Machine Learning: Identification of attackers and clients during DDOS

Course:
Aprendizagem aplicada à segurança

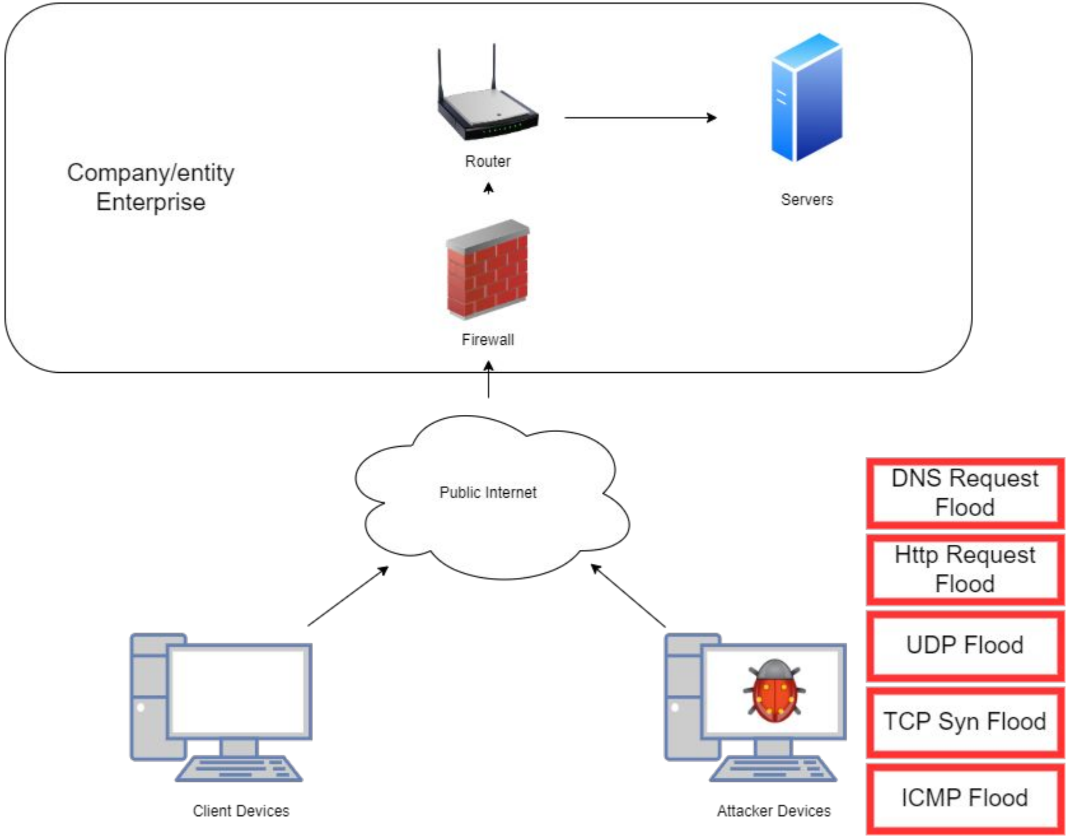
Group:
Guilherme Amaral Ribeiro Pereira 93134
José Luis Rodrigues Costa 92996

Objective

Distributed denial-of-service is one of the most popular and important attack of today's cyber world.

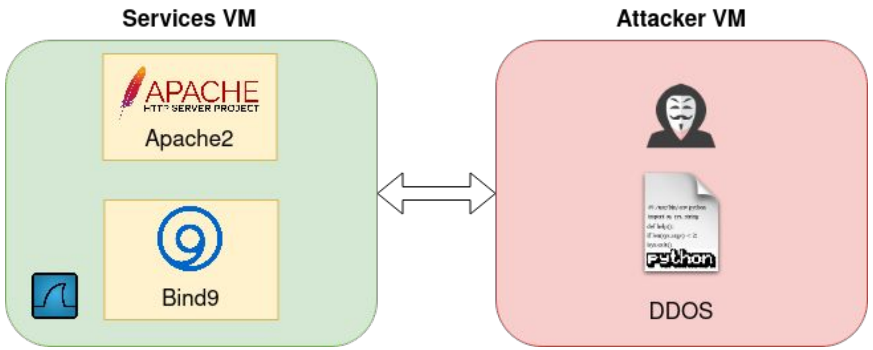
During a ddos attack it’s important to detect which users are real clients, and block the malicious.

Threat Model



Simulation - Attacker

- Simulation with 2 Virtual machines, one for the services and another to produce the attack.
- A python script was developed to perform these 5 types of attacks
- The services running are Apache2 and Bind9



Data sources - Attack data

Network Packets

Capture of network traffic using wireshark

Apache Logs

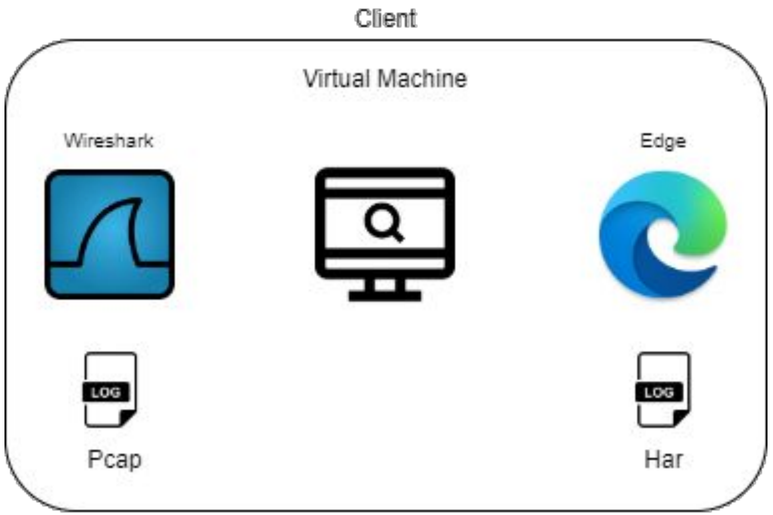
Retrieve information from apache log files

DNS Logs

Retrieve information from DNS log files

Simulation - Regular User

- Active website logs access was not available
- To simulate regular usage behavior, browser logs and packet capture was used
- Regular browsing as a normal user generated valid data



Data sources - Regular data

Network Packets

Capture of network traffic

Browser Logs

Browser log of all information.

Network Packets

DNS Logs

Apache Logs

Metrics

No. Packets

Packet Size Sum

No. TCP Packets

No. UDP Packets

No. ICMP Packets

No. Apache Req.

Number of requested webpages

Metrics

Apache Logged
User authenticated

No. Different Pages Accessed
Apache requests

No. DNS Queries

No. DNS Errors

Features

Number Packets

Max, Min and Mean

Packet Size

Max, Min and Mean

Number TCP Packets

Max, Min and Mean

Number UDP Packets

Max, Min and Mean

Number ICMP Packets

Max, Min and Mean

Number Apache Requests

Max, Min and Mean

Features (cont.)

Number Apache Errors

Max, Min and Mean

Apache Logged

User is logged in

Number of queries (Dns)

Max, Min and Mean

Number of errors (Dns)

Max, Min and Mean

Apache Time Request Variance

Time variance between apache requests

Variance on number of different pages

Variance on the number of different apache web pages requested

Features (cont.)

Source Ip Distance

Relative distance between the geolocations of the source ip's and the servers location

Total number of features:

31

Methodologies

First approach:

- One Class SVM
- One Class SVM using k-fold cross validation

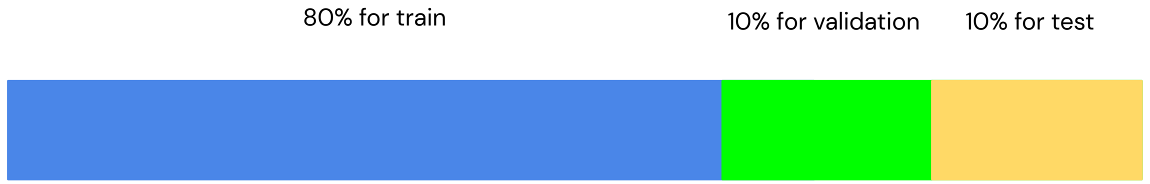
Second approach:

- Logistic Regression using k-fold cross validation

One Class SVM (Datasets)

We started by splitting our non anomalous data between the **train, validation** and **test** set.

Non anomalous data (829):



One Class SVM (Datasets)

Then we added an equal amount of DNS flood anomalies to our **validation** and **test** so the sets had the following sizes:

Train: 663 non anomalous data points

Validation: 166 data points where 83 are anomalous.

Test: 166 data points where 83 are anomalous

One Class SVM (Data normalization and PCA)

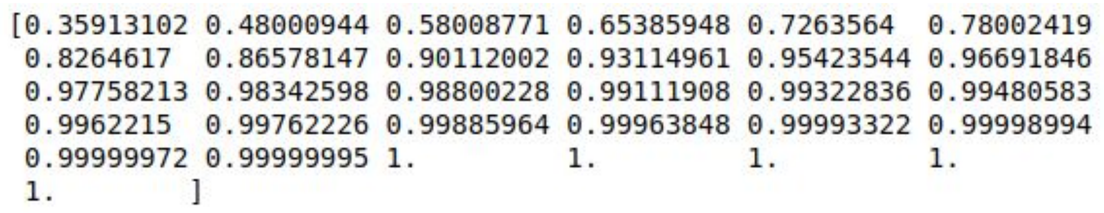
Taking into account the origin of the data our hypothesis was that our 31 features were highly correlated, so we could reduce the number of features without losing too much variance

Before applying PCA we needed to normalize the data by using:

- MinMaxScaler
- StandardScaler

One Class SVM (Data normalization and PCA)

By applying PCA we could reduce the number of features to the smallest number that kept 99 % of the variance which in our case was 16:



```
[0.35913102 0.48000944 0.58008771 0.65385948 0.7263564  0.78002419
 0.8264617  0.86578147 0.90112002 0.93114961 0.95423544 0.96691846
 0.97758213 0.98342598 0.98800228 0.99111908 0.99322836 0.99480583
 0.9962215  0.99762226 0.99885964 0.99963848 0.99993322 0.99998994
 0.99999972 0.99999995 1.         1.         1.         1.
 1.        ]
```

One Class SVM (Hyper-parameters)

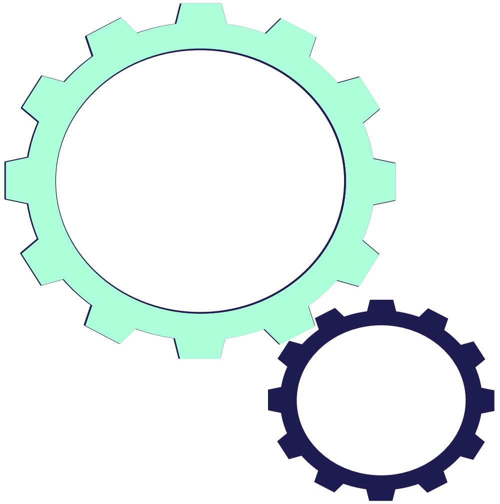
To obtain the best hyper parameters for one class svm we used the validation set and F1 score as an indicator.

One class svm hyper parameters values:

Nu: Values in range (0,1].

Kernel: [‘ linear ’ , ‘ poly ’ , ‘ rbf ’ , ‘ sigmoid ’].

Gamma: [' scale ' , ' auto ']



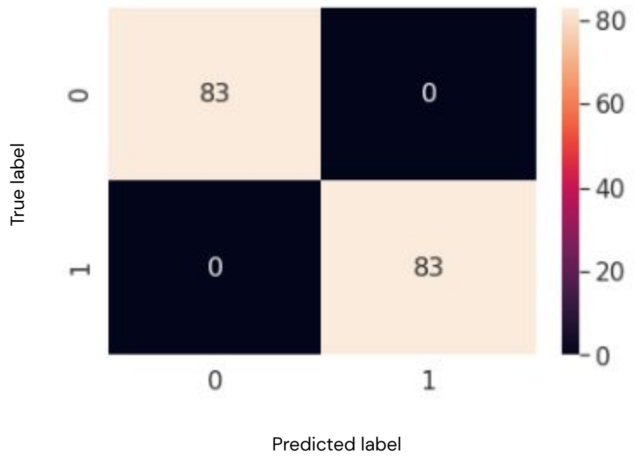
One Class SVM (Hyper-parameters)

The best hyper parameters were the following :
On the test we obtained a F1 score of 1.0 .

Nu: 0.03.

Kernel: sigmoid.

Gamma: scale



One Class SVM (Testing for other types of attack)

By using the previously trained model we tested it for the other 4 types of attacks using a small dataset of 96 non anomalies and 96 anomalies for udp icmp and HTTP and 65 anomalies and 96 non anomalies for tcp .

TCP SYN flood: F1 score of 0.93.

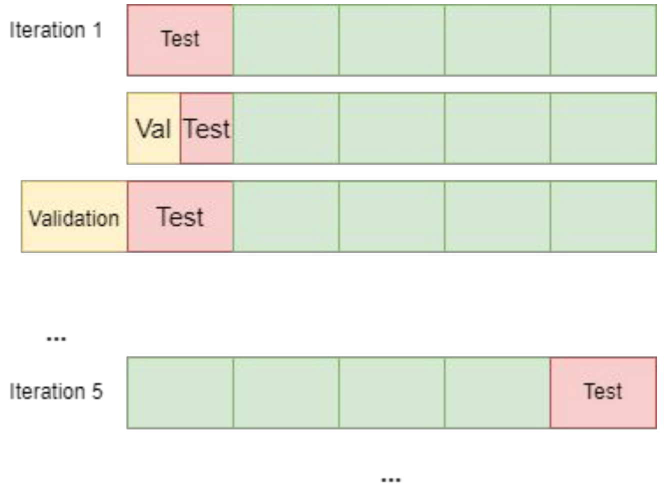
ICMP flood: F1 score of 1.0.

UDP flood: F1 score of 1.0.

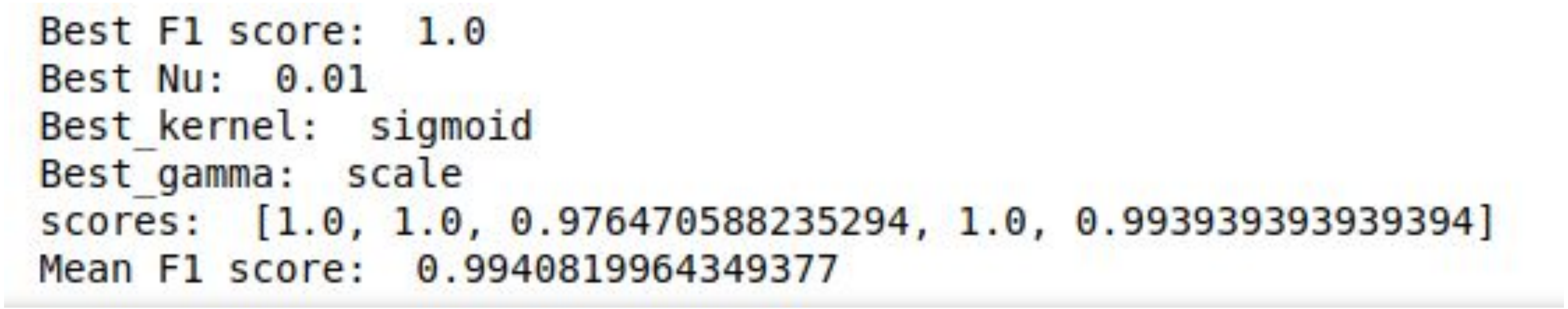
HTTP Request flood: F1 score of 0.89.

One Class SVM with K-Fold cross validation

The previous implementation presented a problem, the train, validation and test set split affected the outcome of the the model, so to we decided to implement k-fold to evaluate the machine learning model. Where k = 5



One Class SVM with K-Fold cross validation

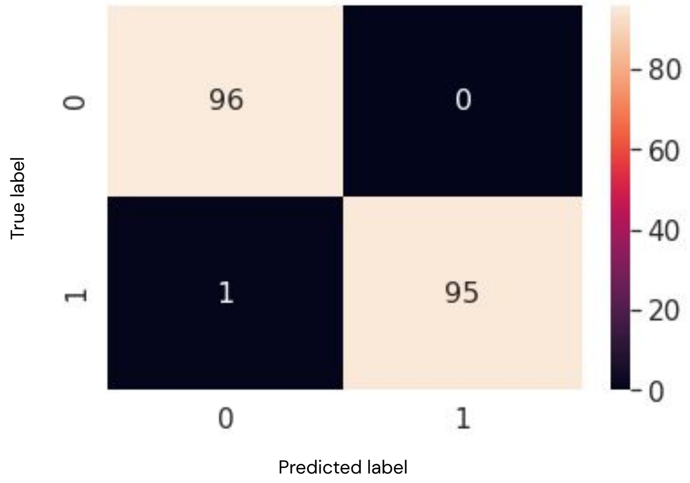


```
Best F1 score:  1.0
Best Nu:  0.01
Best_kernel:  sigmoid
Best_gamma:  scale
scores:  [1.0, 1.0, 0.9764705882352942, 1.0, 0.9939393939393939394]
Mean F1 score:  0.9940819964349377
```

One Class SVM with K-Fold cross validation

By “storing” the model that performed better in k-fold we tested once again for the other anomalies

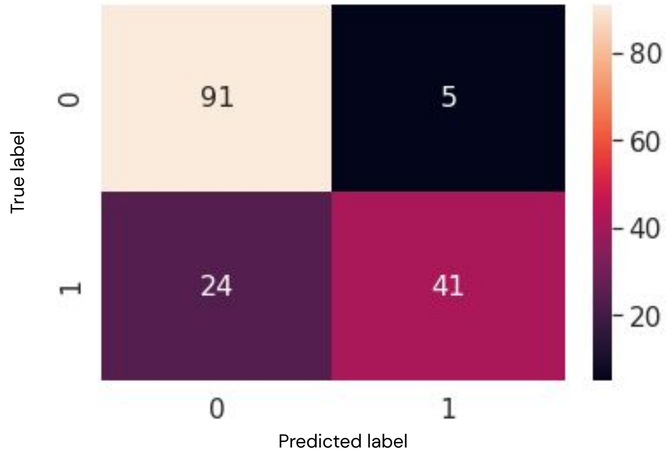
UDP flood: F1 score of 0.99.



One Class SVM with K-Fold cross validation

By “storing” the model that performed better in k-fold we tested once again for the other anomalies

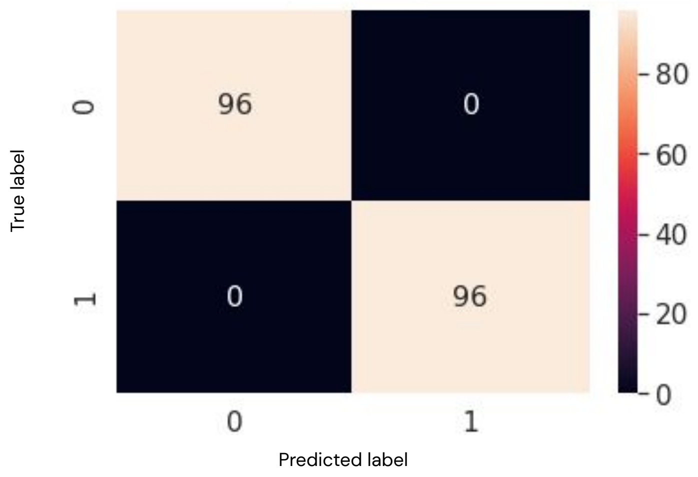
TCP SYN flood: F1 score of 0.74.



One Class SVM with K-Fold cross validation

By “storing” the model that performed better in k-fold we tested once again for the other anomalies

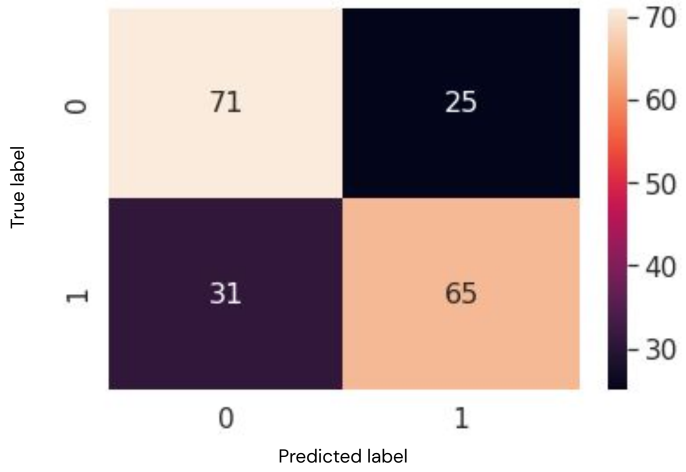
ICMP flood: F1 score of 1.0.



One Class SVM with K-Fold cross validation

By “storing” the model that performed better in k-fold we tested once again for the other anomalies

HTTP Request flood: F1 score of 0.69.



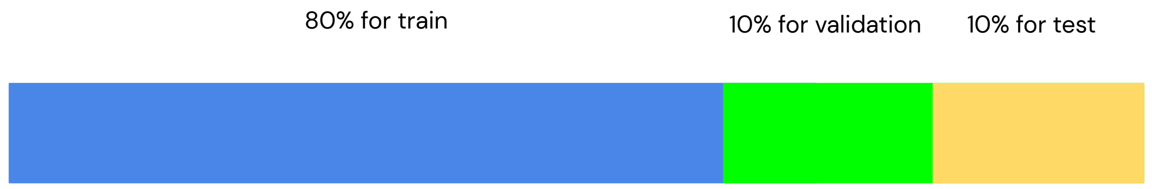
Logistic Regression with K-fold Cross validation

Since it was more easy for us to generate anomalous data, we decided that it would be interesting to see how a supervised algorithm would perform.

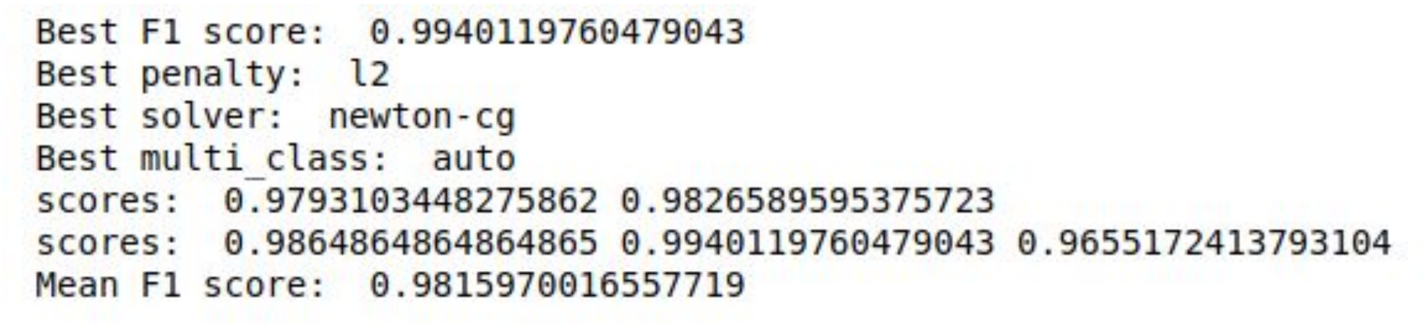
We started by attach the non anomalous data to the anomalous data of all the different attacks.

Non anomalous data + anomalous data (1658)

In each k-fold iteration we have:

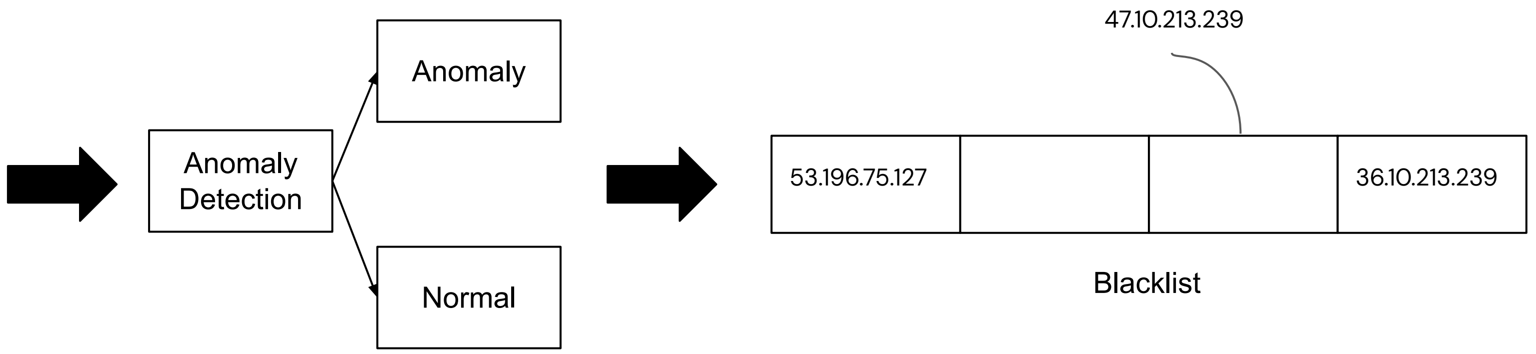


Logistic Regression with K-fold Cross validation



```
Best F1 score:  0.9940119760479043
Best penalty:  l2
Best solver:  newton-cg
Best multi_class:  auto
scores:  0.9793103448275862 0.9826589595375723
scores:  0.9864864864864865 0.9940119760479043 0.9655172413793104
Mean F1 score:  0.9815970016557719
```

Potencial blacklist model use



Anomaly detection

Creation of a black list based on the number of times an anomaly was detected for each IP

Future Improvements

- Bigger data set
- Real data from a web server, for both attacks and regular use
- Search for a better seed on the first approach
- Different metric and feature extraction

References

- Cloud flare DDoS trends". [Online], Available: https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/
- A Machine Learning Approach for DDoS Detection on IoT Devices [Online], Available: https://arxiv.org/pdf/2110.14911.pdf
- D-SCAP: DDoS Attack Traffic Generation Using Scapy Framework [Online], Available: https://link.springer.com/chapter/10.1007/978-981-13-1882-5_19
- Apache documentation [Online], Available: https://httpd.apache.org/docs/2.4/
- Bind9 documentation [Online], Available: https://bind9.readthedocs.io/en/latest/
- Scapy documentation [Online], Available: https://scapy.readthedocs.io/en/latest/

THANKS!

Do you have any questions?