AnubisFlow: A Feature Extractor for Distributed Denial of Service Attack Classification

Alan Barzilay*, **Caio L. Martinelli***,
Michele Nogueira†, **Daniel M. Batista***, **Roberto Hirata Jr.**,*
*Department of Computer Science, University of São Paulo, Brazil
† Department of Computer Science, Federal University of Minas Gerais, Brazil
Emails: alan.barzilay@usp.br, {caio.martinelli, batista, hirata}@ime.usp.br
michele@dcc.ufmg.br

2021 12th International Conference on Network of the Future (NoF) | 978-1-6654-2434-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/NoF52522.2021.9609918

***Abstract*—The detection and mitigation of DDoS attacks require a system to analyze and process the incoming network flow in a live capture manner. In this scenario, an efficient analysis depends on a good set of features to classify the traffic. With this goal in mind, we propose a technique based on a new set of features that are computationally inexpensive and descriptive of the data stream. Moreover, the technique considers the flows in many moments, not only when they are finished. We analyze its predicting performance by creating a decision tree model and a logistic regression, which achieved 99.98% and 95.99% Cohen's Kappa coefficient, respectively. In spirit with the recent trend toward reproducibility of research results, we integrate the proposal in an open-source tool called AnubisFlow. Also, our analysis for the models is available as open data to the scientific community.**

***Index Terms*—Distributed Denial of Service, Intrusion Detection System, Feature Extraction**

I. Introduction

Distributed Denial of Service (DDoS) attacks have been a menace to service providers, with the increasing number of IoT devices resulting in several new opportunities for attackers [1]. The Mirai botnet [2] is an example of such opportunities, where a botnet of vulnerable IoT devices has generated a massive DDoS attack that took down several services in the Internet [2]–[4]. In face of new and powerful attacks being ever-present, the need for efficient and agile tools and models for DDoS detection is paramount.

In a nutshell, Intrusion Detection Systems (IDSs) with support against DDoS comprise three main parts: DDoS detection, flow classification, and reaction. Once the IDS detects a possible attack, it classifies the incoming flows to distinguish which flows are legitimate and to be able to protect the networking services. Depending on the attack, a plethora of actions may be taken to protect services. In order to defend against a volumetric attack, the system may implement rate-limiting policies or request the completion of CAPTCHAs [5]. Against a low and slow attack such as Slowloris, it may limit the number of connections allowed by a single IP [6]. Against a botnet that spans the globe, it may be useful to use Content Delivery Networks (CDNs) to distribute the traffic among multiple servers [7]. Each DDoS attack is different and has its particularities that must be considered when deciding on efficient counter-measures.

Despite the well-defined structure for an IDS, the massive volume of data generated and transmitted everyday imposes a severe challenge to these systems. Some security mechanisms improve the detection of attacks by using non-traditional sources, such as posts from online social networks [8], but, in general, most of the mechanisms consider the raw, or anonymized, traffic data captured from the interconnection equipment or directly from the target. Considering the raw or anonymized traffic data, the overhead of processing and evaluating the incoming information has to be taken into account. If an IDS model that is too computationally expensive is deployed, it may hinder the service resources since it will lead to less computational power being available to endure the DDoS. It is clear that there is a trade-off between system complexity and robustness. To develop new models and tools to deal against DDoS attacks, we need to be capable of manipulating the data stream and easily extract features that are descriptive of the flow being analyzed.

In this work, we define a new set of features that are computationally inexpensive and descriptive of the data stream. We generate this set by reviewing the literature for features already in use and developing new ones. Once we have this set, we compare its descriptive power with other sets of features used in other models. For this, we use the DDoS dataset described on [9], which gathers a variety of attacks and modern techniques. We extract relevant features to classify the bidirectional flows, using the same definition of flow as the authors of [9]: a sequence of packets with identical values for (*Source IP, Destination IP, Source Port, Destination Port, and Protocol*). We call this the 5-tuple flow. Additionally, we use another kind of flow, the 2-tuple flow, which is the packets' sequence with identical values for (*Source IP, Destination IP*). The 2-tuple flow may be seen more intuitively as the combination of multiple 5-tuple flows.

Some authors of the dataset described on [9], proposed a feature extractor called CICFlowMeter [10], which gathers information on the bidirectional (5-tuple) flow. In this work, we aggregate more variables to this previous feature set,

using information from different flows for the same IP or even on individual packets. Different from [10], our proposal considers the flows in many moments, not only when they are finished. Besides, we developed a computationally inexpensive open-source tool capable of extracting these features called AnubisFlow [11].

We evaluate the quality of the features proposed by AnubisFlow in terms of the Pearson Correlation and they are better than those proposed by CICFlowMeter. We also analyze the predicting performance of our proposal by creating a decision tree model and a logistic regression, which achieved 99.98% and 95.99% Cohen's Kappa coefficient, respectively.

This paper proceeds as follows. Section II presents related works. Section III overviews the methodology we employed. Section IV details the proposal. Section V provides details about AnubisFlow, the software that implements our proposed mechanism to classify traffic with a new set of features and considering the flows in many moments, not only when they are finished. Section VI discusses the results of the performance evaluation of our proposal. Finally, Section VII concludes this work and highlights future works.

II. Related Works

In Andreoni et al. [12], the authors investigated features to detect DDoS attacks based on 2-tuple flows. These features include the number of ports and protocols present in that flow, which would be impossible when creating features restricted to the 5-tuple bidirectional flows without aggregating them onto 2-tuple flows.

Li et al. [13] developed a real-time attack detection alarm based on the entropy of bidirectional flows, considering sliding time windows to keep the entropy calculation computationally efficient. Although the alarm was efficient, it was only capable of detecting the existence of an attack. It was incapable of discerning the legitimate flows from the rest.

Many DDoS detection features are based on the ideas of abrupt traffic change, flow asymmetry, distributed source IP addresses, and concentrated target IP addresses [15]. Although useful for detecting the attack, they suffer from the inability to distinguish and classify flows.

Jieren et al. [15] designed an algorithm to detect DDoS attacks based on IP address features value (IAFV). Cheng et al. [16] utilized the periodicity of TCP flow by spectrum analysis to propose an attack detection method. Both [17] and [18] suggested that the number of SYN packets and the number of FIN/RST packets have the same ratio under normal conditions and could be used to detect SYN flooding. Abdelsayed et al. [19] proposed a detection method based on the unbalance of network traffic. In general, these previous methods are outdated and easily countered by attackers. Attackers may use spoofed source IP addresses or even balance the number of SYN packets and FIN/RST packets by simulating normal flows and sending out attacks through many sources without hindering its performance. Also, the rate of inflow and outflow is not always in balance, such as in audio/video flows.

In Table I, we provide a summary of the works here presented and the central ideas behind them for quick reference.

TABLE I
Related work and their key concepts and ideas

Andreoni et. al. [12]	Features from 2-tuple flows
Li et. al. [13]	Entropy of bidirectional flows
Jieren et. al. [15]	IP address features value
Cheng et. al. [16]	Periodicity of TCP flow
Wang et. al. [17]	FIN/RST packets ratio
Lee et. al. [18]	FIN/RST packets ratio
Abdelsayed et. al. [19]	Network traffic unbalance/asymmetry

III. Methodology

We divide this work into three steps. Firstly we design a mechanism that receives packets and stores relevant information of the flow they are part of and generates features of the flows at any given time.

Secondly, we generate features and compare them to those generated using the CICFlowMeter, a feature extractor provided in [10]. Since CICFlowMeter generates features of finished flows, we do the same. We use Spearman's correlation with the target to make this comparison.

Finally, we want to be able to classify the flows using our features. We want to distinguish the regular flows from the DDoS attack ones, but we don't want to do it only with finished flows, so we generate our features in many moments of the flow. We train a classification model on the first day of the dataset and test it on the second day.

These three steps were implemented by us in the AnubisFlow, an open-source software that generates features to identify DDoS flows.

A. Dataset

When proposing mechanisms to data classification, the selection of the dataset that will be used in the training, validation, and testing is crucial.

The publication where the UNB CIC DDoS 2019 dataset is presented [9] describes the creation of the dataset that we used and the use of supervised machine learning algorithms to classify bidirectional flows, distinguishing the regular traffic from the attack traffic. In the article, it is described the testbed architecture, which consists of two separated networks, employing commonly used equipment in the victim network, such as router, firewall, switch and different versions of operating systems, the benign profile agent, which is obtained by extracting the abstract behavior of 25 users, and the 11 different DDoS attack profiles (UDP-lag, UDP Flood, SYN Flood, TFTP, NTP, CharGen, PORTMAP, SNMP, NETBIOS, LDAP, DNS, SSDP, and MSSQL). The dataset is composed of pcap files of the simulated attacks and the tabular data representing the set of features extracted with the CICFlowMeter [10], a tool initially developed to classify different types of Tor traffic. With these features, a Random Forest model calculates the feature importance for different kinds of attack traffic present in the dataset, besides regular traffic.

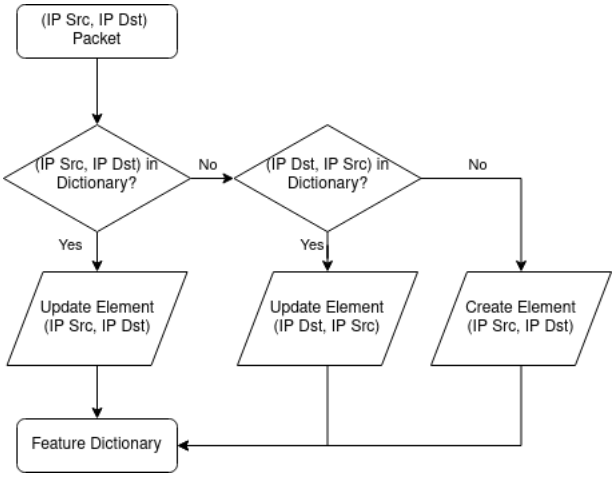

Fig. 1. Update procedure of the 2-tuple dictionary.

IV. Proposal

Our proposal is able to generate features for the flows at any given time. It supports two dictionaries, one with information on the 2-tuple flows (*Source IP, Destination IP*) and one with information on the 5-tuple flows (*Source IP, Destination IP, Source Port, Destination Port, and Protocol*). It can keep only one of these dictionaries, defined as a parameter. Another parameter is whether to consider unidirectional or bidirectional flows.

The update procedure of the bidirectional 2-tuple dictionary is illustrated in Fig. 1, and that of the bidirectional 5-tuple occurs similarly. It receives a packet in the Scapy[1] [20] format, checks if the flow, or the reverse flow, from each incoming packet, is already present in the dictionary, and then updates the information or create a new entry if they are not present. The unidirectional procedure doesn't have the reverse flow check step.

Our technique generates a set of 21 features for each 2-tuple unidirectional flow and a set of 41 features for the bidirectional case. The bidirectional flow features are the same as those in the unidirectional case but calculated twice, one for each direction of the flow. The only feature not duplicated is the time duration, since it is calculated on the totality of the flow and thus is the same in both directions. Analogously, the set of features for the unidirectional 5-tuple flow is 16 features long, and the bidirectional case is 31 features long, with again the time duration of the flow being the only feature not duplicated since it is the same in both directions. Tables II and III contain a description of all features generated by our technique for the unidirectional 2-tuple and 5-tuple flows, respectively.

A. Feature Comparison

To compare the features generated by our technique with the ones of the CICFlowMeter, we must generate them at the same stage of the flow, i.e., after completing the flow, because of CICFlowMeter's limitations. For this, we use the UNB CIC DDoS 2019 dataset. We calculate the timestamp of the last packet of the flow with the duration of the flow added to its timestamp. With the last timestamp of each flow, we create a queue of flows to be generated.

Then we start reading the pcap files also provided in [9], and when the last timestamp is reached, we generate the features of the flow and get the next element in the queue. The procedure is detailed in Algorithm 1.

This step has two goals. The first one is to validate our mechanism by checking if it generates features consistent with the ones given by the CICFlowMeter. The second goal is to compare its performance with CICFlowMeter. We expect that our mechanism, designed to extract features from DDoS attacks, has more information to detect attacks than the CICFlowMeter, designed initially to classify Tor traffic.

[1]Scapy is a network packet manipulation program (https://scapy.net/. Accesed: 2021-05-16.)

TABLE II
Feature set generated by the AnubisFlow for the unidirectional 2-tuple flow.

Feature	Description
qt_pkt	Amount of packets
qt_pkt_tcp	Amount of TCP Packets
qt_pkt_udp	Amount of UDP Packets
qt_pkt_icmp	Amount of ICMP Packets
qt_pkt_ip	Amount of IP Packets
qt_prtcl	Amount of protocols
qt_src_prt	Amount of Source Ports
qt_dst_prt	Amount of Destination Ports
qt_fin_fl	Amount of FIN Flags
qt_syn_fl	Amount of SYN Flags
qt_psh_fl	Amount of PSH Flags
qt_ack_fl	Amount of ACK Flags
qt_urg_fl	Amount of URG Flags
qt_rst_fl	Amount of RST Flags
qt_ece_fl	Amount of ECE Flags
qt_cwr_fl	Amount of CWR Flags
avg_hdr_len	Average Header Size
avg_pkt_len	Average Packet Size
frq_pkt	Frequency of packets
avg_ttl	Average TTL
tm_dur_s	Time duration of the flow

TABLE III
Feature set generated by the AnubisFlow for the unidirectional 5-tuple flow.

Feature	Description
qt_pkt	Amount of packets
qt_fin_fl	Amount of FIN Flags
qt_syn_fl	Amount of SYN Flags
qt_psh_fl	Amount of PSH Flags
qt_ack_fl	Amount of ACK Flags
qt_urg_fl	Amount of URG Flags
qt_rst_fl	Amount of RST Flags
qt_ece_fl	Amount of ECE Flags
qt_cwr_fl	Amount of CWR Flags
avg_hdr_len	Average Header Size
avg_pkt_len	Average Packet Size
max_pkt_len	Length of the longest packet
min_pkt_len	Length of the shortest packet
frq_pkt	Frequency of packets
avg_ttl	Average TTL
tm_dur_s	Time duration of the flow

Algorithm 1 Feature Comparison Data

```
procedure FEATUREDATA(CICFlowMeterData, pcapFiles)
    sort CICFlowMeterData by last_timestamp
    idx ← 0
    id ← CICFlowMeterData[idx][id]
    t ← CICFlowMeterData[idx][last_timestamp]
    for packet in pcapFiles do
        this_time ← packet.time
        update_anubis(packet)
        if this_time ≥ t then
            write_data(id) ▷ Generates features for id and writes to a csv file.
            idx ← idx + 1
            id ← CICFlowMeterData[idx][id]
            t ← CICFlowMeterData[idx][last_timestamp]
```

Algorithm 2 Classification Data

```
procedure CLASSIFICATIONDATA(pcapFiles)
    for packet in pcapFiles do
        update_anubis(packet)
        id ← packet.id
        n ← get_npackets(id)    ▷ Get number of packets in the flow id.
        if n%100 = 0 then
            write_data(id) ▷ Generates features for id and writes to a csv file.
```

B. Classification Models

As it may not be interesting to wait for the end of a flow to classify it as a DDoS attack, we use our mechanism to generate two tables, one to train and another to test our classification models, also considering unfinished flows. We use the first day of the dataset to train and the second to test, as done in [9].

With this reasoning in mind, instead of waiting for the last timestamp to generate the features of a flow, we generate them every hundredth packet, as exemplified in Algorithm 2.

By analyzing the data, we learned that all the attacks come from only one IP address. To give more variance to our data, we erase the information of a flow as soon as it reaches fifty thousand packets. After this, if there is another packet of that flow, it will start a new entry in our dictionary, as if it was part of a new flow.

We also learned that the amount of regular traffic in the data is too small, so we train only simple models, a Decision Tree, and a Logistic Regression. For both of them, we use 10-fold cross-validation to select the best parameters of the model. We use the Scikit-learn python library [21] for this. We explore in a grid search the following parameters of the Decision Tree:

- `criterion`: whether to choose features and the threshold of the split based on entropy or Gini
- `max_depth`: the maximum depth the tree can reach
- `min_samples_leaf`: the minimum amount of observations that must be on each leaf on the training data

For the Logistic Regression, we search into the parameters:

- `penalty`: what kind of regularization to use for the features
- `C`: the inverse of the regularization strength

V. AnubisFlow

The proposal presented in the previous section is implemented in the software AnubisFlow [11]. It is written in Python.

AnubisFlow is a tool designed to generate features to identify DDoS flows from `.pcap` files. It works by creating a class that stores information about the flows. This class is continuously updated whenever a new packet passes through the network interface. The class can store two dictionaries. One for the 2-tuple flow and one for the 5-tuple flow. The features of the flows helps identifying and blocking flows originated from a DDoS attack.

The performance evaluation presented in the next section compares our proposal, implemented in AnubisFlow, with the proposal implemented in the CICFlowMeter tool [10].

VI. Experimental Results and Discussion

The performance evaluation comparing AnubisFlow and the CICFlowMeter was carried out with the UNB CIC DDoS 2019 dataset. The next subsections present the results obtained in the classification of two reflective DDoS attacks present in the dataset: UDP and NTP.

A. Feature Comparison

After generating the features of the finished flows with AnubisFlow, we compared them with the ones generated with the CICFlowMeter. Fig. 2 and Fig. 4 plot the largest Pearson Correlations of the labels (being an attack or not) with each feature in these sets considering the NTP attack present in the dataset.

A higher absolute value of the Pearson Correlation with the target indicates the predictive power of a feature. In essence, the higher the absolute value of the Pearson Correlation, the better. As we can see, AnubisFlow features (Fig. 2) are highly correlated with our target, even more so than the features from CICFlowMeter are (Fig 4). This is a behavior that is consistent among all attacks found in the dataset. Another point to notice is that almost all the features are from the 2-tuple dictionary. It makes sense since an attacker can send packets from and to many ports.

Also, a FIN packet is understood by the CICFlowMeter as the end of a TCP flow. Although it marks the end of a regular traffic flow, it may not be useful in a DDoS attack. AnubisFlow keeps storing information of a flow even though the attacker or the attacked sent a FIN packet, which can be partially responsible for a higher absolute value for the Pearson Correlation.

An important caveat we noticed is a discrepancy between AnubisFlow features and the ones generated by CICFlowMeter. Some of the features for the 5-tuple flow we considered have an overlap with the ones implemented by the

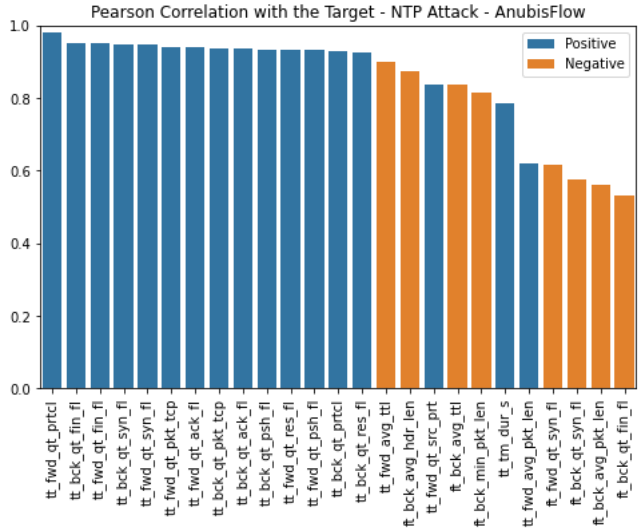

Fig. 2. Pearson Correlation with the target for each feature generated by AnubisFlow for the 2-tuple (tt) and 5-tuple (ft) bidirectional flows, both in the forward (fwd) and backward (bck) directions (NTP attack).

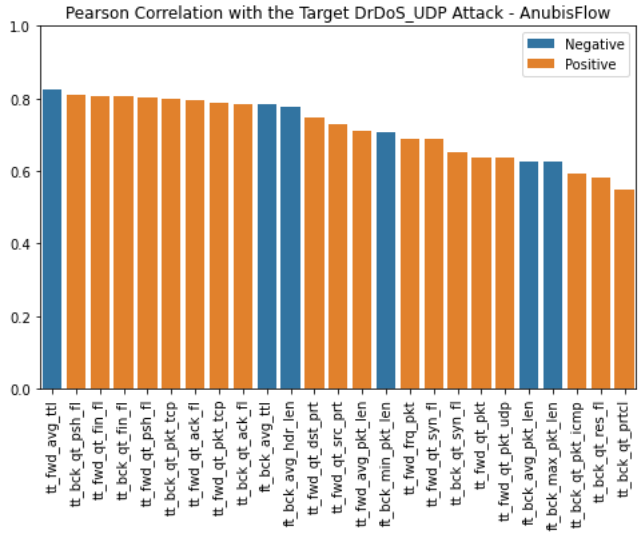

Fig. 3. Pearson Correlation with the target for each feature generated by AnubisFlow for the 2-tuple (tt) and 5-tuple (ft) bidirectional flows, both in the forward (fwd) and backward (bck) directions (UDP attack).

CICFlowMeter, but when we compared our outputs, we noted some discrepancies. For example, our count of SYN flags was different from theirs for specific flows. After a validation step where we inspected the pcap files, we concluded that there must have been an error on the CICFlowMeter tool or on how it was applied to this dataset. Further investigation may be needed to clarify this matter.

Fig. 3 and Fig. 5 present the Pearson Correlation considering the UDP attack present in the dataset. Despite the values are not the same that were obtained when considering the NTP attack, the conclusions are the same: AnubisFlow features are more correlated than those from CICFlowMeter.

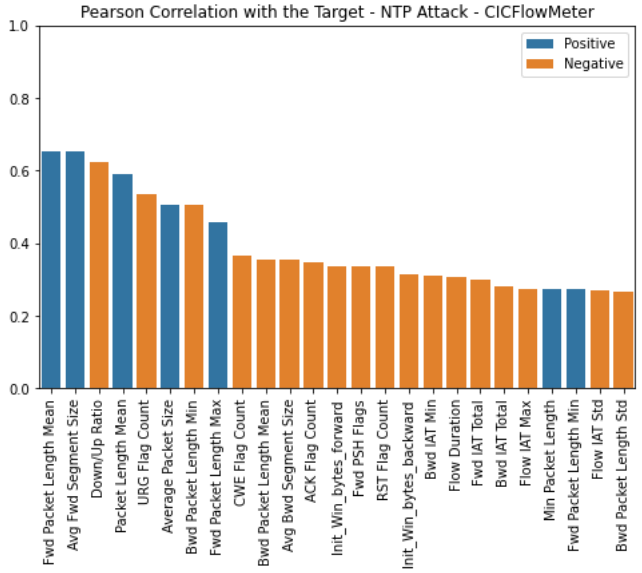

Fig. 4. Pearson Correlation with the target for each feature generated by CICFlowMeter, features labeled as described in its documentation (NTP attack).

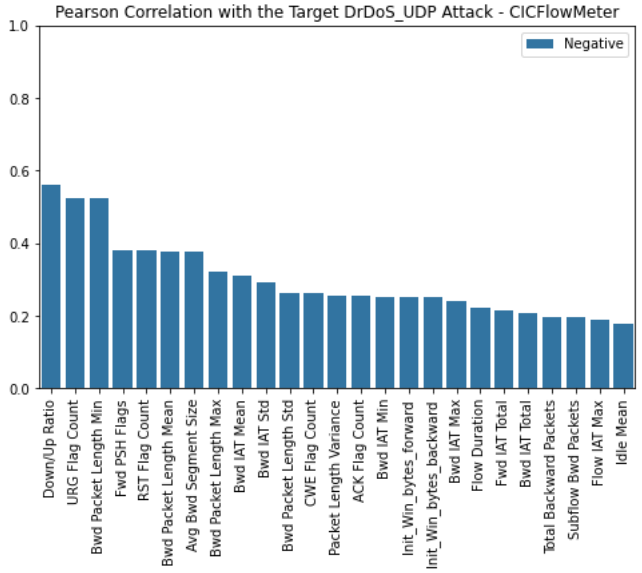

Fig. 5. Pearson Correlation with the target for each feature generated by CICFlowMeter, features labeled as described in its documentation (UDP attack).

With these results, we understand that AnubisFlow features have significant predictive power to detect DDoS attacks and that the 2-tuple flows have more predictive power on the DDoS attack classification.

B. Classification Models

As we previously mentioned, we modeled the attacks using partial flows and erasing flows from the dictionary after reaching fifty thousand packets. Based on the results of the feature comparison part, we chose to generate our data using

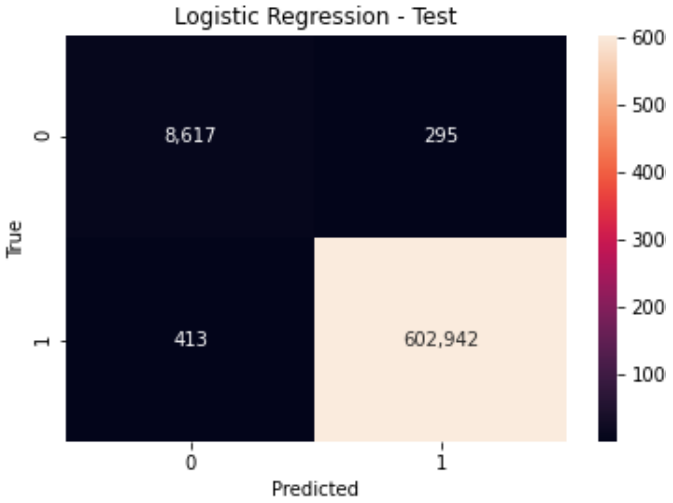

Fig. 6. Confusion Matrix for the Logistic Regression.

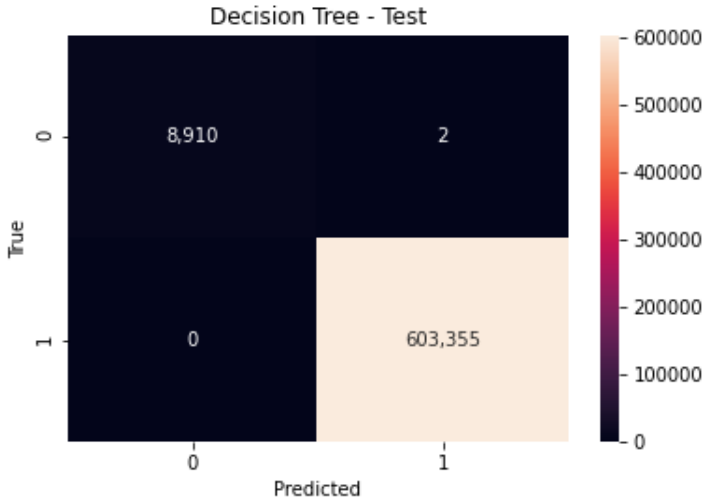

Fig. 7. Confusion Matrix for the Decision Tree.

only the 2-tuple unidirectional flows. This had two goals, first is to make it more computationally efficient with low information loss, as there are many more unique 5-tuple keys, and the 2-tuple features have more significant correlations with the target. The second goal is to generate more variance in the data. Instead of only one bidirectional attack flow, now we have two unidirectional attack flows - the response from the local IP to the attacker is also considered malicious.

The results obtained using a Logistic Regression model and using a Decision Tree model were quite similar, with excellent performance of both, but better for the Decision Tree. Since the data is imbalanced, to evaluate the performance we used Cohen's Kappa coefficient [22], which is robust to the imbalance of the classes. After the grid search on the models parameters, we obtained a value of 95.99% for the Logistic Regression and 99.98% for the Decision Tree, both on the test data.

The confusion matrix of the Logistic Regression can be seen on Fig. 6 and the one of the Decision Tree on Fig. 7. As we can see, our models have a near-perfect performance.

In Fig. 8, we can see the representation of the Decision Tree model. We understand that the splits are consistent with our intuition of DDoS attacks. For example, the first split regards the frequency of packets of the flow, and high packet frequency is expected in a DDoS attack.

We conclude that the data of the regular benign traffic has an entirely different feature distribution to the attack traffic. This makes the problem easy to solve with our features.

C. Packet Filter Policy

Originally we intended to develop a policy to filter the network traffic, such as ignore all the packets sent from/to some specific IP given a number of partial (or completed) flows that were classified as attacks by our models. This policy could be applied on the network by a firewall such as Linux's `iptables`. To analyze the efficiency of this DDoS countermeasure, we could calculate metrics such as the precision, the True Positive Rate (TPR), and the False Positive Rate (FPR) of these filters, simulating how much of the attack traffic could be filtered out and how much of the benign traffic we would be impacting.

After fitting our models and testing it in the test day, we could realize that the two false-positive observations of the Decision Tree were the first observations of their respective flows, that is, flows with only one hundred packets. A simple policy that could be applied with no impact in the benign traffic would be: ban the external IP as soon as it has two flows classified as an attack by our model. That would make our two false-positive observations not to impact our policy, and our service would receive and respond maximally to two hundred attack packets before the attacker was banned.

In a more complex situation, we could model our policy as a Hidden Markov Model, where each observation (flow information after the total packets is a multiple of one hundred in our case) could be classified and possibly change the situation of the IP (banned or not). Any IP that is not banned but has one observation classified as attack could change its state, and a banned IP also could change its state over time.

VII. Conclusion and Future Work

This paper defines a rich feature set for DDoS detection and proposes a tool for extracting these features in a manner that is cheap in computational resources as well as being specifically designed to detect DDoS attacks.

The near to perfect classification of our models and the fact that there is only one attack IP raise a flag on how realistic is the dataset we used. In future work, we want to apply our proposal to more DDoS datasets. Also, we intend to investigate further the computational resources required to keep this system we built working. Also, we intend to

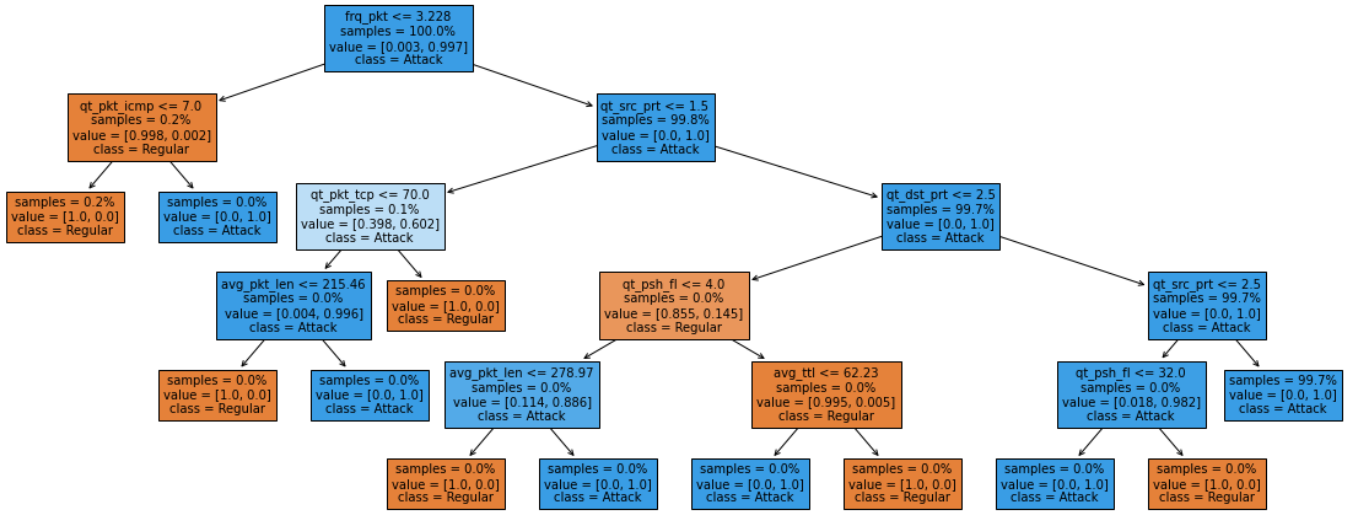

Fig. 8. The representation of our final model. Left exiting arrows mean the condition in the box is met, and right exiting arrows not met.

explore more mechanisms of counter-measures to take against malicious packets and flows.

VIII. Acknowledgments

This research is part of the INCT of the Future Internet for Smart Cities funded by CNPq proc. 465446/2014-0, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001, FAPESP proc. 14/50937-1, and FAPESP proc. 15/24485-9. It is also part of the FAPESP proc. 18/22979-2 and FAPESP proc. 18/23098-0.

References

[1] B. M. Rahal, A. Santos, and M. Nogueira, "A Distributed Architecture for DDoS Prediction and Bot Detection," *IEEE Access*, vol. 8, pp. 159 756–159 772, 2020.

[2] C. Kolias, G. Kambourakis, A. Stavrou and J. Voas, "DDoS in the IoT: Mirai and Other Botnets," in Computer, vol. 50, no. 7, pp. 80-84, 2017, doi: 10.1109/MC.2017.201.

[3] Scott Hilton , "Dyn Analysis Summary Of Friday October 21 Attack", https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/, Accessed: 2020-04-10

[4] Nicky Woolf, "DDoS Attack that Disrupted Internet was Largest of its Kind in History, Experts Say", https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet, Accessed: 2020-04-10

[5] K. J. Singh and T. De, "DDoS Attack Detection and Mitigation Technique Based on HTTP Count and Verification Using CAPTCHA," in *2015 International Conference on Computational Intelligence and Networks*. IEEE, 2015, pp. 196–197.

[6] Shorey, T., Subbaiah, D., Goyal, A., Sakxena, A., & Mishra, A. K. (2018, September). "Performance Comparison and Analysis of Slowloris, Goldeneye and Xerxes DDoS Attack Tools." In 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI) (pp. 318-322). IEEE.

[7] K. Kim, Y. You, M. Park, and K. Lee, "DDoS mitigation: Decentralized CDN using Private Blockchain," in *2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN)*. IEEE, 2018, pp. 693–696

[8] R. Campiolo, L. A. F. Santos, D. M. Batista, and M. A. Gerosa, "Evaluating the Utilization of Twitter Messages as a Source of Security Alerts," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, 2013, p. 942–943.

[9] I. Sharafaldin, A. H. Lashkari, S. Hakak and A. A. Ghorbani, "Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy", 2019 International Carnahan Conference on Security Technology (ICCST), 2019, pp. 1-8.

[10] A. Habibi Lashkari, G. Draper Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of Tor Traffic using Time based Features," in *Proceedings of the 3rd International Conference on Information Systems Security and Privacy - ICISSP*, 2017, pp. 253–262.

[11] A. Barzilay, C. L. Martinelli, "Anubis: A Tool for Feature Extraction and DDoS Attack Classification." https://github.com/caiolmart/anubisflow, Accessed: 2020-04-10.

[12] M. Andreoni Lopez, D. M. F. Mattos, O. C. M. B. Duarte, and G. Pujolle, "Toward a Monitoring and Threat Detection System based on Stream Processing as a Virtual Network Function for Big Data," *Concurrency and Computation: Practice and Experience*, vol. 31, no. 20, p. e5344, 2019.

[13] J. Li, M. Liu, Z. Xue, X. Fan and X. He, "RTVD: A Real-Time Volumetric Detection Scheme for DDoS in the Internet of Things," in IEEE Access, vol. 8, pp. 36191-36201, 2020.

[14] M. Gharaibeh, and C. Papadopoulos. "DARPA-2009 Intrusion Detection Dataset Report." Tech. Rep. (2014).

[15] J. Cheng, J. Yin, Y. Liu, Z. Cai and C. Wu, "DDoS Attack Detection Using IP Address Feature Interaction," 2009 International Conference on Intelligent Networking and Collaborative Systems, 2009, pp. 113-118.

[16] C. Cheng, H. T. Kung, K. Tan, "Use of Spectral Analysis in Defense against DoS Attacks," Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE, 2002, pp. 2143-2148 vol.3.

[17] H. Wang, D. Zhang and K. G. Shin, "Detecting SYN Flooding Attacks," Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, 2002, pp. 1530-1539.

[18] K. Lee, J. Kim, K. Kwon, Y. Han, S. Kim, "DDoS Attack Detection

Method using Cluster Analysis. Expert Systems with Applications." 2008 34. 1659-1665.
[19] Abdelsayed, S., Glimsholt, D., Leckie, C., et al.: An Efficient Filter for Denial-of Service Bandwidth Attacks. In: Proceedings of the 46th IEEE GLOBECOM, pp. 1353–1357 (2003)
[20] R. Rohith, M. Moharir, G. Shobha, "SCAPY-A Powerful Interactive Packet Manipulation Program." 2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS). IEEE, 2018.
[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, "Scikit-learn: Machine Learning in Python" in Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.
[22] J. Cohen, "A coefficient of Agreement for Nominal Scales." 1960 Educational and Psychological Measurement. 20 (1): 37–46.