PCS - 2 Final Project Report

Firewall implementation and Further Research

Rohit Doriya (B20AI034)

Naikawadi Shivaprasad (B20AI057)

Priyank Mandal (B20AI055)

—

Introduction:

The initial idea was to implement a firewall in order to block the illegal video streaming sites with the help of the concepts from the course, But when we finished with the project we realized that we still have nearly 1/4th of the course time left and we could implement additional features of research on a different firewall working principle.

Ideology and Implementation:

First part of the project (Firewall for Blocking Video streaming sites):

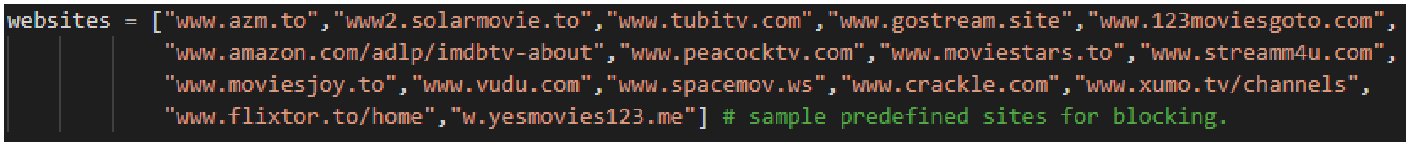
So, while exploring various approaches on implementing a firewall we came up with different ideas. Some included analyzing the packets sent and received from our machine to the server and deciding if the machine is accessing a video streaming site online and also which platform to implement the firewall.

Finally we came across a functionality called IPTABLES, that was available on linux and had the features to implement rules on the ip address that are allowed and not allowed by the machine.

So, after deciding the platform and package for implementing the firewall we needed versatility in our firewall as we can't just type out the rules for firewall one by one for every site. So we decided to add more functionality to our code by doing it in python and add features to develop a full fledged firewall for blocking illegal video streaming sites.

The idea for the working of the python firewall was that, we initially take a set of predefined websites that are infamous and well recognised as illegal video streaming sites.

The websites in this set were initially blocked when the firewall was initialized. The predefined set of websites can be set by the user and can have any range of sites depending on the computational power available.



```
websites = ["www.azm.to","www2.solarmovie.to","www.tubitv.com","www.gostream.site","www.123moviesgoto.com",
            "www.amazon.com/adlp/imdbtv-about","www.peacocktv.com","www.moviestars.to","www.streamm4u.com",
            "www.moviesjoy.to","www.vudu.com","www.spacemov.ws","www.crackle.com","www.xumo.tv/channels",
            "www.flixtor.to/home","w.yesmovies123.me"] # sample predefined sites for blocking.
```

Now we need to check if the websites feeded are actually existing and valid or not. This was done by parsing the output of <u>nslookup</u> by running a system command, and if the command gave an error code the website's ip wasn't stored and set to 0 in the array that stores the ip addresses of all the valid websites. And if the website was valid then the ip address was stored.

From all the valid ip stored we can now use the <u>IPTABLES</u> command to block each one of them.

<u>The command:</u> **sudo iptables -A INPUT -s <IP Address> -j DROP**

After the initial sites were blocked the firewall outputted a list of option for the user to command the firewall on:

- 1 <url of website> : To block a new website for the host machine.
- 2 <ip address> : To block a new ip address for the host machine.
- 3 <url of website> : To unblock a website for the host machine.
- 4 <ip address> : To unblock an ip address for the host machine.
- 5 : To block alternate sites of the previously blocked websites by brute force approach.
- 6 <string> : To block alternate sites of the previously blocked websites by string matching approach.
- 7 : To view the iptable rules for the blocked websites.
- EXIT : To close the firewall and restore the iptable rules

Respective functions were made for each of the options for further customizability of the user.

The main idea was to give the user the freedom to block new sites/IP addresses, unblock sites/Ip addresses and also block alternate or clone sites.

Blocking and Unblocking the sites and IP addresses was very easy and was just usage of the previously defined functions.

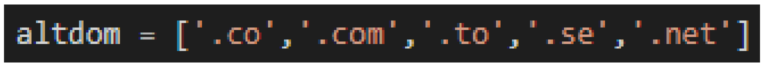
<u>Major Task:</u>

We know that there exists Alternate/Clone sites for illegal streaming sites as the creators of these sites do have alternates if one of the sites is blocked. We noticed that these sites are generally directly done by having a different Top Level Domain (TLD) or by adding extra words or random letters to the base domain.

To solve this task we had two approaches:

1) <u>Brute Force Approach:</u>

This was done by directly taking the sites from the blocked sites list and also taking a list of possible TLD's and making all possible combinations of such sites and then passing them through previously defined functions and checking if they are valid, and if valid then blocking them.



```
altdom = ['.co','.com','.to','.se','.net']
```

The number of alternate TLDs can be decided by the user based on the computational power available to allocate to the firewall (As all the possible combinations are checked.

<u>Limitation:</u> Couldn't block the sites with modified base domain names.

2) <u>String Matching Approach:</u>

The idea for this approach came to us while exploring the functionality of the IPTABLES package. It could allow and block sites by matching strings in the domain of the sites. This could solve the limitation of the first approach as it could now block the sites with modified domains that contain the base string in the domain name.

Command: `sudo iptables -A OUTPUT -p udp -m string --string <STRING> --algo kmp -j DROP`

These two approaches combined could solve the problem of alternate/clone sites.

This concluded the firewall program for blocking illegal video streaming sites.

Second Part (Machine Learning model to detect SQL injection):

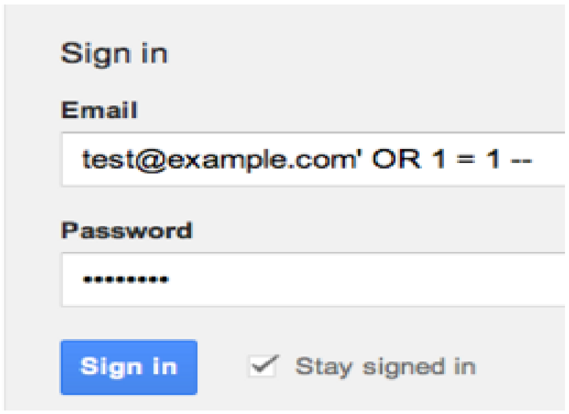
As we know that a full fledged firewall not only blocks the sites it is set to block but also has measures set in to block certain attacks (for the firewall used for servers for websites) to the databases by hackers/attackers.

This is generally referred to as SQL injections methods to breach the security of the database of a website and retrieve sensitive data/passwords from the database of the users. These occur due to the architecture of how the databases are built and how they respond to requests made to them.

Now we decided to implement this in our previously made firewall in order to make it a fully fledged firewall for security of servers. We decided to implement a Machine Learning model to detect breaches, as the SQL injections aren't fixed and many arise everyday and hence cannot be blocked by a brute force method.

The inspiration of such an implementation of a firewall comes from one of the newer inventions of NGFW which also uses machine learning to learn through the new and unknown threats and provide security.

The main idea for implementing this kind of a machine learning model by somehow finding patterns in the famous pre-existing SQL injections. This would be a problem similar to Natural Language Processing (NLP), as we could vectorize certain words from the total dataset of pre-defined legal and illegal SQL commands.



Email: test@example.com' OR 1 = 1 --

Password: •••••••••

This is an example of an SQL Injection on a username/password field. This command in the email part has certain commands that the SQL recognizes (The component of OR 1=1 - - in the statement is an equivalent to always TRUE and hence returns the respective password and username from the dataset) and due to which it is forced to return the data stored by it bypassing it's security.

Now from this example we can see that certain keywords like OR, 1=1, - - , etc occur frequently in the SQL injections and hence from this we can have an NLP preprocessing done on the dataset and then the rest is a binary classification problem and various models can be trained to get the best results.

For training this model we required a dataset to train on and which we have found online with data consisting of famous SQL injections and normal commands provided with a label signifying if it's illegal or legal.

For the NLP preprocessing we used a Countvectorizer from the Sklearn library that tokenizes the words from the dataset and then returns a CSR matrix for further analysis.

Coming to addressing the binary classification problem, we needed to decide upon a model that fits the problem best.

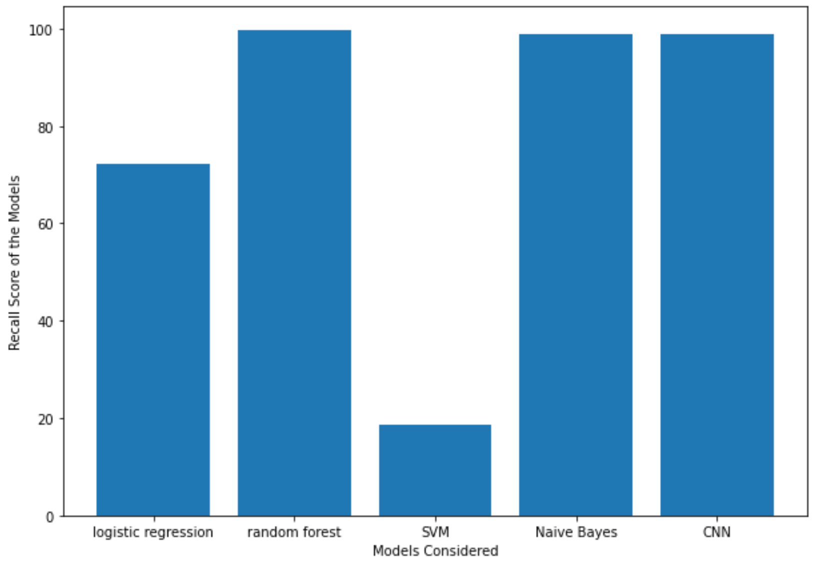
The decision criteria/metric was Recall as it is the measure of the ability of a model to predict the True Positives, which is required by us as we need to focus on maximizing the SQL injection even though we trade off some legal operation misclassification.

Decided upon the metric of the model we chose the following classifiers:

Logistic Regression, Random Forest, SVM, Naive Bayes, Neural Network.

We imported each of the following models from the respective modules, fitted the training data from the dataset and had predictions made for the testing dataset and calculated the accuracy, F1 score and also the Recall and stored the recall in an array.

This array of recall scores were later plotted to determine which model had better performance as recall was the deciding metric for performance.



From the obtained results, we could conclude that the Random Forest model performed the best and the Naive Bayes and CNN had an equal but slightly lower performance than that of the Random Forest model.