

Analysis of Efficient Cyber Hacking Breaches Using Machine Learning

¹Achanta Ayyappa, ²P.Rama Krishna

^{1,2}Dept. of CSE, KIET, Kakinada, AP, India

Abstract

Contrasted with the past, improvements in PC and correspondence innovations have given broad and propelled changes. The use of new innovations give incredible advantages to people, organizations, and governments, be that as it may, messes some up against them. For instance, the protection of significant data, security of put away information stages, accessibility of information and so forth. Contingent upon these issues, digital fear based oppression is one of the most significant issues in this day and age. Digital fear, which made a great deal of issues people and establishments, has arrived at a level that could undermine open and nation security by different gatherings, for example, criminal association, proficient people and digital activists. Along these lines, Intrusion Detection Systems (IDS) has been created to maintain a strategic distance from digital assaults.

Right now, learning the bolster support vector machine (SVM) calculations were utilized to recognize port sweep endeavors dependent on the new CICIDS2017 dataset with 97.80%, 69.79% precision rates were accomplished individually. Rather than SVM we can introduce some other algorithms like random forest, CNN, ANN where these algorithms can acquire accuracies like SVM – 93.29, CNN – 63.52, Random Forest – 99.93, ANN – 99.11.

Keywords

Hacking Breach, Data Breach, Cyber Threats, Cyber Risk Analysis,

Breach Prediction, Trend Analysis, Time Series, Cyber Security Data Analytics

I. Introduction

A. Objective of the Project

The use of new innovations give incredible advantages to people, organizations, and governments, be that as it may, messes some up against them. For instance, the protection of significant data, security of put away information stages, accessibility of information and so forth. Contingent upon these issues, digital fear based oppression is one of the most significant issues in this day and age. Digital fear, which made a great deal of issues people and establishments, has arrived at a level that could undermine open and nation security by different gatherings, for example, criminal association, proficient people and digital activists. Along these lines, Intrusion Detection Systems (IDS) has been created to maintain a strategic distance from digital assaults.

II. Proposed System

The important steps of the algorithm are given in below.

1. Normalization of every dataset.
2. Convert that dataset into the testing and training.
3. Form IDS models with the help of using RF, ANN, CNN and SVM algorithms.
4. Evaluate every model's performances

Methodology

```

: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

: import itertools
import seaborn as sns
import pandas_profiling
import statsmodels.formula.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from patsy import dmatrices

/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use the functions in the public API at pandas.testing instead.
import pandas.util.testing as tm

: from sklearn import datasets
from sklearn.feature_selection import RFE
import sklearn.metrics as metrics
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2, f_classif, mutual_info_classif

: train=pd.read_csv('/content/drive/My Drive/kdd/NSL Dataset/Train.txt',sep=',')
test=pd.read_csv('/content/drive/My Drive/kdd/NSL Dataset/Test.txt',sep=',')

```

Data preprocessing

```

n [6]: columns=["duration","protocol_type","service","flag","src_bytes","dst_bytes","land",
"wrong_fragment","urgent","hot","num_failed_logins","logged_in",
"num_compromised","root_shell","su_attempted","num_root","num_file_creations",
"num_shells","num_access_files","num_outbound_cmds","is_host_login",
"is_guest_login","count","srv_count","error_rate","srv_error_rate",
"error_rate","srv_error_rate","same_srv_rate","diff_srv_rate","srv_diff_host_rate","dst_host_count","dst_host_sr
"dst_host_diff_srv_rate","dst_host_same_src_port_rate",
"dst_host_srv_diff_host_rate","dst_host_error_rate","dst_host_srv_error_rate",
"dst_host_rerror_rate","dst_host_srv_rerror_rate","attack","last_flag"]

n [7]: train.columns=columns
test.columns=columns

n [8]: train.head()

```

ut[8]:	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_compromised	root_
0	0	udp	other	SF	146	0	0	0	0	0	0	0	0	0
1	0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0
2	0	tcp	http	SF	232	8153	0	0	0	0	0	1	0	0
3	0	tcp	http	SF	199	420	0	0	0	0	0	1	0	0
4	0	tcp	private	REJ	0	0	0	0	0	0	0	0	0	0

```

n [9]: test.head()

```

Data EDA

```

: # Protocol type distribution
plt.figure(figsize=(9,8))
sns.countplot(x="protocol_type", data=train)
plt.show()

```

Model Building

```

train_X=train_new[cols]
train_y=train_new['attack_class']
test_X=test_new[cols]
test_y=test_new['attack_class']

```

ML Deploy

```

Logistic Regression

# Building Models
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(random_state=0,solver='lbfgs',multi_class='multinomial')
logreg.fit( train_X, train_y)
logreg.predict(train_X) #by default, it use cut-off as 0.5

list( zip( cols, logreg.coef_[0] ) )

logreg.intercept_

logreg.score(train_X,train_y)

```

Decision Trees

```
train_X.shape
```

```
param_grid = {'max_depth': np.arange(2, 12),
              'max_features': np.arange(10,15)}
```

```
train_y.shape
```

```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier, export_graphviz, export
tree = GridSearchCV(DecisionTreeClassifier(), param_grid, cv = 10,verbose=1,n_jobs=-1)
tree.fit( train_X, train_y )
```

```
tree.best_score_
```

```
tree.best_estimator_
tree.best_params_
```

```
train_pred = tree.predict(train_X)
```

```
print(metrics.classification_report(train_y, train_pred))
```

```
test_pred = tree.predict(test_X)
```

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
pargrid_rf = {'n_estimators': [50,60,70,80,90,100],
              'max_features': [2,3,4,5,6,7]}
```

```
from sklearn.model_selection import GridSearchCV
gscv_rf = GridSearchCV(estimator=RandomForestClassifier(),
                       param_grid=pargrid_rf,
                       cv=10,
                       verbose=True, n_jobs=-1)
gscv_results = gscv_rf.fit(train_X, train_y)
```

```
gscv_results.best_params_
```

```
gscv_rf.best_score_
```

```
radm_clf = RandomForestClassifier(oob_score=True,n_estimators=80, max_features=5, n_jobs=-1)
radm_clf.fit( train_X, train_y )
```

```
radm_test_pred = pd.DataFrame( { 'actual': test_y,
                                'predicted': radm_clf.predict( test_X ) } )
```

Support Vector Machine (SVM)

```
from sklearn.svm import LinearSVC
svm_clf = LinearSVC(random_state=0, tol=1e-5)
svm_clf.fit(train_X,train_y)
```

```
print(svm_clf.coef_)
print(svm_clf.intercept_)
print(svm_clf.predict(train_X))
```

```
from sklearn.svm import SVC
from sklearn.pipeline import make_pipeline
model = SVC(kernel='rbf', class_weight='balanced',gamma='scale')
```

```
model.fit(train_X,train_y)
```

```
from sklearn.model_selection import GridSearchCV
param_grid = {'C':[1, 10],
              'gamma': [0.0001, 0.001]}
grid = GridSearchCV(model, param_grid)
grid.fit(train_X,train_y)
```

```
print(grid.best_params_)
```

From the score accuracy we concluding the DT & RF give better accuracy and building pickle file for predicting the user input Application

```
jupyter app.py 24/06/2020
File Edit View Language Python
3 import joblib
4
5 app = Flask(__name__)
6 model = joblib.load('model.pkl')
7
8 @app.route('/')
9 def home():
10     return render_template('index.html')
11
12 @app.route('/predict', methods=['POST'])
13 def predict():
14
15     int_features = [float(x) for x in request.form.values()]
16
17     if int_features[0]==0:
18         f_features=[0,0,0]+int_features[1:]
19     elif int_features[0]==1:
20         f_features=[1,0,0]+int_features[1:]
21     elif int_features[0]==2:
22         f_features=[0,1,0]+int_features[1:]
23     else:
24         f_features=[0,0,1]+int_features[1:]
25
26     if f_features[6]==0:
27         fn_features=f_features[:6]+[0,0]+f_features[7:]
28     elif f_features[6]==1:
29         fn_features=f_features[:6]+[1,0]+f_features[7:]
30     else:
31         fn_features=f_features[:6]+[0,1]+f_features[7:]
32
33     final_features = [np.array(fn_features)]
```

Localhost - in cmd python app.py

```
user@ramesh:~/Desktop/41/finished/second/3/Network-Intrusion-Detection-System-master$ python3 app.py
/home/user/.local/lib/python3.6/site-packages/sklearn/base.py:334: UserWarning:
Trying to unpickle estimator LogisticRegression from version 0.22.1 when using v
ersion 0.23.2. This might lead to breaking code or invalid results. Use at your
own risk.
  UserWarning)
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deploye
nt.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

<https://www.python.org>

Network Intrusion Detection System

Attack:
Other

Number of connections to the same destination host as the current connection in the past two seconds:
count

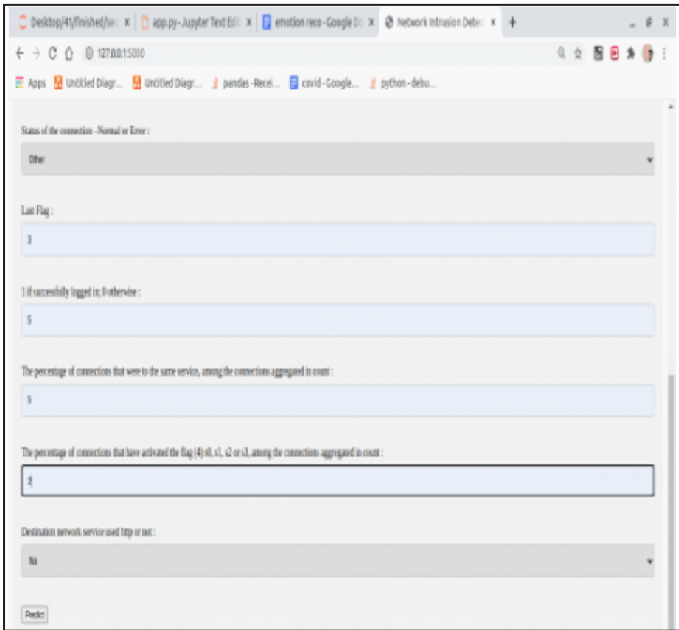
The percentage of connections that were to different services, among the connections aggregated in dst_host_count:
dst_host_dst_svc_rate

The percentage of connections that were to the same source port, among the connections aggregated in dst_host_svc_count:
dst_host_same_src_port_rate

The percentage of connections that were to the same service, among the connections aggregated in dst_host_count:
dst_host_same_svc_rate

Number of connections having the same port number:
dst_host_svc_count

Enter the input



Predict attack

Download the Correct version into the system

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: <https://www.python.org>

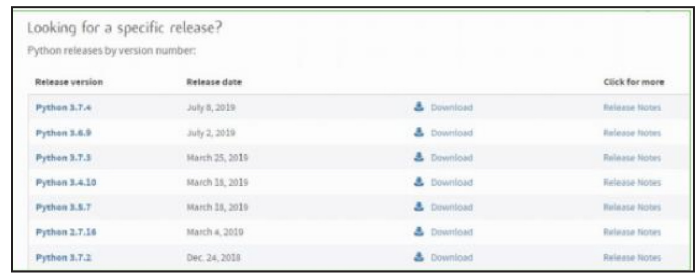


Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.

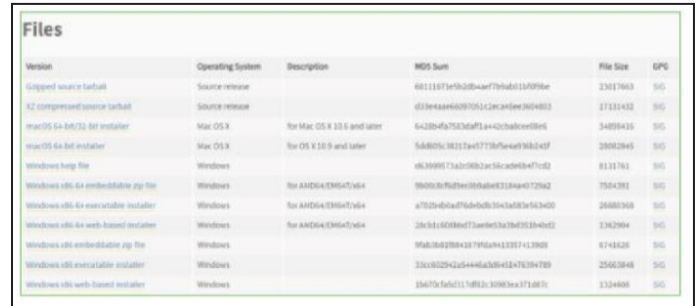


Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4



Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.

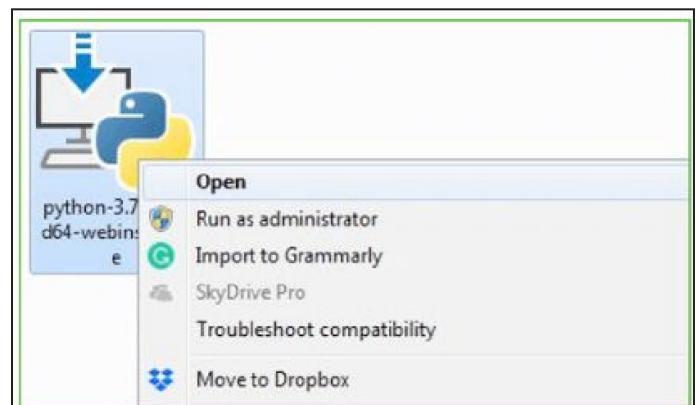


- To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
- To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

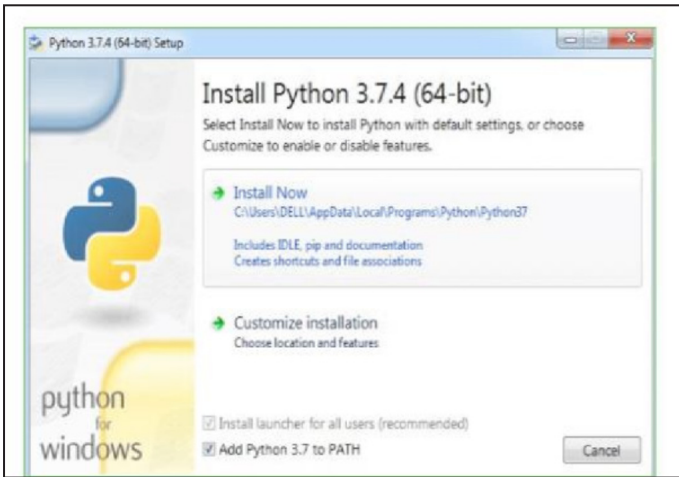
Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option. Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.



Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.



Step 3: Click on Install NOW After the installation is successful. Click on Close.

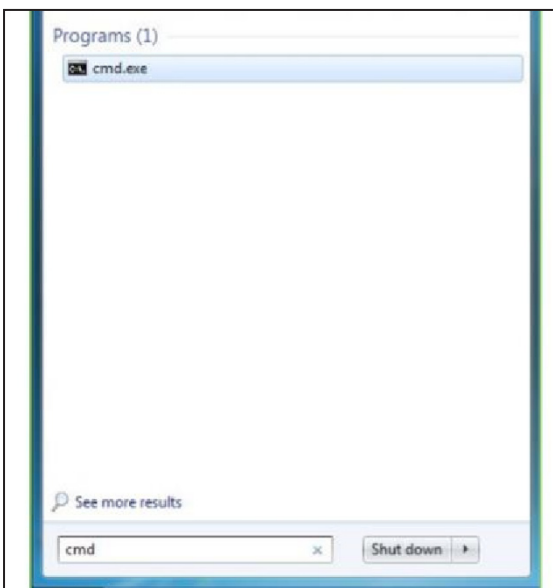


With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation. Note: The installation process might take a couple of minutes.

Verify the Python Installation

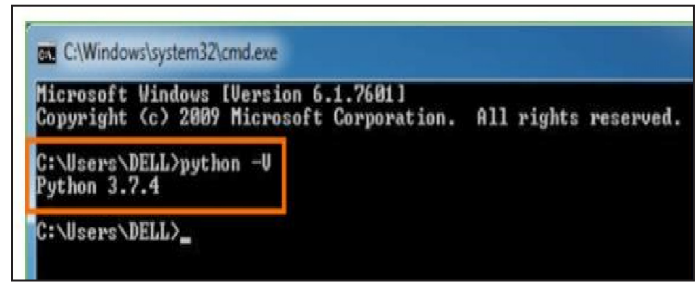
Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.



Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type python -V and press Enter.



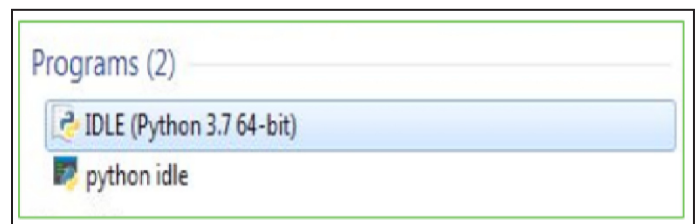
Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python IDLE works

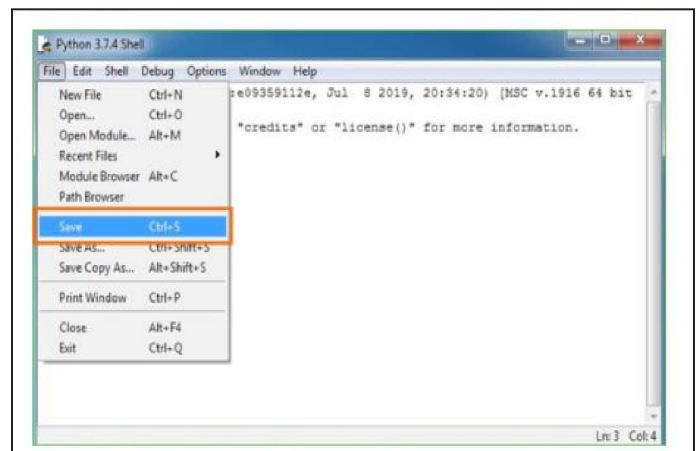
Step 1: Click on Start

Step 2: In the Windows Run command, type “python idle”.



Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. Click on File > Click on Save



Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. enter print

Conclusion

Right now, estimations of help vector machine, ANN, CNN, Random Forest and profound learning calculations dependent on modern CICIDS2017 dataset were introduced relatively. Results show that the profound learning calculation performed fundamentally preferable outcomes over SVM, ANN, RF and CNN. We are going to utilize port sweep endeavors as well as other assault types with AI and profound learning calculations, apache Hadoop and sparkle innovations together dependent on this dataset later on. All these calculation helps us to detect the cyber attack in network. It happens in the way that when we consider long back years there may be so many attacks happened so when these attacks are recognized then the features at which values

these attacks are happening will be stored in some datasets. So by using these datasets we are going to predict whether cyber attack is done or not. These predictions can be done by four algorithms like SVM, ANN, RF, CNN this paper helps to identify which algorithm predicts the best accuracy rates which helps to predict best results to identify the cyber attacks happened or not.

Reference

- [1] CERTPoliska Annual Report 2012. http://www.cert.pl/PDF/Report_CP_2012.pdf
- [2] SOPHOS homepage <http://www.sophos.com>
- [3] CiscoAnnualReport2013. http://www.cisco.com/web/about/ac49/ac20/ac19/ar2013/docs/2013_Annual_Report.pdf
- [4] BYOD: Bring Your Own Device. <http://www.vs.inf.ethz.ch/publ/papers/rohs-byod2004.pdf>
- [5] OWASP Top 10 2013. https://www.owasp.org/index.php/Top_10_2013-Top_10
- [6] NSG. <http://www.ijcst.com/vol31/4/sridevi.pdf>
- [7] LESG. <http://www.cs.northwestern.edu/~ychen/Papers/LESG-ICNP07.pdf>
- [8] A. Shabtai, E. Menahem and Y. Elovici. F-Sign: automatic, function-based signature generation for malware, systems, man, and cybernetics, Part C: applications and reviews. *Transactions on IEEE*, 41, 494–508, 2011.
- [9] D. Kong, J. Gong, S. Zhu, P. Liu and H. Xi. SAS: semantics aware signature generation for polymorphic worm detection. *International Journal of Information Security*, 50, 1–19, 2011.
- [10] M. Sharma and D. Toshniwal. Pre-clustering algorithm for anomaly detection and clustering that uses variable size buckets. *Recent Advances in Information Technology*, 515–519, 2012.
- [11] M. H. A. C. Adaniya, M. F. Lima, J. J. P. C. Rodrigues, T. Abrao and M. L. Proenca. Anomaly detection using DSNS and FireflyHarmonic Clustering Algorithm. *Communications (ICC)*, 1183–1187, 2012.
- [12] J. Mazel, P. Casas, Y. Labit and P. Owezarski. Sub-space clustering, Inter-Clustering Results Association and anomaly correlation for unsupervised network anomaly detection. *Network and Service Management (CNSM)*, 1–8, 24–28 October 2011.
- [13] Kaspersky Lab Security report. <http://www.securelist.com/en/analysis/204792244/Thegeography-of-cybercrime-WesternEurope-and-North-America>
- [14] 14 ESET threat report 12-2012. <http://go.eset.com/us/resources/threat-trends/GlobalThreatTrends-November-2012.pdf>
- [15] F. Felzenszwalb and P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59, 167–181, September 2004.
- [16] B. Needleman Saul and D. Wunsch Christian A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48, 443–453, 1970.
- [17] CSIC 2010 HTTP Dataset in CSV format. http://users.aber.ac.uk/pds7/csic_dataset/csic2010http.html