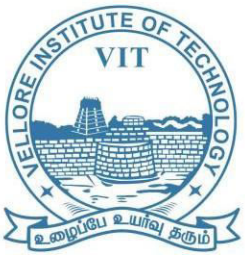


VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SIGNATURE BASED INTRUSION DETECTION SYSTEM

PROJECT REPORT
CYBER SECURITY
(CSE4003)

By

STUDENT NAME	REGISTRATION NUMBER
RONAK JAIN	17BCE0481
SHOBHIT BISHNOI	17BCE2067
BHARAT JOSHI	18BCE0003

Slot : B2/TB2

Fall 2019-2020

UNDER THE GUIDANCE OF

PROF. PRAMOD KUMAR MAURYA

School Of Computer Science and Engineering

VELLORE INSTITUTE OF TECHNOLOGY, VELLORE

ABSTRACT

Securing information systems these days in not an option rather than it is a must. The increasing number of attacks on networks and individual systems raised the need for a fast, light, and reliable Intrusion Detection System (IDS). There are two types of detection that an IDS uses: anomaly based detection and signature based detection. This paper focus on the signature based detection. Signature based IDS suffers from the huge number of signatures stored in its database.The purpose of this report is to introduce the user to Intrusion Detect Systems .Intrusion Detection is an important component of infrastructure protection mechanisms. Given the increasing complexities of today's network environments, more and more hosts are becoming vulnerable to attacks and hence it is important to look at systematic, efficient and automated approaches for Intrusion Detection. Here we discuss some data mining based approaches for intrusion detection and compare their merits and demerits. We also look at some signature based detection techniques. This project represents a proposed module that uses parallel processing with small databases with most frequent signatures and an updating agent. This module could be used for both Host based IDS and Network based IDS.

Keywords: Intrusion Detection, Data Mining, Signature based detection, Anomaly based detection.

Chapter 1: What is an IDS?

1.1. Introduction

An Intrusion Detection System is used to detect all types of malicious network traffic and computer usage that can't be detected by a conventional firewall. This includes network attacks against vulnerable services, data driven attacks on applications, host based attacks such as privilege escalation, unauthorized logins and access to sensitive files, and malware (viruses, trojan horses, and worms).

An IDS is composed of the following three components:

Sensors: - which sense the network traffic or system activity and generate events.

Console: - to monitor events and alerts and control the sensors,

Detection Engine: - that records events logged by the sensors in a database and uses a system of rules to generate alerts from the received security events.

There are several ways to categorize an IDS depending on the type and location of the sensors and the methodology used by the engine to generate alerts. In many simple IDS implementations all three components are combined in a single device or appliance.

1.2. Types of Intrusion-Detection systems

Network Intrusion Detection System: - identifies intrusions by examining network traffic and monitors multiple hosts. Network Intrusion Detection Systems gain access to network traffic by connecting to a hub, network switch configured for port mirroring, or network tap. An example of a NIDS is Snort.

Host-based Intrusion Detection System: - consists of an agent on a host which identifies intrusions by analyzing system calls, application logs, file-system modifications (binaries, password files, capability/acl databases) and other host activities and state.

Hybrid Intrusion Detection System: - combines one or more approaches. Host agent data is

combined with network information to form a comprehensive view of the network. An example of a Hybrid IDS is Prelude.

1.3. Passive system v/s reactive system

In a **passive system**, the IDS sensor detects a potential security breach, logs the information and signals an alert on the console. In a reactive system, which is known as an Intrusion Prevention System (IPS) the IDS responds to the suspicious activity by resetting the connection it believes to be suspicious or by reprogramming the firewall to block network traffic from the suspected malicious source, either autonomously or at the command of an operator.

Though they both relate to network security, an IDS differs from a firewall in that a firewall looks outwardly for intrusions in order to stop them from happening. The firewall limits the access between networks in order to prevent intrusion and does not signal an attack from inside the network. An IDS evaluates a suspected intrusion once it has taken place and signals an alarm. An IDS also watches for attacks that originate from within a system.

1.4. Signature Based Detection v/s Anomaly Based Detection

Signature based detection:-

This detection technique uses specifically known patterns to detect malicious code. These specific patterns are called signatures. Identifying the worms in the network is an example of signature based detection.

- monitor packets on the network
- compare them against a database of signature or attributes from known malicious threats
- issue is that there will be a lag between a new threat being discovered in the wild and the signature for detecting that threat being applied to the IDS.
- During that lag time the IDS would be unable to detect the new threat.

Anomaly Detection:-

These techniques are designed to detect abnormal behavior in the system. The normal usage pattern is baselined and alerts are generated when usage deviates from the normal behavior.
Example if a user logs on and off 20 times a day while the normal behavior is 1-2 times.

Chapter 2:-problem identification

2.1 problem definition

IDS technology itself is undergoing a lot of enhancements. It is therefore very important for organizations to clearly define their expectations from the IDS implementation. IDS technology has not reached a level where it does not require human intervention. Of course today's IDS technology offers some automation like notifying the administrator in case of detection of a malicious activity, shunning the malicious connection for a configurable period of time, dynamically modifying a router's access control list in order to stop a malicious connection etc. But it is still very important to monitor the IDS logs regularly to stay on top of the occurrence of events. Monitoring the logs on a daily basis is required to analyze the kind of malicious activities detected by the IDS over a period of time. Today's IDS has not yet reached the level where it can give historical analysis of the intrusions detected over a period of time. This is still a manual activity. It is therefore important for an organization to have a well-defined Incident handling and response plan if an intrusion is detected and reported by the IDS. Also, the organization should have skilled security personnel to handle this kind of scenario.

The question is, where does the Intrusion detection system fit in the design. To put it in simpler terms, an Intrusion detection system can be compared with a burglar alarm. For example, the lock system in a car protects the car from theft. But if somebody breaks the lock system and tries to steal the car, it is the burglar alarm that detects that the lock has been broken and alerts the owner by raising an alarm.

The Intrusion detection system in a similar way complements the firewall security. The firewall protects an organization from malicious attacks from the Internet and the Intrusion detection system detects if someone tries to break in through the firewall or manages to break in the firewall security and tries to have access on any system in the trusted side and alerts the system administrator in case there is a breach in security .Moreover, Firewalls do a very good job of filtering incoming traffic from the Internet; however, there are ways to circumvent the firewall. For example, external users can connect to the Intranet by dialing in through a modem installed in the private network of the organization. This kind of access would not be seen by the firewall. Therefore, an Intrusion detection system (IDS) is a security system that monitors computer systems and network traffic and analyzes that traffic for possible hostile attacks originating from outside the organization and also for system misuse or attacks or iginating from inside the organization.

2.2 Literature Survey on Intrusion Detection System

Chunjie Zhou, Shuang Huang, et al.[1] in this paper they have used an anomaly detection based on multimodel has proposed and intelligent detection algorithms are designed. Classifier based on an intelligent hidden Markov model. A novel multimodel-based anomaly intrusion detection system with embedded intelligence and resilient coordination for the field control system in industrial process automation is designed. In this system, an anomaly detection based on multimodel has proposed, and the corresponding intelligent detection algorithms are designed. Furthermore, to overcome the disadvantages of anomaly detection, a classifier based on an intelligent hidden Markov model, have designed to differentiate the actual attacks from faults. The unique feature of the proposed intelligent intrusion detection system has that it uses complete multiple models of PCS built through the integration of multi domain knowledge to detect system anomaly, and employs HMM models to identify attacks from the sequential anomaly alerts. So in conclusion, the proposed intelligent intrusion detection system can detect the attack from both spatial and temporal aspects. In addition, since our intrusion detection system developed for PCSs takes into account the system knowledge instead of attack signatures, unknown type and new type of attack can also be detected by the proposed IDS. This paper is based on anomaly intrusion detection without consideration of attack knowledge. For the future, a comprehensive intrusion detection system for PCSs, which integrates system knowledge and attack knowledge, will be researched to optimize resources and time.

Al-Jarrah, O. ; Dept. of Comput. Eng., et al.[2] in this paper they have used an intelligent system to maximize the recognition rate of network attacks by embedding the temporal behavior of the attacks into a TDNN neural network structure. The proposed system consists of five modules: packet capture engine, preprocessor, pattern recognition, classification, and monitoring and alert module. This system captures packets in real time using a packet capture engine that presents the packets to a preprocessing stage using two pipes. The preprocessing stage extracts the relevant features for port scan and host sweep attacks, stores the features in a tapped line of a TDNN, and produces outputs that represent possible attack behaviors in a pre-specified number of packets. These outputs are used by the pattern recognition neural networks to recognize the 23 attacks, which are classified, by the classifier network to generate attack alerts. DARPA data sets are used to evaluate the systems in terms of recognition capability and throughput. Test results show that this system detects all types of attacks much faster than rule based systems such as SNORT.

Chapter 3:-Methodology

The intrusion detection model used in this project is consist of five parts: selection of dataset for training and testing phase, pre-processing of dataset, classification method, training and testing the system. The different phases of the implemented methodology is shown in the Figure 1.

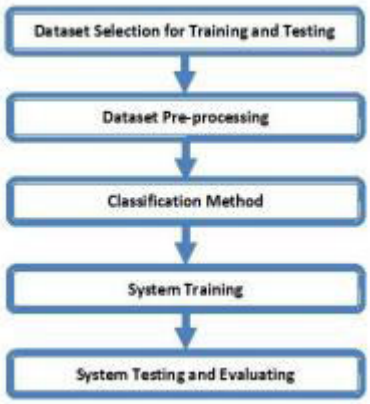


Fig. 3.2 Phases of Implemented Methodology

3.1 Dataset Selection and Pre-Processing

data set has been used as a benchmark dataset for training and testing the proposed system and the first part of analysis engine component of the IDS model is the pre-processing dataset. The preprocessing of dataset has a great importance as it results in increasing the efficiency of the detection for cases of training, testing, and classifying network activity into normal and abnormal. Pre-processing over the original dataset is a substantial step to make the dataset suitable and appropriate for the IDS structure.

3.2 Classification Method

Classification task is an important part of the constructed model for detecting intrusions, the classification model that has been used for this model is Logistic Model Trees (LMT). Classification Trees with Logistic Regression Functions for Network Based Intrusion Detection .

This classifier is a product of combining the tree structure and the function of logistic regression to construct a single decision tree that has the function of logistic regression in the leaves , that is providing a model of piecewise linear regression which is a real valued function , while the ordinary decision trees that have constants at their leaves produced the piecewise constant model which is a function that has the same output vale for each input value . The LogitBoost which is a boosting algorithm used to product the Logistic Regression model for every node of the constructed tree; the node is then splitting by the C4.5 criterion. The basic Logistic Model Trees induction algorithm uses the cross validation for finding the number of iterations of the LogitBoost that doesn't over fit the training data.

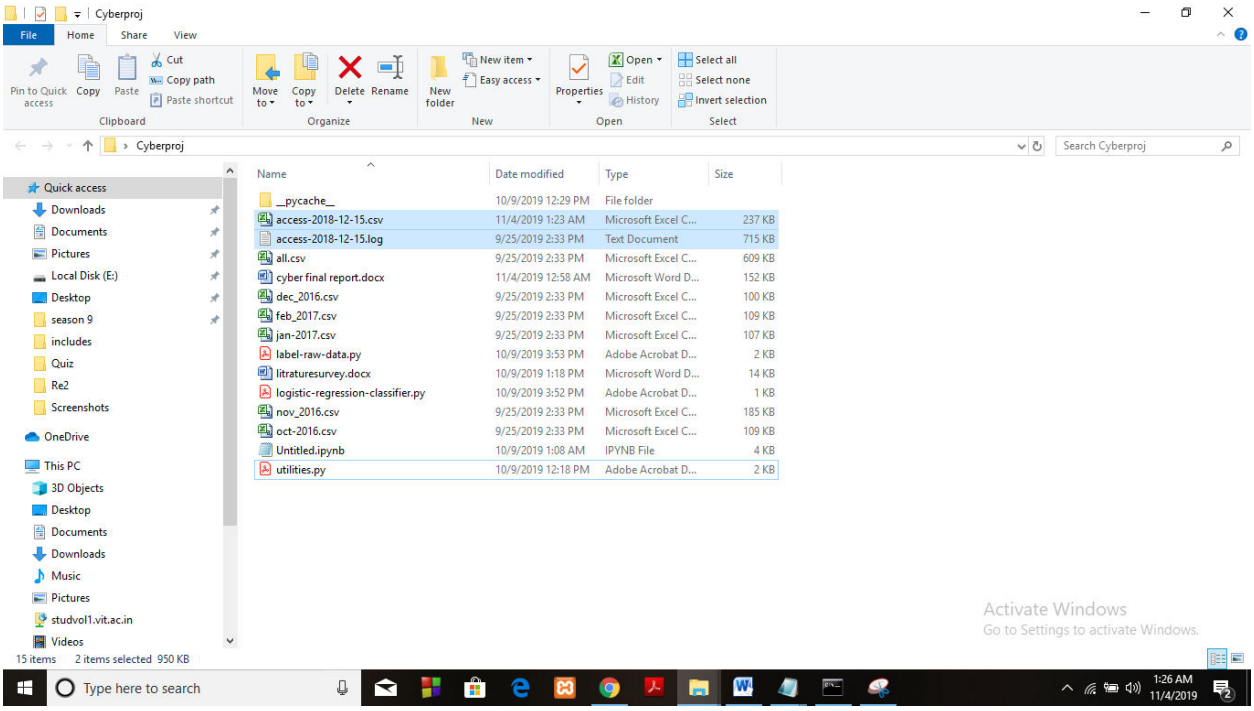
3.3 System Training

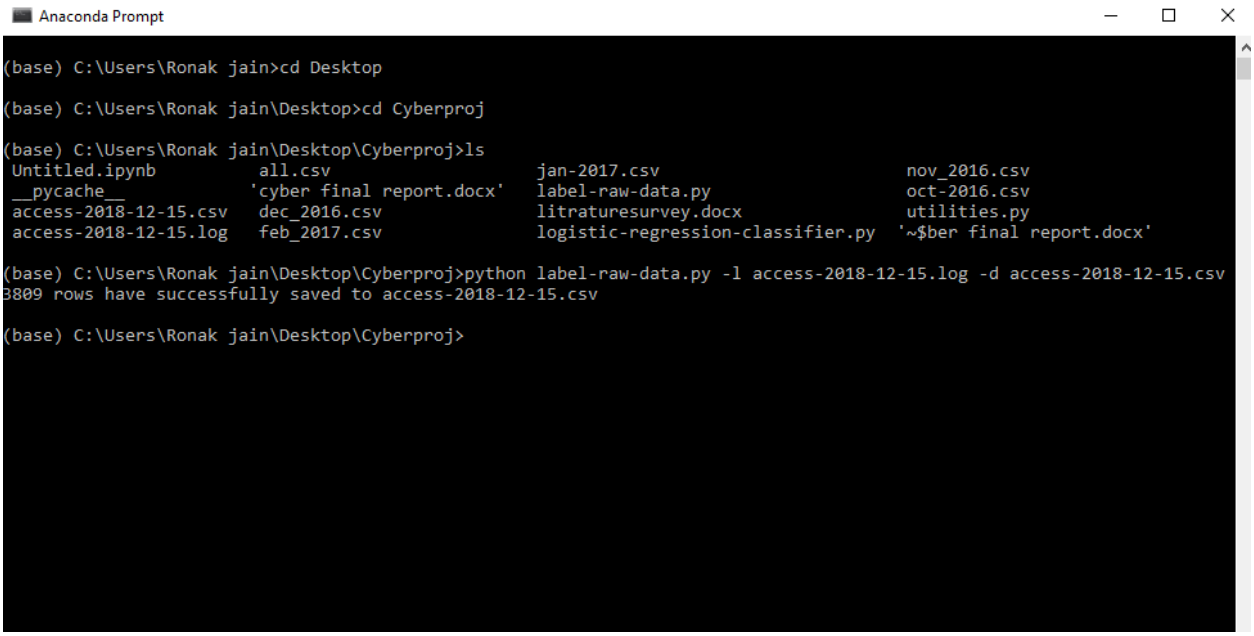
The training phase of the implemented model is about feeding the algorithm by the dataset to construct the classification model which is based on classification tree with logistic regression. Data set has been used with a split percentage (60%) as the training set and the rest for the testing set and two class (Normal, Attack) classification has been performed.

A data mining algorithm is applied to the above audit data to find the frequent patterns (i.e. to study the normal behavior of the secretary). The original association rules algorithm searches for all possible frequent associations among the set of given features. However, not all associations are necessarily useful for analyzing program or user behavior. Certain features are essential in describing the data, while others provide only auxiliary information. Domain knowledge is used to determine the appropriate essential features for an application.

Chapter 4 – Results

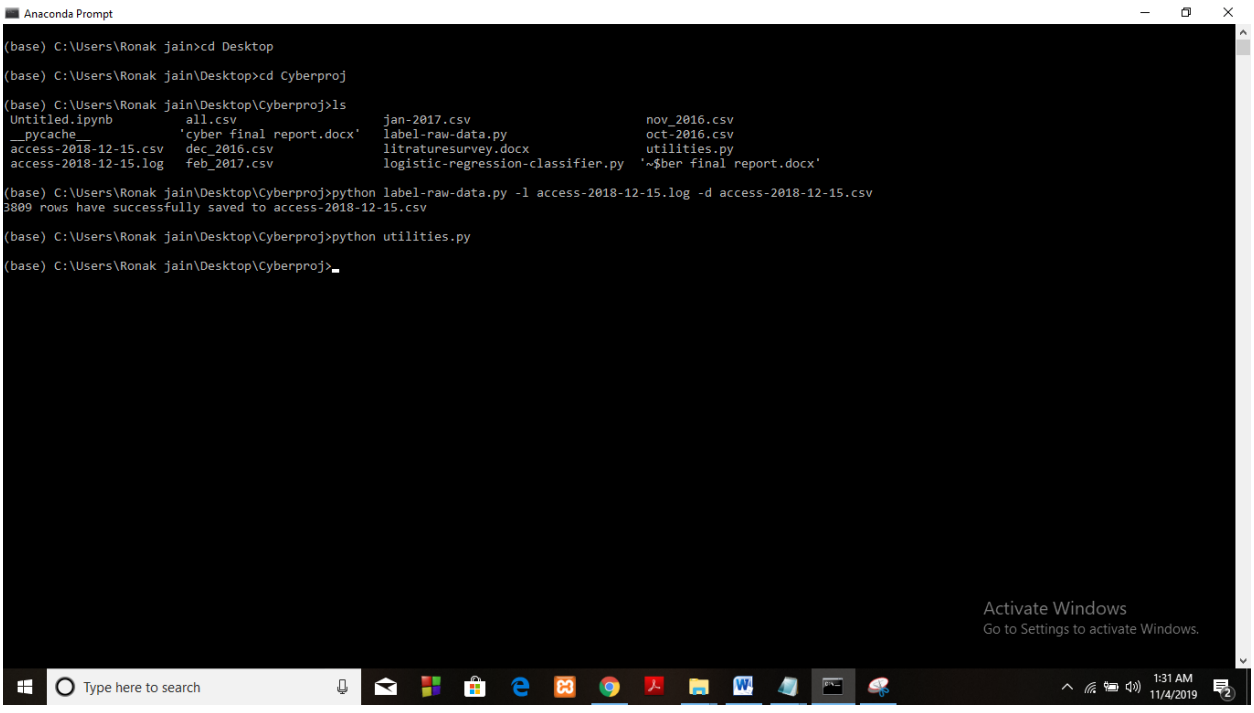
Data inserted from log file to csv file:



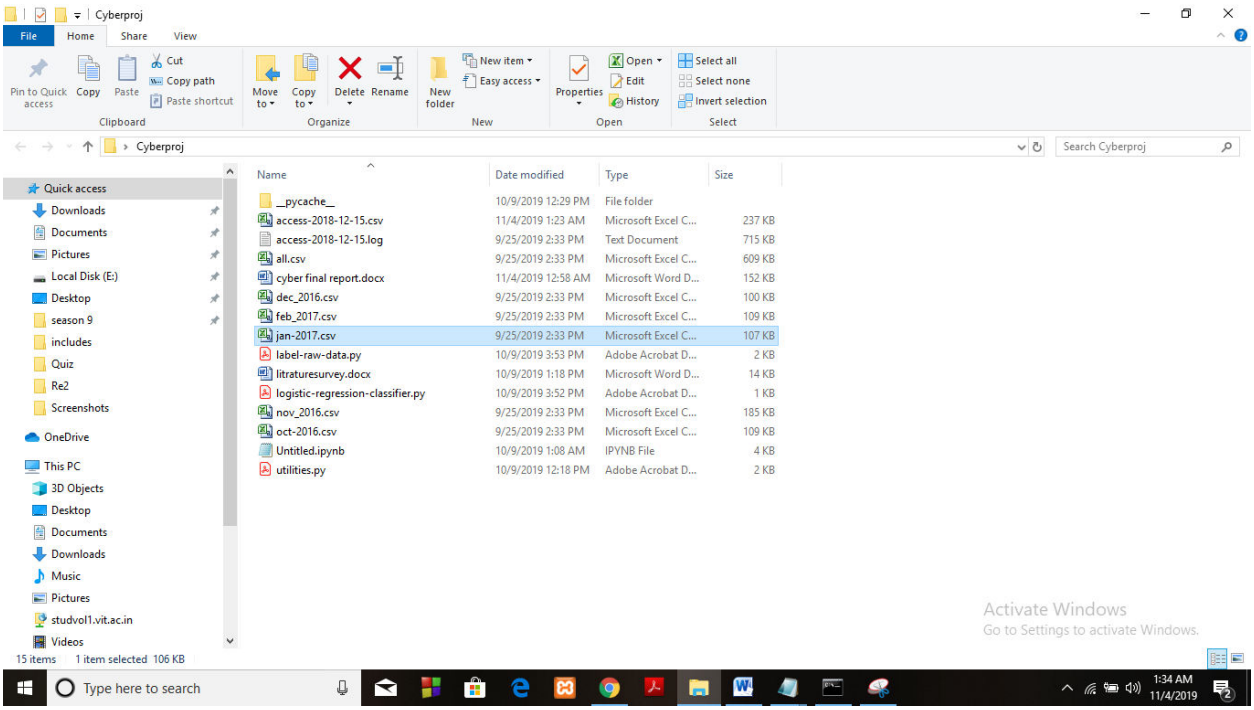


Compiling utilities.py:

Which load compare the trained data in which all libraries of attacks are mentioned to the sample data file.

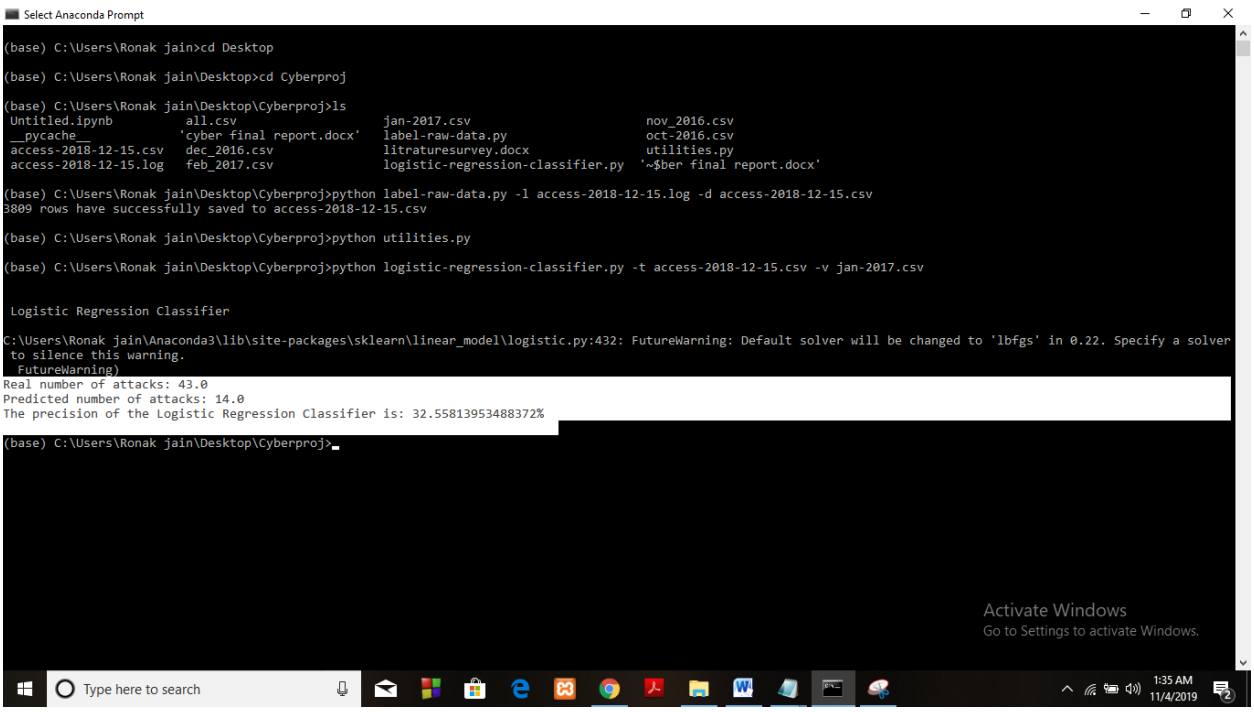


Sample csv file: jan-2017.csv(in our case)

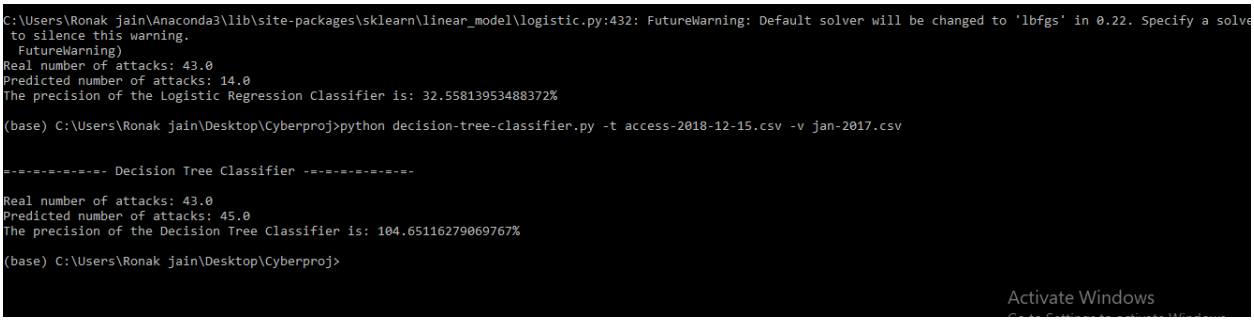


Logistic-regrassion-classifier.py

Attacks happen , predicted attacks ,precision values:



Decision tree classifier implementation:



Chapter 5 - Conclusions and Future Scope

5.1 Conclusion

As we know Signature-based IDS refers to the detection of attacks by looking for specific patterns, such as byte sequences in network traffic, or known malicious instruction sequences used by malware. This terminology originates from anti-virus software, which refers to these detected patterns as signatures.

So this project is helpful for detecting signature based attack. Specifically the pattern that we have selected is a web site ip address and that is only the one type attack that is faced. But that can be implemented in many other fields like web attack.Basically it compare with the known number of attacks which can be manully updated (if in case a new kind of attack is introduced). Any kind of pattern can be compared here.

5.2 Future Scope

During our project work we read some literature which described some novel ways of constructing polymorphic worms which makes it difficult to detect these worms. We intend to study these methods and see if the CFG based approach works against such worms. We also intend to come up with some methods to generate polymorphic worms which completely alter the control flow graph of the executable code and then design some technique to detect such worms.

We can also apply Decision tree classifier to implement singnature based attack.it can also show the attacks happen and predicted number of attacks of trained data.

5.3 Bibliography

Kruegel, Christopher, and Thomas Toth. "Using decision trees to improve signature-based intrusion detection." *International Workshop on Recent Advances in Intrusion Detection*. Springer, Berlin, Heidelberg, 2003.

Anjum, Farooq, Dhanant Subhadrabandhu, and Saswati Sarkar. "Signature based intrusion detection for wireless ad-hoc networks: A comparative study of various routing protocols." *2003 IEEE 58th Vehicular Technology Conference. VTC 2003-Fall (IEEE Cat. No. 03CH37484)*. Vol. 3. IEEE, 2003.

Stein, Gary, et al. "Decision tree classifier for network intrusion detection with GA-based feature selection." *Proceedings of the 43rd annual Southeast regional conference-Volume 2*. ACM, 2005.

Abbes, Tarek, Adel Bouhoula, and Michaël Rusinowitch. "Protocol analysis in intrusion detection using decision tree." *International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004.*. Vol. 1. IEEE, 2004.

Logistic regression and classifier- https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8

Du, Wenliang, and Zhijun Zhan. "Building decision tree classifier on private data." *Proceedings of the IEEE international conference on Privacy, security and data mining-Volume 14*. Australian Computer Society, Inc., 2002.

5.4 Appendix

Label-raw-data.py

```
import re

import sys

import argparse


parser = argparse.ArgumentParser()

parser.add_argument('-l', '--log_file', help = 'The raw http log file', required = True)

parser.add_argument('-d', '--dest_file', help = 'Destination to store the resulting csv file', required = True)

args = vars(parser.parse_args())


log_file = args['log_file']

dest_file = args['dest_file']



# Retrieve data form a a http log file (access_log)

def extract_data(log_file):

regex = '([(\d\.)]+) - - \[(.*?)\] "(.*?)" (\d+) (.+) "(.*?)" "(.*?)"'

data = {}

log_file = open(log_file, 'r')

for log_line in log_file:

log_line=log_line.replace(',','_')

log_line = re.match(regex,log_line).groups()

size = str(log_line[4]).rstrip('\n')

return_code = log_line[3]

url = log_line[2]

param_number = len(url.split('&'))
```

```
url_length = len(url)
if '-' in size:
size = 0
else:
size = int(size)
if (int(return_code) > 0):
charcs = {}
charcs['size'] = int(size)
charcs['param_number'] = int(param_number)
charcs['length'] = int(url_length)
charcs['return_code'] = int(return_code)
data[url] = charcs
return data



# Label data by adding a new raw with two possible values: 1 for attack or suspecious activity and 0 for normal behaviour
def label_data(data,labeled_data):
for w in data:
attack = '0'
patterns = ['honeypot', '%3b', 'xss', 'sql', 'union', '%3c', '%3e', 'eval']
if any(pattern in w.lower() for pattern in patterns):
attack = '1'
data_row = str(data[w]['length']) + ',' + str(data[w]['param_number']) + ',' + str(data[w]['return_code']) + ',' + attack + ',' + w + '\n'
labeled_data.write(data_row)
print (str(len(data)) + ' rows have successfully saved to ' + dest_file)


label_data(extract_data(log_file),open(dest_file, 'w'))
```

utilities.py

```
import sys
import numpy as np
from sklearn import tree, linear_model
import argparse

def get_args():
parser = argparse.ArgumentParser()
parser.add_argument('-t', '--traning_data', help = 'Training data', required = True)
parser.add_argument('-v', '--testing_data', help = 'Testing data', required = True)
return vars(parser.parse_args())


def get_data_details(csv_data):
data = np.genfromtxt(csv_data, delimiter = ",")
features = data[:, [0, 1, 2]]
labels = data[:, 3]
return features, labels


def get_occuracy(real_labels, predicted_labels, fltr):
real_label_count = 0.0
predicted_label_count = 0.0

for real_label in real_labels:
if real_label == fltr:
real_label_count += 1
```

```
for predicted_label in predicted_labels:
if predicted_label == fltr:
predicted_label_count += 1

print ("Real number of attacks: " + str(real_label_count))
print( "Predicted number of attacks: " + str(predicted_label_count))

precision = predicted_label_count * 100 / real_label_count
return precision
```

logistic-regression-classifier.py

```
from utilities import *

args = get_args()
traning_data = args['traning_data']
testing_data = args['testing_data']

# Get training features and labeles
training_features, traning_labels = get_data_details(traning_data)

# Get testing features and labels
testing_features, testing_labels = get_data_details(testing_data)

# LOGISTIC REGRESSION CLASSIFIER
print( "\n\n Logistic Regression Classifier \n")

attack_classifier = linear_model.LogisticRegression(C = 1e5)
attack_classifier.fit(training_features, traning_labels)
```

```
predictions = attack_classifier.predict(testing_features)

print ("The precision of the Logistic Regression Classifier is: " +
str(get_occuracy(testing_labels,predictions, 1)) + "%")
```

decision-tree-classifier.py

```
# About: Use supervised learning decision tree classifier to predict intrusion/suspecious activities in http logs

# Author: walid.daboubi@gmail.com

# Version: 1.2 - 2019/07/13


from utilities import *


args = get_args()

traning_data = args['traning_data']

testing_data = args['testing_data']


# Get training features and labeles

training_features, traning_labels = get_data_details(traning_data)


# Get testing features and labels

testing_features, testing_labels = get_data_details(testing_data)


# DECISON TREE CLASSIFIER

print ("\n\n=-=-=-=-=-=-=-=- Decision Tree Classifier -=-=-=-=-=-=-=-=\n")


# Instanciate the classifier

attack_classifier = tree.DecisionTreeClassifier()


# Train the classifier
```

```
attack_classifier = attack_classifier.fit(training_features, traning_labels)


# Get predections for the testing data

predictions = attack_classifier.predict(testing_features)


print( "The precision of the Decision Tree Classifier is: " + str(get_occuracy(testing_labels,predictions,
1)) + "%")
```