon. "A machine learning framework for network anomaly detection using SVM and GA." In Proceedings from the sixth annual IEEE SMC information assurance workshop, pp. 176-183. IEEE, 2005.

[23] Zhou, Qifeng, Hao Zhou, Qingqing Zhou, Fan Yang, and Linkai Luo. "Structure damage detection based on random forest recursive feature elimination." Mechanical Systems and Signal Processing 46, no. 1 (2014): 82-90.

[24] Joseph, John Felix Charles, Bu-Sung Lee, Amitabha Das, and Boon-Chong Seet. "Cross-layer detection of sinking behavior in wireless ad hoc networks using SVM and FDA." IEEE Transactions on Dependable and Secure Computing 8, no. 2 (2010): 233-245.

[25] Memon, Saifullah, Jingyu Wang, Ali Raza Bhangwar, Suliman Mohamed Fati, Amjad Rehman, Tong Xu, and Lei Zhang. "Temperature and reliability-aware routing protocol for Wireless Body Area networks." IEEE Access 9 (2021): 140413-140423.

[26] "University of New Brunswick (UNB), Canadian Institute for Cybersecurity. Available at https://www.unb.ca/cic/datasets/nsl.html"