DDoS and DOS Attack detection using IDS

Daniele Pintore
71/10/00036

Gloria Saba
71/10/00043

Massimo Sanna
71/10/00011

Abstract: The internet usage is growing, the entire world relies on it heavily. With the increase on the usage, the number of attacks is also growing. A common attack used to make resources unavailable for the client is the DoS attack. The IDS system proposed in this paper tries to detect and block common DoS attacks.

I. Introduction

The rapid expansion of the Internet of Things (IoT) and cloud-based services has made network availability a critical asset for modern infrastructure. However, this has also widened the attack surface for malicious actors. Denial of Service (DoS) and Distributed Denial of Service (DDoS) are among the most common attacks. These attacks aim to damage the availability of services, like websites, making them inaccessible to legitimate users, by exhausting system resources such as bandwidth, CPU, or memory.

The specific layers of the OSI model involved in these attacks are the Network Layer and the Application Layer.

The Network Layer is responsible for delivering data packets from a source host to a destination host across one or more interconnected networks. Like all DDoS attacks, the goal of a layer 3 attack is to slow down or crash a program, service, computer, or network, or to fill up capacity so that no one else can receive service. Typically, this is accomplished by targeting network equipment and infrastructure.

The Application Layer is the top layer of the OSI model and provides network services directly to end-user applications, and enables communication between software applications and the network by defining protocols for data exchange. DDoS attacks involving the Application Layer have a malicious behavior designed to target the "top" layer in the OSI model, where common internet requests such as HTTP GET and HTTP POST occur. These attacks are particularly effective due to their consumption of server resources in addition to network resources. An example of attack in which the Application Layer is involved is the Slowloris attack, which operates by utilizing partial HTTP requests. The attack functions by opening connections to a targeted Web server and then keeping those connections open as long as they can. This type of attack uses a low amount of bandwidth, and aims to use up server resources with requests that seem slower than normal that mimic regular traffic.

II. Scenario

The current state of the art tools were developed for pen-testing purposes, but they are also used in malicious attack campaigns. Tools such as `DoS Hulk`, `GoldenEye`, `Slowloris`, and `Slowhttptest`, utilize advanced mechanisms to mimic legitimate user behavior. For instance, "Low and Slow" attacks like `Slowloris`, operate by

keeping many connections open with minimal bandwidth usage, effectively bypassing traditional firewall thresholds that rely on rate limiting.

Detecting these attacks requires an Intrusion Detection System (IDS). This kinds of systems often use machine learning algorithms that can analyze complex traffic features and identify anomalies that deviate from the baseline of normal traffic. It is important for this type of systems to have a fast speed of inferences and a low computational cost; this is due to the fact that in networks there is a very high speed. Among various machine learning algorithms, Random Forest is particularly well-suited for this task. This algorithm uses a multitude of decision trees to offer high accuracy and robustness against overfitting. Overfitting means that the model is overtrained on the training data and cannot perform well with real-world data. With the usage of Random Forest algorithms it is possible to effectively classify traffic flow features, such as packet size variance, flow duration, and inter-arrival times, to distinguish between benign and malicious traffic. Robustness is a crucial property for both the system and the algorithm. The network environment is constantly changing but, regardless, the system must keep a stable performance. The algorithm must be stable even if packets are lost.

III. Implementation

The dataset used for this implementation is *CIC-IDS2017* [1]. This dataset was chosen since it is recent, has a large number of features and various attacks types. Each line of the dataset represents a bidirectional network flow between two end-points. Each column describes a feature associated to the specific flow. Google Colab is used as platform to create the model, since it allows to have a reproducible environment, and provides the GPU power needed to train the model. The random forest algorithm implementation was taken from the *sklearn* python library, which is also used for other utility functions like *train_test_split*, *classification_report*, *confusion_matrix*, *accuracy_score*. The dataset [1] was provided in the *parquet* format and it was divided in various files. The post-processing starts by joining all the parquet files provided in the dataset, then keeping only the features that the model will have as input. Once the dataset is cleaned the next step is to perform the training. Since the implementation of the classifier is taken from the `sklearn` library, the training is ran through the *fit* function, which returns the trained model.

All of the features used are extracted at the packet level and at the flow level. The dataset originally contained 79 distinct features. However, using all available features can lead to the "Curse of Dimensionality", where the model overfits to noise and significantly increases the computational cost of training and inference. In order to select the optimal subset of features, the Random Forest Feature Importance metric has been used: it calculates how much each feature contributes to the accuracy of the model. The only influential features for the goal of the project and that have been used during the model's training are the following:

- Bwd Packet Length Mean: Average size of packets in backward direction, namely from the destination back to the source. In many DDoS attacks, like HTTP flood, the attacker sends a small request, but the victim responds with large amounts of data. A drastic shift in the average backward packet size is often a sign of an anomaly, compared to normal user browsing.
- Fwd IAT Std: Forward Inter Arrival Time Standard. Standard deviation time between two consecutive packets sent in the forward direction. Normal human traffic is bursty and irregular, which results in a high standard deviation. Automated DoS tools often send packets at fixed and rapid intervals, in a machine-like precision, bringing to a very low standard deviation. The opposite can also happen: some botnets might introduce artificial jitter to evade detection, but causing specific unnatural variance patterns.
- Fwd IAT Max: Maximum time between two packets sent in the forward direction. During a flood, attackers usually attempt to saturate the link, so they rarely pause. Therefore, the maximum time between packets becomes extremely small.
- Packet Length Variance: Variance length of a packet, meaning how much the packet's size varies from the average. Benign traffic has high variance, while malicious traffic is often uniform and brings the variance close to zero.
- Init Fwd Win Bytes: The total number of bytes sent in initial window in the forward direction. Many operating systems, such as Windows, Linux, or iOS, set default window sizes. Botnets and attack tools, on the other hand, often use custom or randomized window sizes that do not match standard OS behaviour. In SYN Flood attacks, for example, this often reveals that the user is actually a script.

In order to go on with the network capture it is suitable to use Wireshark. Wireshark is a powerful, free, and open-source network protocol analyzer (or packet sniffer), that captures and interactively browses live or offline network traffic, letting users inspect data packets to troubleshoot network issues, analyze protocols, debug applications, and so on. It is compatible with many platforms, such as Windows, macOS, Linux, etc. Wireshark is an indispensable tool for understanding, securing, and maintaining any network. In particular, it is helpful to use TShark, the CLI version of WireShark. TShark is ideal for scripting, automation, and server-based analysis, offering similar deep packet inspection and filtering capabilities as Wireshark but in a text-based format, making it great for tasks like troubleshooting, security analysis, and extracting specific data from network flows. The script provided alongside this paper, that is used to implement the IDS, does the following actions:

- After choosing a network interface, for example `eth0`, it captures 30 seconds of network traffic, then generates a pcap file, which is a file that contains network packets.

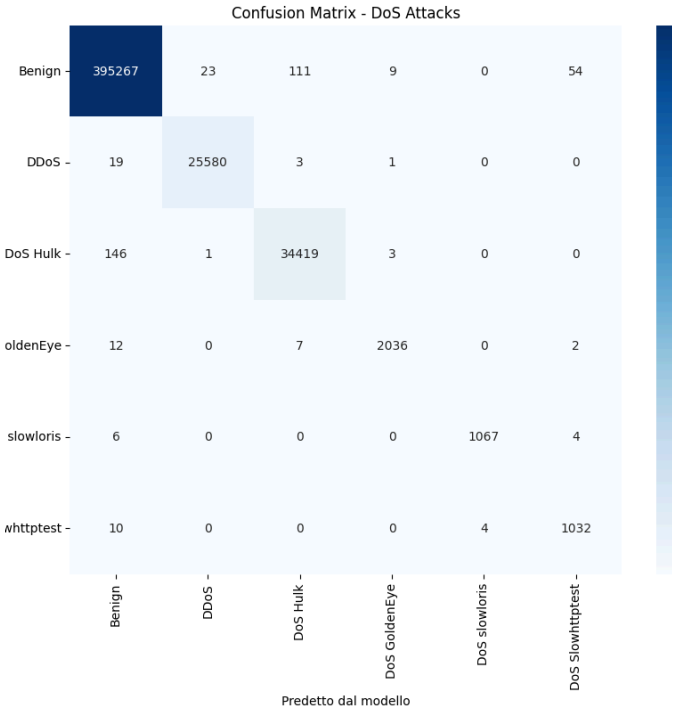
- Parses the pcap in order to extract the features needed by the model (`Bwd Packet Length Mean`, `Fwd IAT Std`, `Fwd IAT Max`, `Packet Length Variance`, `Init Fwd Win Bytes`).
- Runs the inference using the features mentioned above, then, if the number of packets classified as malevolent exceeds a threshold, the traffic is classified according to its type.

IV. Results

The model can be evaluated calculating metrics as accuracy, precision, recall and f1-score.

Label	Precision	recall	f1-score
Benign	1.00	1.00	1.00
DDoS	1.00	1.00	1.00
DoS GoldenEye	0.99	0.99	0.99
DoS Hulk	1.00	1.00	1.00
DoS Slowhttptest	0.95	0.99	0.97
DoS slowloris	1.00	0.99	0.99
-	-	-	-
accuracy	-	-	1.00
macro avg	0.99	0.99	0.99
weighted avg	1.00	1.00	1.00

To check if the model has some bias the confusion matrix must be computed. This matrix shows for each traffic flow type the model's prediction. It summarizes the number of true positives, true negatives, false positives and false negatives, providing insights into the model's accuracy and errors.



V. Conclusions

In conclusion, this project demonstrates how this pipeline of work is able to efficiently discriminate different real-time DoS attacks with an almost perfect accuracy. Features analysis and selection make up a crucial role in order to react to this kind of attacks.

References

[1] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "CIC-IDS2017." [Online]. Available: https://www.kaggle.com/dsv/4059877