# Revisiting *FlowGuard*: A Critical Examination of the Edge-Based IoT DDoS Defense Mechanism

Nikola Gavric[1], Guru Bhandari[1], and Andrii Shalaginov[1]

Kristiania University College, Oslo, Norway
`{nikola.gavric, guru.bhandari, andrii.shalaginov}@kristiania.no`

**Abstract.** Efficient detection and mitigation of Distributed Denial of Service (DDoS) attacks targeting Internet of Things (IoT) infrastructure is a challenging task in the field of cybersecurity. Y. Jia et al. propose Flowguard, an extraordinary solution to the mentioned problem that relies on inspecting network flow statistics leveraging statistical models and Machine Learning (ML) algorithms. Flowguard utilizes CICDDoS2019 dataset and the authors' unique dataset. The authors did not provide the source code or the complete dataset, yet, motivated by their findings, we decided to reproduce Flowguard. However, we ran into numerous theoretical and practical challenges. In this paper, we present all of the issues related to Flowguard's foundations and practical implementation. We highlight the false and missing premises as well as methodological flaws, and lastly, we attempt to reproduce the flow classification performance. We dismantle Flowguard and show that it is unrelated to IoT due to the absence of IoT devices and communication protocols in the testbeds used for generating their and CICDDoS2019 datasets. Moreover, Flowguard applies nonsensical statistical models, and uses an overfitted ML model that is inapplicable in real-world scenarios. Furthermore, our findings indicate that Flowguard's binary ML classification results were manipulated. They were presented in a misleading manner and improperly compared against another paper's multi-class classification results without a reference. Our results show that Flowguard did not solve the problem of DDoS detection and mitigation in IoT.

**Keywords:** Flowguard · IoT Security · DDoS Attacks.

## 1 Flowguard

Flowguard [4] is an edge-based Network Intrusion Detection System (NIDS) designed to protect Internet of Things (IoT) devices from Distributed Denial of Service (DDoS) attacks. Flowguard examines features of both ingress and egress network packet flows to determine whether there is an attack. Flowguard is based on the flow records from the CICDDoS2019 [8] dataset enhanced by the authors' dataset containing low-rate (Slow Request/Response) attacks and additional DDoS attack traffic generated by the Bonesi botnet simulator [3]. Flowguard consists of four modules designed for different purposes: attack detection, identification, classification, and mitigation. To achieve this, Flowguard relies on flow statistics, Long Short-Term Memory (LSTM), and Machine Learning (ML), more precisely, Convolutional Neural Networks (CNN). These four steps occur in cascade and provide malicious flow identification accuracy of over 99%, while the attack type classification is reportedly 100%. The results indicate that Flowguard is an efficient solution against DDoS attacks in terms of detection rate and computational demand. The authors have not provided any open-source code or a replication package with the paper. However, while attempting to reproduce the results considering the same set of configuration parameters, we ran into many theoretical and practical issues, leading us to conclude that the published results were manipulated. In this paper, we dismantle Flowguard into its principal components as described by the authors, and we address the following issues:

1. False and missing premises.
2. Flawed and nonsensical methodology for DDoS detection.
3. Validity of the results.

## 2 False and Missing premises

In this section, we describe the missing and false premises of the paper discussed. We primarily question Flowguard's foundations and feasibility. In the current section, we clarify terms and practical limitations before dismantling Flowguard's inner mechanisms.

### 2.1 Definition of a Flow

A network packet flow typically refers to a set of packets exchanged between two points in a network. However, there are practical considerations that are vital in implementing software that were not mentioned in the text. Despite not clearly defining a flow, we can assume what the authors meant by analyzing the features obtained by the CICFlowMeter tool [5] that they used. The authors do not specify the necessary conditions for defining a flow with respect to temporal parameters. For example, if there is a brief data exchange between points A and B once per hour, should this be interpreted as 24 flows per day or merely a single flow? Since Flowguard relies on flow statistics, defining the terminating condition for a flow is of fundamental importance because otherwise, every initiated connection could turn into a lengthy and potentially infinitely long flow.

### 2.2 DDoS Attack Types

The authors state that there are many ways to classify DDoS attack types, but they choose to focus only on IoT DDoS attacks. However, they fail to provide a reference or a reason why Flooding and Slow Request/Response are IoT DDoS attacks, whereas other types are supposedly not applicable to IoT.

Flooding attacks are said to be based on massive disguised network packets, but there are no references to the claim that flooding attacks necessarily consist of massive packets. For instance, flooding attacks consisting of low-payload packets can be just as harmful to IoT devices [10]. Furthermore, it is unclear how these attacks are or can be disguised, as flooding attacks are easily distinguishable by their bandwidth consumption. The authors proceed to describe subtypes of Naive Flooding attacks and claim that naive flooding attacks, as a consequence, fill an unspecified buffer on the victim's end. After that buffer is full, the victim can no longer serve legitimate users. However, Flooding attacks aim to overwhelm the victim's network infrastructure rather than fill a buffer. Once the resources are exhausted, the victim may become unresponsive to legitimate requests due to the high volume of malicious traffic [12].

### 2.3 Description of the Slowhttp DoS Attack

The authors claim that slow request/response attackers hold the communication channel and exhaust the victim's resources by spoofing high-workload requests or responses. However, slow DoS attacks typically do not consume a lot of resources in terms of CPU or memory usage on the victim server [11]. Instead, they aim to exhaust available network connections by tying them up with slow or incomplete requests. The authors also claim that the attacker segments the legitimate HTTP packets into tiny fragments and sends them as slowly as possible within the maximum allowed communication time. However, Slowhttptest sends incomplete requests rather than fragmented ones [9]. Furthermore, applications typically cannot create fragmented packets, as the network layer handles fragmentation.

### 2.4 Convolutional Neural Networks

The authors state they choose CNNs for flow inspection because of their ability to deal with classification in unrelated domains. Their explanation is based on a false equivalency between network flow classification and tasks such as image, audio, and text classification. The cited papers demonstrate the effectiveness of CNNs in the domains unrelated to network flow data. Consequently, the reasoning behind the application of CNNs to network flow classification is misleading, as the cited references do not support the appropriateness of CNNs for this type of data.

### 2.5 Practical Implementation

Flowguard is a proof-of-concept software that utilizes a large dataset of flows to generate results. It is unknown which programming language was used to write Flowguard. Nevertheless, the authors emphasize low delay in detecting malicious flows, based solely on ML inference time, without considering the time required for the program to obtain the packets, parse them, and form flow statistics. Moreover, they fail to address the time required for a flow to appear malicious compared to the training data, considering that some of the flows in the CICDDoS2019 describe more than 10 minutes of network traffic. This is particularly an issue with the initial scanning of each new flow. Additionally, since the entire dataset consists of flow statistics, benign flows can turn malicious and launch attacks undetected if they have a sufficiently long benign history, as drastically altering the statistics may require plenty of time.

## 3 Methodological Issues

The inner mechanisms of the Flowguard are complex and overcomplicated without justification. The cascade order of actions performed on flows involves statistical analysis, followed by applying an LSTM model to identify malicious flows (binary classification) and CNN to classify the attacks to choose one of the two mitigation strategies and update the malicious flow statistics database. The mitigation strategy involves dropping the session in case of slow request/response attacks and dropping a flow in case of flooding attacks. It is not stated why sessions should not be dropped in case of flooding attacks. The authors state that Flowguard should be upgradable to handle zero-day attacks. However, the remainder of the paper neither mentions zero-day attacks nor addresses the upgradability or any other enhancements of Flowguard.

### 3.1 Dataset

Before getting into the implementation details, we have to discuss the training datasets. The authors opted for CICDDoS2019 to train a defense mechanism against what they called IoT attacks. However, the mentioned dataset is unrelated to IoT, as its victim devices are Windows operating system (OS) PCs and a Ubuntu OS server hosting a website. The computers are locally connected via an Ethernet link. Additionally, the benign dataset in CICDDoS2019 is not IoT-specific either, as the background traffic was generated by mimicking human interactions across protocols like HTTP, HTTPS, FTP, SSH, and email, which contrasts with the typical device-to-device communication present in IoT environments [7].

The authors decided to enrich their training dataset by using a DDoS botnet simulator and Slowhttptest to generate additional data at the rate of 10,000 packets per second. To achieve this, they set up a testbed consisting of virtual machines on a single Windows OS PC. The constructed dataset consists of roughly a million packets per attack type. An eye-catching detail is that for flooding attacks, there are also roughly a million flows per attack type. Therefore, most flood attack flows consist of just a single packet, thus beating the purpose of flow-based detection. Regarding the slow request/response attacks, a rate of 10,000 packets per second does not seem very slow, hence defying the purpose of a low-rate attack by turning it into a high-rate one. Moreover, the authors did not explain how they achieved a fixed attack rate with an attack that periodically sends packets and does not offer an option to set the rate manually [1].

In summary, the training dataset intended to defend against IoT DDoS attacks was created without using a single IoT device or machine-to-machine communication. Instead, it was generated using computers mimicking human behavior that were connected via Ethernet or virtual machines on a local network, whereas IoT environments typically use wireless communications. Therefore, we conclude that the dataset used for training Flowguard is unfit for the IoT-related tasks.

### 3.2 Statistical Analysis

The initial step in dealing with the flows involves statistical analysis, or flow filtration, as the authors named it. The process consists of checking each flow's ten feature statistics against the average value obtained through training, and if they closely match (within 1% deviation), then it is safe to say that the new flow is malicious and mitigation takes place. The authors mention four feature sets for the four types of DDoS attacks they are defending against (three types of flood and slow attacks), so we can assume that each new flow is compared against four feature sets. This process only occurs once for each new flow, so it does not make sense to compare its features against an average set of values of the entire training dataset in which some entries describe lengthy flows. Furthermore, it is hard to determine whether even the exact entries from the training dataset would fit into the set 1% deviation of the entire dataset's average values for the ten chosen features. It is unclear why the threshold is set to exactly 1% and why they use exactly ten features. The authors did not present a performance evaluation of this step; hence, its success rate in detecting DDoS attacks is unknown. All of the flows that pass this step proceed to the next module in the cascade.

The DDoS attack detection module is supposed to determine suspicious network behavior and pass flows for further analysis if deemed necessary. The authors boldly assume that IoT traffic is typically static, where sensors periodically send data. This assumption excludes many IoT applications and devices despite Flowguard being initially described as a solution that generally works in IoT. For instance, audio and video streaming services are entirely omitted, even from a theoretical perspective.

The authors propose using yet another statistical method for detecting DDoS attacks, but this time over aggregated traffic instead of individual flows. The method to detect DDoS attacks is now described using a time series, as shown in (1), to obtain $D(t)$, shown in (2), whose value is used to make decisions. If $D(t)$ is higher than a set threshold, all flows will be forwarded for further inspection. Otherwise, only a tiny portion of flows will be sent for inspection, but only after a certain period of $D(t)$ being lower than the threshold. There are no instructions on how to obtain the threshold value. We assume that if $D(t)$ is above the threshold, the flows will be instantly sent (event-driven algorithm) for further inspection, whereas in other cases, the system is time-based. The assumption is that if $D(t)$ is high, many flows are malicious, while if it is low, only some may be malicious.

This assumption is unjustified, as in the described static environment, just a single high-bandwidth flow is sufficient to cause anomalous traffic to spike. On the contrary, if $D(t)$ is small, and an attack is happening, it may go undetected indefinitely.

$$T(t) = P(t) + V(t) + A(t) \tag{1}$$

Traffic at time $t$ is $T(t)$, while $P(t)$ is the stable traffic rate, $V(t)$ represents the variance, and $A(t)$ accounts for the anomaly due to unknown sources. The authors claim it is hard to obtain $P(t)$ and $V(t)$, so they introduce their estimates $P'(t)$ and $V'(t)$ without explaining the techniques used to estimate the newly introduced variables. After removing the unnecessary brackets from the original paper's equation, we obtain the following:

$$D(t) = \frac{T(t) - P'(t)}{V'(t)} \tag{2}$$

$D(t)$ is a variable that the authors use to compare against a fixed threshold to determine whether there is an ongoing DDoS attack. $T(t)$ is easily obtainable, but dismantling it into the three mentioned components is challenging. However, to clarify what the authors meant, we put equation (1) into (2) to get the following equation:

$$D(t) = \frac{V(t) + A(t) + P(t) - P'(t)}{V'(t)} \tag{3}$$

If we assume that $P'(t)$ and $V'(t)$ are accurate estimates of $P(t)$ and $V(t)$, we can use them interchangeably and neglect the minor differences to simplify the explanation of the equation.

$$D(t) = 1 + \frac{A(t)}{V'(t)} \tag{4}$$

In other words, the authors suggest that a spike in anomalous traffic or a drop in variance are signs of malicious traffic. While high bandwidth intuitively indicates there could be a flooding attack, there is no reference or explanation as to why low variance would be an indicator of malicious traffic. For instance, in [6], the authors present a large standard deviation, and thus variance increases during the flooding DDoS attacks. It is unclear why the authors chose the ratio of anomalous traffic to variance as a metric because no supporting literature or explanation has been provided. It is also unclear how the differences between the actual and estimated parameters play a role in DDoS detection, such as $P(t) - P'(t)$ and $V(t)/V'(t)$. Additionally, it undefined what happens in case $V'(t)$ is zero.

### 3.3 Machine Learning

If the statistical methods identify potentially malicious flows, the flow handling module will perform an ML-based analysis on each flow to determine it. The first step in this cascade is to perform a binary classification using LSTM. If the flow is deemed malicious, it will be sent to the classification module; otherwise, it will be considered benign. The choice of LSTM is justified by its ability to deal with sequential data, implying scanning the same flow sequentially over time, which is not the case, making this choice misleading. Furthermore, this raises a question of what happens with benign flows if the value of $D(t)$ is continuously above the threshold. According to the algorithm, all flows should be inspected. However, there is no mention of a whitelist, meaning that the benign flows return to the pool of all flows considered for DDoS detection right after inspection. This behavior leads to an infinite loop due to the algorithm's event-driven nature, which strives to minimize the detection time. Therefore, the benign flows are repeatedly scanned as long as the $D(t)$ value is above the threshold.

To perform binary detection using LSTM, the authors choose a set of 40 out of 83 extracted features because the other 43 supposedly do not carry any relevant information. Within the selected features, they name flow-specific identifiers such as flow ID as well as source and destination IP addresses. This results in model overfitting because these values are arbitrary in real-world scenarios. To demonstrate model overfitting in the CICDDoS2019 dataset, we choose the exact features named in the paper and estimate their reminder based on the descriptions provided. Nonetheless, the results shown in Fig. 1 univocally confirm model overfitting.

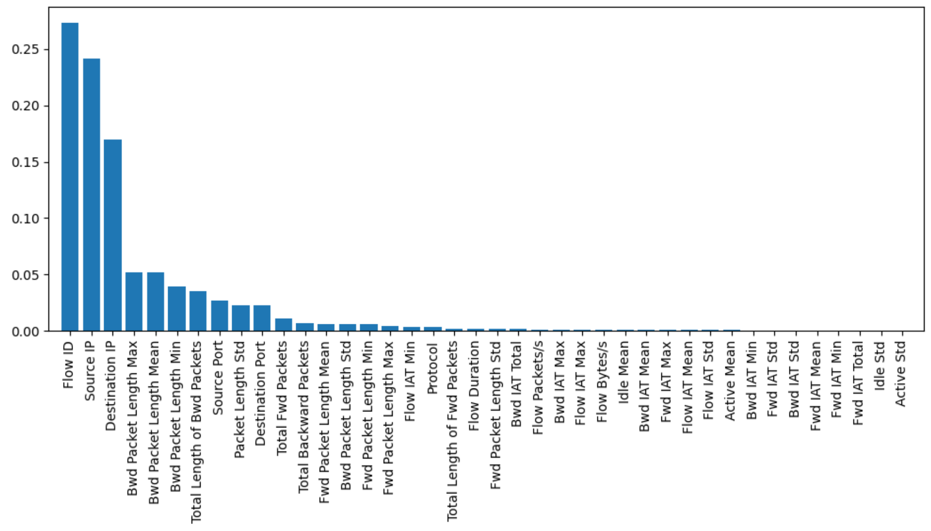


**Fig. 1.** Feature importance scores of the RF binary classifier.

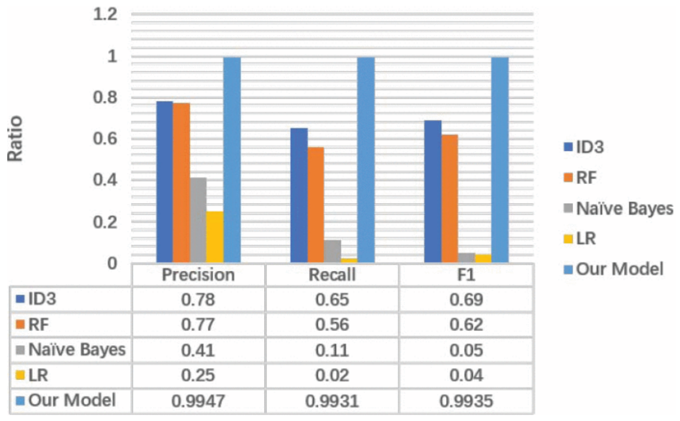
In the case of detecting a malicious flow, the next step is to classify it by applying a CNN multi-class classifier. The authors proposed using 80 features for training the CNN model, omitting only three out of 83 available features. The selection of features contradicts their previous statement, due to which they omitted 43 supposedly useless features for binary classification. In other words, the authors claim that 40 features are sufficient for a 98.9% accurate binary classification, while 40 additional features play no role in binary classification, yet they can be used for a 99.9% accurate multi-class classification. We assume that the remaining three features (RST Flag Cnt, PSH Flag Cnt, and ECE Flag Count) were omitted from both CNN and LSTM training datasets.

## 4 Validity of the Results

In this section, we discuss the results presented by the authors and challenge their claims and validity. The authors train Flowguard using the CICDDoS2019 dataset present their approach and compare its performance against other ML algorithms proposed in the original CICDDoS2019 paper. Except for LSTM, the classification results obtained in the Flowguard paper appear to be 100% matching with ones obtained in CICDDoS2019, including an error showing a lower F1 score than both accuracy and precision for the Naive Bayesian classifier. We show CICDDoS2019 and Flowguard's classification performance results exactly as presented in the original papers in Fig. 2.

The most obvious flaw in this comparison is that in CICDDoS2019, the authors applied ML algorithms for multi-class classification over a dataset of 80 features. While they did not mention specific training and testing dataset sizes, we can assume the ratio of 80% for training and 20% for testing based on the statement of using five-fold cross-validation. On the other hand, Flowguard utilizes binary classification over a set of 40 features in datasets whose training-to-testing size ratio is in the range of 95.2% / 4.8% to 99.6% / 0.4%, with training dataset sizes ranging from 2 to 30 million and testing dataset being fixed at 100 thousand flows. The authors mention variable training dataset sizes without providing a reason or mentioning the methodology used to form the training and test datasets. This dataset distribution poses a problem since some attack types have significantly more samples than others. Benign samples account for merely 0.16% of the entire dataset, translating to roughly 160 benign samples in a 100 thousand sample dataset. In summary, the authors of Flowguard obtained the same classification results as the authors of CICDDoS2019 while using an entirely different methodology without explicitly referencing the results used in their work. The only mention of CICDDoS2019 is that in Flowguard they tested the four proposed ML algorithms.

To clarify the ambiguities, we tested all of the mentioned ML algorithms using methodology identical to Flowguard's 30 million training samples scenario to perform binary classification on the CICDDoS2019 dataset. In our case, the testing dataset contained 175 randomly selected benign samples. While we acknowledge that this methodology is not representative, we proceed with the



| | Precision | Recall | F1 |
|---|---|---|---|
| ID3 | 0.78 | 0.65 | 0.69 |
| RF | 0.77 | 0.56 | 0.62 |
| Naïve Bayes | 0.41 | 0.11 | 0.05 |
| LR | 0.25 | 0.02 | 0.04 |
| Our Model | 0.9947 | 0.9931 | 0.9935 |

a) Flowguard

| Algorithm | Pr | Rc | F1 |
|---|---|---|---|
| ID3 | 0.78 | 0.65 | 0.69 |
| RF | 0.77 | 0.56 | 0.62 |
| Naïve Bayes | 0.41 | 0.11 | 0.05 |
| Logistic regression | 0.25 | 0.02 | 0.04 |

b) CICDDoS2019

**Fig. 2.** Classification results as shown in a) Flowguard [4] and b) CICDDoS2019 [8] papers.

experiment in an attempt to reproduce Flowguard's results. The results shown in Fig. 3 indicate a significant discrepancy between our findings and Flowguard's results. Contrary to Flowgurd's results, our findings show that all of the classifiers perform much better than they reported. The results indicate that all classifiers outperform LSTM, with ID3 and RF exhibiting impeccable performance, thus contradicting the results presented in the Flowguard paper. In the case of applying weighted average metrics, all models perform similarly because misclassifying 175 benign samples cannot drastically affect results over a 100 thousand samples dataset. Therefore, we show more detailed results in the Table 1. Due to a substantial difference in the number of benign and malicious samples in the datasets, we can conclude that the binary LSTM classification performance metrics presented in Flowguard are misleading. The list of exact features and source code will be available on the GitHub repository. [2].

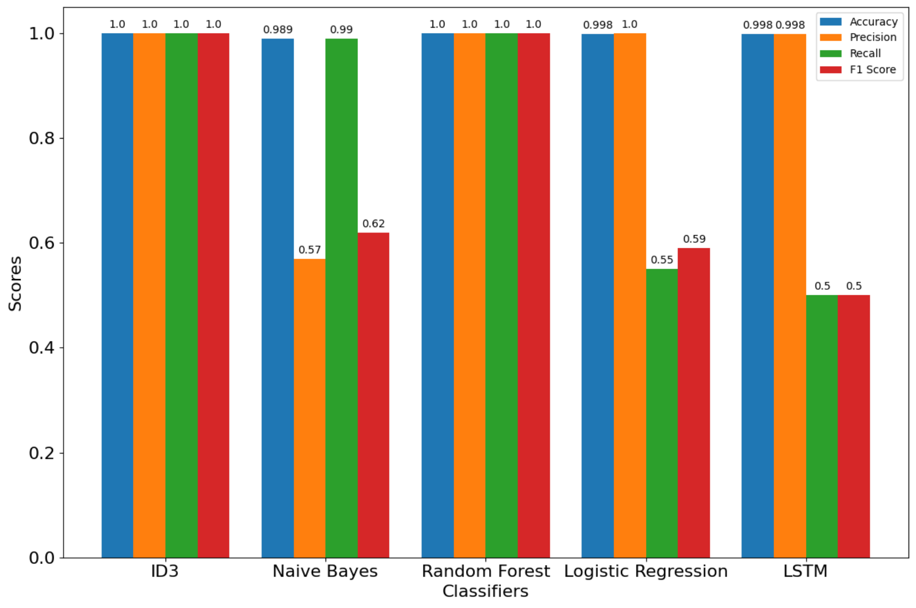


**Fig. 3.** Average performance metrics of the binary classifiers on the CICDDoS2019 dataset.

| **Classifier** | **Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
|---|---|---|---|---|---|
| ID3 | Benign | 1.00 | 1.00 | 1.00 | 175 |
| | Malicious | 1.00 | 1.00 | 1.00 | 100159 |
| | Macro avg | 1.00 | 1.00 | 1.00 | 100334 |
| | Weighted avg | 1.00 | 1.00 | 1.00 | 100334 |
| | Accuracy | | 1.00 | | 100334 |
| Naive Bayes | Benign | 0.14 | 1.00 | 0.24 | 175 |
| | Malicious | 1.00 | 0.99 | 0.99 | 100159 |
| | Macro avg | 0.57 | 0.99 | 0.62 | 100334 |
| | Weighted avg | 1.00 | 0.99 | 0.99 | 100334 |
| | Accuracy | | 0.9891 | | 100334 |
| Random Forest | Benign | 1.00 | 1.00 | 1.00 | 175 |
| | Malicious | 1.00 | 1.00 | 1.00 | 100159 |
| | Macro avg | 1.00 | 1.00 | 1.00 | 100334 |
| | Weighted avg | 1.00 | 1.00 | 1.00 | 100334 |
| | Accuracy | | 1.00 | | 100334 |
| Logistic Regression | Benign | 1.00 | 0.10 | 0.19 | 175 |
| | Malicious | 1.00 | 1.00 | 1.00 | 100159 |
| | Macro avg | 1.00 | 0.55 | 0.59 | 100334 |
| | Weighted avg | 1.00 | 1.00 | 1.00 | 100334 |
| | Accuracy | | 0.9984 | | 100334 |
| LSTM | Benign | 0.00 | 0.00 | 0.00 | 175 |
| | Malicious | 1.00 | 1.00 | 1.00 | 100159 |
| | Macro avg | 0.50 | 0.50 | 0.50 | 100334 |
| | Weighted avg | 1.00 | 1.00 | 1.00 | 100334 |
| | Accuracy | | 0.9982 | | 100334 |

**Table 1.** Performance metrics for different binary classifiers on the CICDDoS2019 dataset.

## 5 Conclusion

In this paper, we examined Flowguard, a seemingly perfectly suited network flow-based detection and mitigation mechanism against DDoS attacks targeting IoT devices. We discussed the reproducibility challenges of the proposed system and questioned the credibility of its efficiency. Our findings indicate that Flowguard fundamentally lacks a comprehensive overview of DDoS attacks, and its relevance to IoT is questionable due to several critical issues. The authors fail to justify why only four selected DDoS attack types are considered IoT-specific. They provide incorrect descriptions of some attacks and rely on a training dataset that does not include IoT devices or communication protocols. Moreover, the authors fail to define what constitutes a flow and the criteria for terminating flows, causing practical issues.

Upon further analysis of the DDoS attack detection mechanisms in Flowguard, we found an application of nonsensical statistical properties for malicious flow detection that the authors did not justify and whose effectiveness was not demonstrated. Flowguard heavily relies on the dataset's flow-specific features such as Flow ID and source and destination IP addresses, making it inapplicable in real-world scenarios where these features always differ. Furthermore, there are reasons to believe that the binary classification results were copied from another paper's multi-class classification without a reference rather than obtained through experimentation. We validated these hypotheses through experiments, showing a considerable discrepancy in comparison to Flowguard's findings, suggesting that they were made in such a manner to make Flowguard appear significantly better. Additionally, we show that justification for the usage of LSTM is false and that the classifier performed the worst, misclassifying every single benign sample as malicious. In summary, we conclude that the problem of timely DDoS detection in IoT, leveraging flow statistics, was not solved by Flowguard.

**Disclosure of Interests.** The authors do not have any competing interests to declare that are relevant to the content of the article.

# Bibliography

[1] Die.net: Slowhttptest - man page. https://linux.die.net/man/1/slowhttptest (2024), accessed: 2024-07-04

[2] Gavric, N., Bhandari, G.: Anti-flowguard (2024), https://github.com/SmartSecLab/anti-flowguard, github repository, accessed July 16, 2024

[3] Go, M., Contributors, B.: Bonesi - ddos botnet simulator (2013), https://github.com/Markus-Go/bonesi, gitHub repository, accessed July 4, 2024

[4] Jia, Y., Zhong, F., Alrawais, A., Gong, B., Cheng, X.: Flowguard: An intelligent edge defense mechanism against iot ddos attacks. IEEE Internet of Things Journal **7**(10), 9552–9562 (2020). https://doi.org/10.1109/JIOT.2020.2993782

[5] Lashkari, A., Contributors, C.: Cicflowmeter (2017), https://github.com/ahlashkari/CICFlowMeter, gitHub repository, accessed July 4, 2024

[6] Liu, L., Jin, X., Min, G., Xu, L.: Real-time diagnosis of network anomaly based on statistical traffic analysis. In: 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications. pp. 264–270 (2012). https://doi.org/10.1109/TrustCom.2012.233

[7] Rose, K., Eldridge, S., Chapin, L.: The internet of things: An overview. The internet society (ISOC) **80**(15), 1–53 (2015)

[8] Sharafaldin, I., Lashkari, A.H., Hakak, S., Ghorbani, A.A.: Developing realistic distributed denial of service (ddos) attack dataset and taxonomy. In: 2019 international carnahan conference on security technology (ICCST). pp. 1–8. IEEE (2019)

[9] Tripathi, N., Hubballi, N., Singh, Y.: How secure are web servers? an empirical study of slow http dos attacks and detection. In: 2016 11th International Conference on Availability, Reliability and Security (ARES). pp. 454–463 (2016). https://doi.org/10.1109/ARES.2016.20

[10] Tushir, B., Dalal, Y., Dezfouli, B., Liu, Y.: A quantitative study of ddos and e-ddos attacks on wifi smart home devices. IEEE Internet of Things Journal **8**(8), 6282–6292 (2021). https://doi.org/10.1109/JIOT.2020.3026023

[11] Yevsieieva, O., Helalat, S.M.: Analysis of the impact of the slow http dos and ddos attacks on the cloud environment. In: 2017 4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T). pp. 519–523 (2017). https://doi.org/10.1109/INFOCOMMST.2017.8246453

[12] Zargar, S.T., Joshi, J., Tipper, D.: A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks. IEEE Communications Surveys & Tutorials **15**(4), 2046–2069 (2013). https://doi.org/10.1109/SURV.2013.031413.00127