Intrusion Detection System Using Machine Learning Algorithms

Table Of Contents

Abstract

The goal of this project is to give the user an overview of Intrusion Detection Systems and a thorough understanding of some advanced intrusion detection techniques. Intrusion detection is a crucial part of infrastructure security. Given the rising complexities of today's network infrastructures, an increasing number of hosts are becoming vulnerable to attacks, making it critical to investigate systematic, efficient, and automated Intrusion Detection methods. We strive to build a predictive model capable of distinguishing between good and bad connections, through which we examine and contrast the strengths and drawbacks of different machine learning based intrusion detection techniques.

Key-words

Intrusion Detection, Signature based detection, Anomaly based detection, Gaussian Naive Bayes, Decision Tree, Random Forest, Support Vector Machine, Logistic Regression

Problem Statement

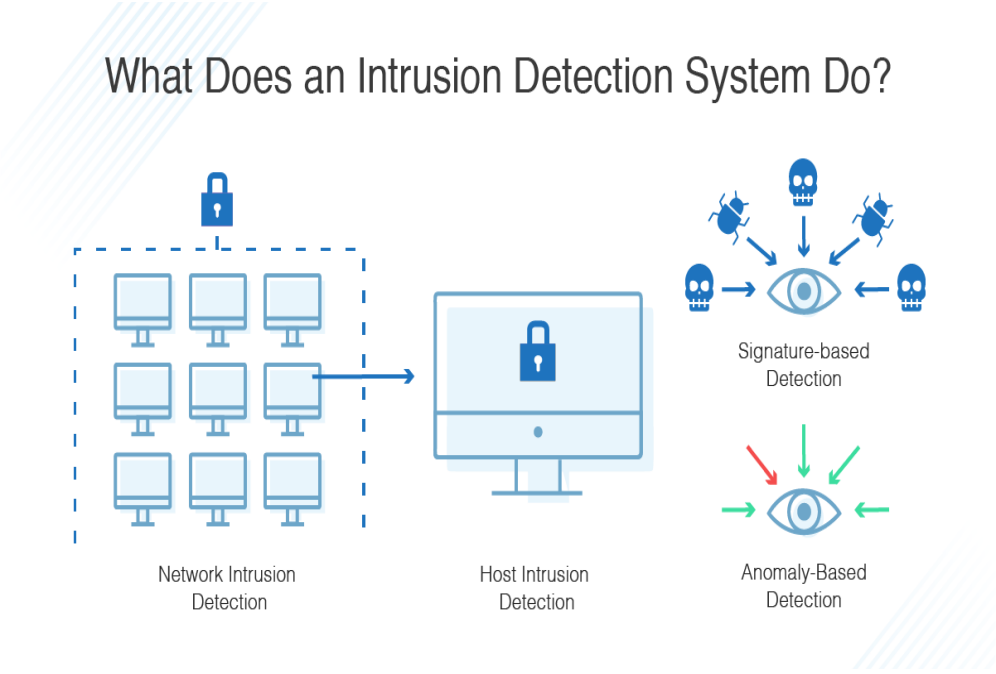
The task is to build a network intrusion detector model that is able to predict and distinguish between bad connections, called intrusions or attacks, and good normal connections, unlike the signature based IDS that scans for specific patterns. Such a system will be able to classify incoming connections at an earlier stage and protect the host network.

Introduction

An intrusion detection system (IDS) is a device or software application that monitors a network for malicious activity or policy violations. Any malicious activity or violation is typically reported or collected centrally using a security information and event management system. Some IDS's are capable of responding to detected intrusion upon discovery. These are classified as intrusion prevention systems (IPS).

Signature based IDS detects possible threats by looking for specific patterns such as byte sequences, malicious instruction sequences used by malware. However, it does not detect new attacks, for which no pattern is available.

Focusing on anomaly based IDS, it is newer technology to detect and adapt to unknown attacks. This method uses Machine Learning to create a model of trustworthy activity. While it detects previously unknown attacks, it suffers from false positives.



Unlike signature-based IDS, which scans for specific patterns, the goal is to create a network intrusion detector model that can forecast and distinguish between bad connections, also known as intrusions or attacks, and good regular connections. Such a system will be able to classify incoming connections and safeguard the host network at an earlier stage.

Literature Review

Authors	Journal/Year	Title/Study	Concept/Theoretical Model/Relevant Finding
Sui Xin	IEEE International Conference on Computational and Information Sciences, 2013	Research of Intrusion Detection System	Firstly, this article introduces the development of intrusion detection system. Secondly, this paper introduces the new technologies of intrusion detection systems. Finally, the paper prospects the development of intrusion detection systems.
Manzoor Chachoo , Uzair Bashir	IEEE International Conference on Computing for Sustainable Global Development (INDIACom), 2014	Intrusion detection and prevention system: Challenges & opportunities	To understand the security risks and IDPS (intrusion detection and prevention system), a survey is made about the common security breaches and then after discuss what are different opportunities and challenges in this particular field.

Amit Kumar, Harish Chandra Maurya, Rahul Misra	International Journal of Engineering and Advanced Technology (IJEAT),2013	A Research Paper on Hybrid Intrusion Detection System	Number of approaches have been developed for intrusion detection including k-nearest-neighbor (KNN) classification, Naïve Bayes classification, support vector machines (SVM), decision tree (DT), neural network (NN), and maximum entropy.
Hamza Nachan,Dristi Poddar,Sambhaji Sarode	International Journal of Engineering Research & Technology (IJERT),2021	Intrusion Detection System: A Survey	Summarized the IDSs proposed to date based on machine learning, to abstract the key ideas of applying machine learning to security domain issues

W. Lee, S. J. Stolfo, and K. W. Mok	IEEE Symposium on Security and Privacy, 1999	A data mining framework for building intrusion detection models	A data mining framework for constructing intrusion detection models. The key idea is to apply data mining programs to audit data to compute misuse and anomaly detection models, according to the observed behavior in the data.
Lee, W. and Stolfo, S. J.	ACM Transactions on Information and System Security, 2000	A Framework for Constructing Features and Models for Intrusion Detection Systems	Proposed the use of meta-learning as a means to construct a combined model that incorporates evidence from multiple base models.
Leonid Portnoy, Eleazar Eskin, and Salvatore J. Stolfo	DMSA, 2001	Intrusion detection with unlabeled data using clustering	Presented a new type of clustering-based intrusion detection algorithm, unsupervised anomaly detection, which trains on unlabeled data in order to detect new intrusions
W. Lee, S. Stolfo, P.	DARPA Informati	Real Time Data Mining-based	Addresses issues of accuracy, efficiency, and usability of real-time IDSs. Feature extraction and construction

Chan, E. Eskin, W. Fan, M. Miller, S. Hershkop, J. Zhang	on Survivability Conference and Exposition II, 2001	Intrusion Detection	algorithms for labeled audit data have been implemented
Chris Sinclair, Lyn Pierce, Sara Matzner	Proceedings 15th Annual Computer Security Applications Conference	An Application of Machine Learning to Network Intrusion Detection	An application which enhances domain knowledge with machine learning techniques to create rules for an intrusion detection system was created. Genetic algorithms and decision trees were employed.

Chih-Fong Tsai, Yu-Feng Hsu, Chia-Ying Lin, Wei-Yang Lin	Expert Systems with Applications 36 (2009)	Intrusion detection by machine learning: A review	Reviews studies related to intrusion detection systems focusing on developing single, hybrid, and ensemble classifiers. Related studies are compared by their classifier design, datasets used, and other experimental setups

Overview

Classification in Machine Learning

- Supervised learning is an approach to creating artificial intelligence (AI), where a computer algorithm is trained on input data that has been labeled for a particular output.
- Supervised Machine Learning algorithms can be broadly classified into Regression and Classification. In Regression, we have predicted the output for continuous values, but to predict the categorical values, we need Classification algorithms, that is, the output variable of Classification is a category.
- In Classification, a program learns from the given dataset or observations and then classifies new observations into a number of classes or groups.
- In this project, the aim is to classify network connections into anomalous and normal.

Technologies Used

- Python
- VS-Code
- CLI
- DataSet Used : KDD CUPS 1999
- Python Libraries Used:
    - TKinter
    - CryptoDome
    - PIL
    - OS

Implementation

List of Features

Table 1: Features

Feature name	Description	Type
duration	length (number of seconds) of the connection	continuous
protocol_type	type of the protocol, e.g. tcp, udp, etc.	discrete
service	network service on the destination, e.g., http, telnet, etc.	discrete
src_bytes	number of data bytes from source to destination	continuous
dst_bytes	number of data bytes from destination to source	continuous
flag	normal or error status of the connection	discrete
land	1 if connection is from/to the same host/port; 0 otherwise	discrete

wrong_fragment	number of “wrong” fragments	continuous
urgent	number of urgent packets	continuous

Table 2: Basic features of individual TCP connections.

Feature name	Description	Type
hot	number of “hot” indicators	continuous
num_failed_logins	number of failed login attempts	continuous
logged_in	1 if successfully logged in; 0 otherwise	discrete
num_compromised	number of “compromised” conditions	continuous
root_shell	1 if root shell is obtained; 0 otherwise	discrete
su_attempted	1 if “su root” command attempted; 0 otherwise	discrete
num_root	number of “root” accesses	continuous
num_file_creations	number of file creation operations	continuous

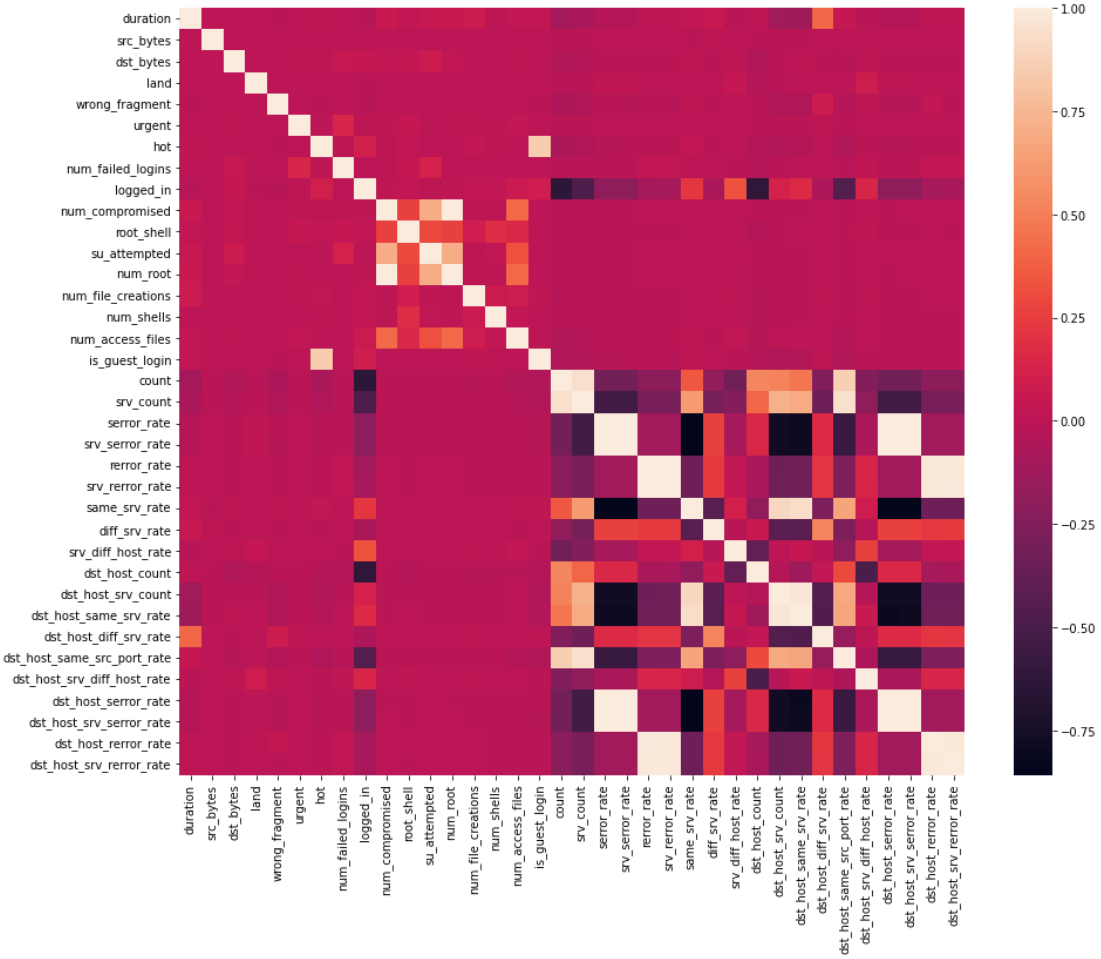
num_shells	number of shell prompts	continuous
num_access_files	number of operations on access control files	continuous
num_outbound_cmds	number of outbound commands in an ftp session	continuous
is_hot_login	1 if the login belongs to the “hot” list; 0 otherwise	discrete
is_guest_login	1 if the login is a “guest”login; 0 otherwise	discrete

Table 3: Content features within a connection suggested by domain knowledge.

Feature name	Description	Type
count	number of connections to the same host as the current connection in the past two seconds	continuous
serror_rate	% of connections that have “SYN” errors	continuous
rerror_rate	% of connections that have “REJ” errors	continuous
same_srv_rate	% of connections to the same service	continuous
diff_srv_rate	% of connections to different services	continuous
srv_count	number of connections to the same service as the current connection in the past two seconds	continuous
srv_serror_rate	% of connections that have “SYN” errors	continuous

srv_rerror_rate	% of connections that have “REJ” errors	continuous
srv_diff_host_rate	% of connections to different hosts	continuous

Correlation Heatmap Used For Feature Engineering



Machine Learning Algorithms Used

- Gaussian Naive Bayes
- Decision Tree
- Random Forest
- Support Vector Machine
- Logistic Regression

Gaussian Naive Bayes

The naïve Bayes model is a heavily simplified Bayesian probability model.The naive Bayesian intelligent intrusion detection model is based on the number of network connection records. The Naive Bayes-based intrusion detection model consists of two parts: training and detection. The method of establishing an intrusion detection model is to represent the records in the network connection by n-dimensional vectors (A1,A2, ..., An), where n represents the n characteristic attributes of the connection record. Since the structure of the naive Bayes network is static, it is here. In this case, only the conditional probability of each node needs to be calculated.

Given a series of n attributes,the naïve Bayes classifier makes 2n! independent assumptions. It states that the error is a result of three factors: training data noise, bias, and variance. Training data noise can only be minimized by choosing good training data. The training data must be divided into various groups by the machine learning algorithm. Bias is the error due to groupings in the training data being very large. Variance is the error due to those groupings being too small.

Code

```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

clfg = GaussianNB()
start_time = time.time()
clfg.fit(X_train, y_train.values.ravel())
end_time = time.time()
print("Training time: ", end_time-start_time)

start_time = time.time()
y_test_pred = clfg.predict(X_train)
end_time = time.time()
print("Testing time: ", end_time-start_time)


print("Train score is:", clfg.score(X_train, y_train))
print("Test score is:", clfg.score(X_test, y_test))
```

Decision Tree

A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.
A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

Code :

```
from sklearn.tree import DecisionTreeClassifier

clfd = DecisionTreeClassifier(criterion ="entropy",
max_depth = 4)
start_time = time.time()
clfd.fit(X_train, y_train.values.ravel())
end_time = time.time()
print("Training time: ", end_time-start_time)


start_time = time.time()
y_test_pred = clf.predict(X_train)
end_time = time.time()
print("Testing time: ", end_time-start_time)


print("Train score is:", clfd.score(X_train, y_train))
print("Test score is:", clfd.score(X_test, y_test))
```

Random Forest is built upon the Decision Tree algorithm. Random forest consists of a large number of individual decision trees that operate as an ensemble. Ensemble learning is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.
Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.

Assumptions:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

Code :

```
from sklearn.ensemble import RandomForestClassifier

clfr = RandomForestClassifier(n_estimators = 30)
start_time = time.time()
clfr.fit(X_train, y_train.values.ravel())
end_time = time.time()
print("Training time: ", end_time-start_time)


start_time = time.time()
y_test_pred = clfr.predict(X_train)
end_time = time.time()
print("Testing time: ", end_time-start_time)
```

```
print("Train score is:", clfr.score(X_train, y_train))
print("Test score is:", clfr.score(X_test, y_test))
```

Support Vector Machine

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called support vectors, and hence the algorithm is termed as Support Vector Machine.

SVM can be of two types:

- Linear SVM: Linear SVM is used for linearly separable data.
- Non-linear SVM: Non-Linear SVM is used for non-linearly separated data,

Code :

```
from sklearn.svm import SVC

clfs = SVC(gamma = 'scale')
start_time = time.time()
clfs.fit(X_train, y_train.values.ravel())
end_time = time.time()
print("Training time: ", end_time-start_time)
```

```
start_time = time.time()
y_test_pred = clfs.predict(X_train)
end_time = time.time()
print("Testing time: ", end_time-start_time)

print("Train score is:", clfs.score(X_train, y_train))
print("Test score is:", clfs.score(X_test, y_test))
```

Logistic Regression

The method of modeling the probability of a discrete result given an input variable is known as logistic regression. The most frequent logistic regression models have a binary outcome, which might be true or false, yes or no, and so forth. Multinomial logistic regression can be used to model situations with more than two discrete outcomes. Logistic regression is a handy analysis tool for determining if a fresh sample fits best into a category in classification tasks. Because components of cyber security, such as threat detection, are classification problems, logistic regression is a valuable analytic tool.

Code :

```
from sklearn.linear_model import LogisticRegression

clfl = LogisticRegression(max_iter = 1200000)
start_time = time.time()
clfl.fit(X_train, y_train.values.ravel())
end_time = time.time()
print("Training time: ", end_time-start_time)

start_time = time.time()
y_test_pred = clf.predict(X_train)
```

```
end_time = time.time()
print("Testing time: ", end_time-start_time)

print("Train score is:", clfl.score(X_train, y_train))
print("Test score is:", clfl.score(X_test, y_test))
```

Results

Gaussian Naive Bayes

Training time: 0.9059512615203857

Testing time: 0.25504112243652344

Train score: 0.8795114110829804

Test score: 0.8790384414851528

Decision Tree

Training time: 1.5914483070373535

Testing time: 0.026651859283447266

Train score: 0.9905829108684749

Test score: 0.9905230421954646

Random Forest

Training time: 10.687421083450317

Testing time: 0.6355631351470947

Train score: 0.9999728091747887

Test score: 0.9996810344298798

Support Vector Machine

Training time: 214.3988482952118

Testing time: 98.3910117149353

Train score: 0.9987552644458811

Test score: 0.9987916112055059

Logistic Regression

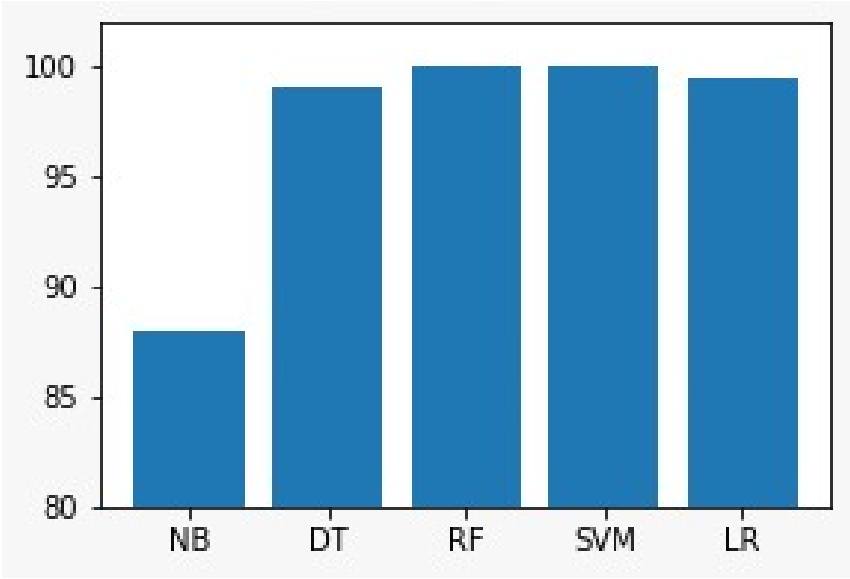
Training time: 86.94573068618774

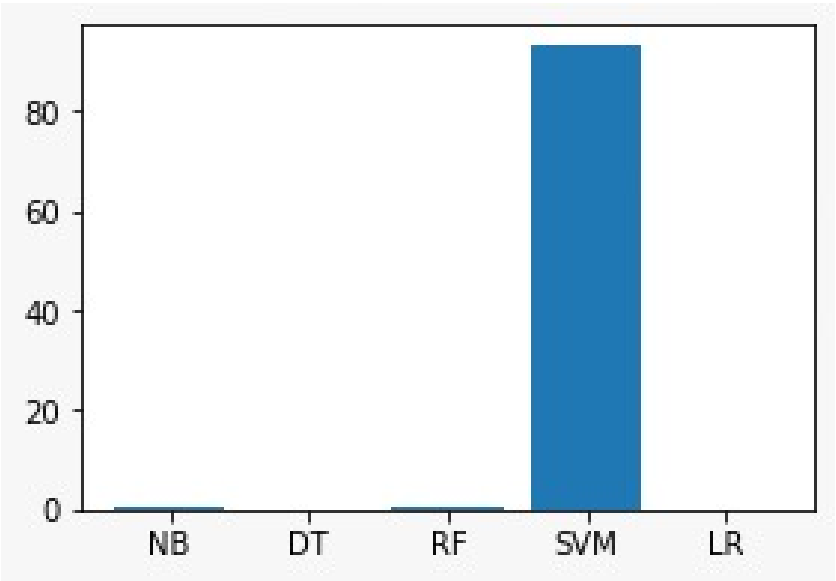
Testing time: 0.049623727798461914

Train score: 0.9935285835997028

Test score: 0.9935286792985211

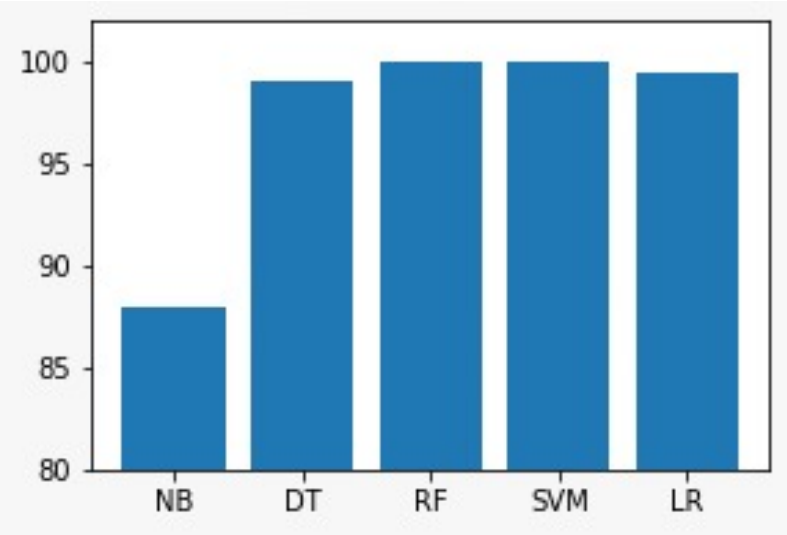
Visual Analysis

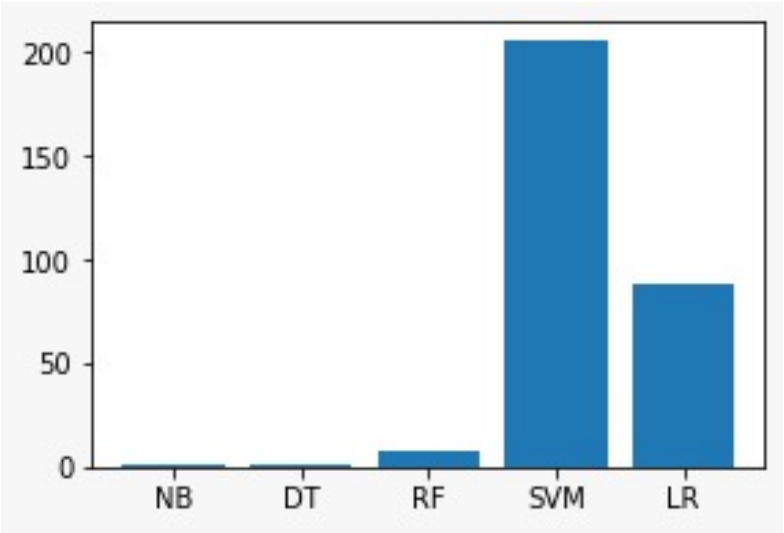




Test Accuracy

Test Time





Training Accuracy

Training Time

Conclusion

We reviewed and implemented several influential algorithms for intrusion detection based on various machine learning techniques. Characteristics of ML techniques makes it possible to design an IDS that has high detection rates and low false positive rates while the system quickly adapts itself to changing malicious behaviors.

Based on the train time, test time, train score and test score, we arrive at the conclusion that SVM is the best predictive model for an Intrusion Detection System.

References

- Qing Si-han. Cryptography and Network Security.Bei Jing: Tsinghua University Press, 2009
- Ge Yan-qiang. Practical Computer Network Security Technology. Bei Jing: Publishing House of China Water Resources and Hydropower, 2010
- Qu Xiao-hong. Research on distributed intrusion detection systems based on Protocol analysis Anti-counterfeiting,Security and Identification in Communication,2009. ASID 2009,3rd International Conference on 20-22 Aug, 2009.
- C. Kaufman, R. Perlman, and M. Speciner, Network Security: Private Communication in a Public World, 2nd Ed. 2002
- R. Krishnapuram and J. M. Keller, "A possibilistic approach to clustering", *IEEE Trans. Fuzzy Systems.*, vol. 1, pp. 98-110, May 1993.
- Ian H. Witten and Eibe Frank, Data Mining Practical Machine Learning Tools and Techniques with Java Implementations, USA:Morgan Kaufmann, 2000.
- Marrack P,kappler J W,How the Immune System Recognizes the Body. Scientific American,2003