Troubleshooting an Intrusion Detection Dataset: the CICIDS2017 Case Study

Gints Engelen, Vera Rimmer, and Wouter Joosen
imec-DistriNet, KU Leuven
Leuven, Belgium
Email:{*firstname.lastname*}@kuleuven.be

***Abstract*—Numerous studies have demonstrated the effectiveness of machine learning techniques in application to network intrusion detection. And yet, the adoption of machine learning for securing large-scale network environments remains challenging. The community acknowledges that network security presents unique challenges for machine learning, and the lack of training data representative of modern traffic remains one of the most intractable issues. New attempts are continuously made to develop high quality benchmark datasets and proper data collection methodologies. The CICIDS2017 dataset is one of the recent results, created to meet the demanding criterion of representativeness for network intrusion detection.**

In this paper we revisit CICIDS2017 and its data collection pipeline and analyze correctness, validity and overall utility of the dataset for the learning task. During this in-depth analysis, we uncover a series of problems with traffic generation, flow construction, feature extraction and labelling that severely affect the aforementioned properties. We investigate the causes of these shortcomings and address most of them by applying an improved data processing methodology. As a result, more than 20 percent of original traffic traces are reconstructed or relabelled. Machine learning benchmarks on the final dataset demonstrate significant improvements. Our study exemplifies how data collection issues may have enormous impact on model evaluation and provides recommendations for their anticipation and prevention.

***Index Terms*—network intrusion detection, machine learning, benchmark dataset, data collection.**

I. Introduction

Network intrusion detection systems (NIDS) are security tools strategically deployed in a network environment that monitor and interpret internal and external traffic in search for malicious activities. Advances in Machine Learning (ML) over the last decades have enabled NIDS to evolve from simple rule-based systems to intelligent automated decision-making engines powered by modern learning algorithms [1]. Despite significant progress in ML-based defensive research, a lot of challenges are still open when it comes to NIDS solutions. For any data-driven technique, the role of *high quality, representative datasets* cannot be underestimated; this is especially true for NIDS research that has to account for high non-stationarity and strong heterogeneity of network traffic, together with an uncontrollable production environment.

In NIDS research, evaluation on benchmark datasets primarily serves a two-fold purpose: (i) *real-world performance estimation* of a particular algorithm, and (ii) *consistent comparison* between different approaches. In this respect, quality of data has a decisive influence on valid outcomes of both objectives. Research has shown that many benchmark datasets do not adequately represent the real problem of network intrusion detection, discrediting performance numbers achieved in laboratory conditions [1]. As a response, over the last 10 years the community has collectively devised the criteria that reliable research traffic data should meet [2]–[4]. These criteria can be seen as objectives of proper dataset generation, be that for capturing real traffic or simulating attack scenarios.

Recently, the Canadian Institute of Cybersecurity released a new dataset – CICIDS2017 [5] – with the aim to address some of the issues that plagued older datasets. CICIDS2017 is comprised of real traffic generated through simulated interactions between hosts in a controlled infrastructure, which means that the resulting network traffic retains basic limitations of a simulated environment. Nevertheless, the authors do a great effort to achieve heterogeneity, diversity and completeness of the dataset in order to maximize its utility for research purposes. Owing to that promise, CICIDS2017 is growing in usage in NIDS research [6]–[9]. Crucially, however, the dataset is largely taken at its face value, assuming that a modern dataset collected with the quality requirements in mind indeed satisfies these criteria. As a result, the studies use successful ML evaluations as a sole confirmation of correctness of CICIDS2017, but do not explicitly validate these results by examining the data. The lack of dedicated analysis sheds uncertainty on the empirical and theoretical research results obtained when leveraging the dataset. We have revisited the properties of CICIDS2017 as a NIDS dataset and analyse to what extent it satisfies the declared requirements.

Our contributions can be summarized as follows:

- We perform a thorough methodological manual analysis of the raw CICIDS2017 dataset and its respective feature extraction tool in order to review the dataset creation pipeline, consisting of attack simulation, flow construction, labelling, feature extraction and ML benchmarking. Our investigation reveals errors in attack simulation, feature extraction, labelling and benchmarking of the dataset. Moreover, we find out that more than 25% of all flows in the dataset turn out to be meaningless artefacts carrying no real identifying characteristics. For several attack categories, their fraction reaches 50%, which is significantly beyond an acceptable noise level. Further investigation confirms that the issue is still present in the newest version of the dataset - CIC-CSE-DS2018.

- We analyse the negative impact of the dataset errors, artefacts and mislabelled traces on ML algorithms, and provide insights into why these issues can easily pass undetected when using the dataset for NIDS research.
- Building on these findings, we modify the feature extractor and regenerate and relabel the CICIDS2017 dataset based on the original PCAPs[1]. Improved correctness and reliability of the regenerated dataset are further supported by ML benchmarks that reach the highest performance ever reported on these data.

II. Dataset & Related Work

A NIDS dataset is considered reliable when it covers the following properties. It should contain (a) real and (b) valid network traffic, such that it constitutes complete scenarios that can be observed in actual production environments. The benign and malicious traces should be (c) correctly labelled, (d) highly variant/diverse in terms of represented services, client behaviours and attacks, (e) correctly implemented, according to commonly accepted standards, and (f) easily updatable, when new services and attacks are introduced. The dataset should also be (g) reproducible, enabling comparison across different datasets, and (h) shareable, containing no private information. Finally, a good dataset should be supplied with (i) documentation and metadata of data collection, including details about the network infrastructure and simulated attack scenarios. CICIDS2017 was created as a response to these dataset quality evaluation criteria, and thus is asserted to satisfy all of them by design.

The dataset is generated in a complete network topology with diverse devices and operating systems as a testbed, where separate victim and attacker networks communicate over the Internet. To collect benign traffic that models realistic user behaviour, the authors use profiling agents [11] that are trained beforehand on network events generated through genuine human interactions in the network. Communication covers all common available protocols, such as HTTP, HTTPS, FTP, SSH and email protocols. In order to fulfil the requirement for attack diversity, the authors use the 2016 McAfee report to compose the list of most common attacks: Brute Force attacks, Heartbleed attack, Botnet communication, several variants of DoS attacks, DDoS, Infiltration attack and Web attacks. In total, the dataset comprises over 2.8 million traffic traces categorized into 15 classes. Individual traffic traces are represented by a feature vector of 80 extracted high-level statistical features, manually engineered based on expert knowledge of traffic characteristics relevant to intrusion detection.

Since 2017, plenty of studies have deployed CICIDS2017 (and its latest version CSE-CIC-IDS2018) for NIDS research [6]–[9]. The studied research problems span ML-assisted network intrusion detection, novelty detection, anomaly detection, online detection, with a large variety of learning algorithms and traffic representations – raw PCAPs, statistical features, graphs, etc. Recently, both datasets started being considered for adversarial ML research, such as evasion of botnet detection [12]. The conclusions reached by these studies largely rely on correctness and validity of the dataset in all its forms. There are studies that have analyzed features used in the dataset [13] and raised the issue of class imbalance [14]. However, these analyses do not address correctness and validity of the dataset – the properties that are strongly required to support any research results obtained from the data.

[1]The regenerated dataset and modified feature extractor can be found at downloads.distrinet-research.be/WTMC2021 [10], where we also provide a more low-level documentation of all our changes.

III. Manual analysis and corrections

We select CICIDS2017 for in-depth analysis based not only on its growing adoption in the literature, but also on sufficient transparency and reproducibility of its generation process. This section describes the main analysis of the dataset and its feature extraction tool. Our goal here is to revise the aforementioned properties of a good NIDS dataset, which we do through examining the following dataset creation stages:

- *Attack simulation.* Most research datasets for NIDS contain malicious traces representative of target malicious activities. To include a wide variety of attacks, and save the effort of annotating real traffic, the attacks are often simulated with open source tools or custom scripts.
- *Flow construction.* Raw network traffic is a continuous data stream that needs to be monitored and analyzed in real time. It is therefore necessary to decide what the level of granularity will be, i.e. what constitutes a single input unit for the detection system. Traffic is commonly processed at granularity of a *flow* associated with one complete network connection.
- *Labelling.* When working with a supervised detection system, the aim is not only to distinguish between benign and malicious traffic, but to label malicious traffic according to the attack class that it belongs to. When training an ML-based classifier for this purpose, it needs to be supplied with example network data from all classes that it needs to be able to detect, along with the correct label for each input. When testing the model, comparing correct labels with predicted classes allows to assess the model's performance.
- *Feature extraction.* Traditional ML systems cannot operate on multidimensional raw traffic data, and instead require some form of structured input. Therefore, a flow-based intrusion detection system would convert each raw traffic flow to a fixed-length structured vector that encapsulates all the important information about the flow in a form of numerical and/or categorical *features*. For instance, a NIDS could extract statistical features of the flow, such as *total flow duration, total forward packets, average inter-packet arrival time, maximum packet length*, and others.
- *ML benchmarks.* Since the dataset is meant for design and optimization of data-driven network security solutions, the final stage of its creation is performing ML benchmarks. Benchmarking includes designing, training and testing one or more ML models on the constructed

dataset for a lab evaluation of network intrusion detection. This stage is meant to confirm that the dataset and its features are meaningful and appropriate for ML-based NIDS, and optionally to compare performance of various algorithms.

Understanding the underlying mechanisms of dataset generation enables researchers and practitioners to properly assess relevance of the data to the problem at hand, foresee potential pitfalls of applying ML to these data, and essentially set correct expectations. Dataset documentation is crucial in facilitating this analysis, and in case of CICIDS2017 we use several sources of documentation: (i) the original paper [5], (ii) the website page describing the dataset and its features [15], and (iii) the github page of the feature extraction tool [16]. In the following, we describe our analysis process for each dataset generation stage of interest, in the order applied during our study, and report our findings.

A. Flow construction

CICIDS2017 flows are constructed from raw PCAP files using the CICFlowMeter tool [16] by the same authors, which outputs CSV files where each row corresponds to a flow, and each flow has 83 (excluding the label) features. They define a flow as a bidirectional exchange of network packets belonging to the same *5-tuple* – a unique set {*source IP address, destination IP address, source port, destination port, transport layer protocol*} – within a certain time period.

1) TCP "appendices". We analyze the source code of CICFlowMeter to understand how each flow is recorded. A flow is started upon observing a packet that does not belong to an active flow. A flow is terminated either upon timing out, or when the network connection is closed. In this dataset the timeout value was set to *120 seconds*. When the underlying network protocol is TCP, CICFlowMeter also considers the network connection closed (and the corresponding flow completed) upon detecting the first packet in the flow that contains a FIN flag. We notice that this design decision violates the TCP specification [17] that stipulates that a FIN flag merely indicates that the sender is done transmitting data. A TCP connection, however, is only terminated when both sides have sent a FIN packet to each other.

The result of the misunderstanding of the TCP mechanism has rather drastic consequences: if a flow is considered terminated already after the first of the actual two FIN packets, any subsequent packets belonging to the same TCP connection will constitute their own flow. These flows mostly consist of just two leftover ACK and FIN packets and are labelled as the original flow, be that benign or an attack. We will henceforth refer to these flows as *"TCP appendices"*, since they constitute the (superfluous) end of the underlying TCP connection. In Section IV we show that the TCP appendices are alarmingly present, making up *25.9* percent of the entire dataset and study their impact.

2) Timeout and TCP appendices. After a flow times out, a subsequent flow with the same *5-tuple* maintains the source and destination of the timed out flow, which guarantees consistency across multiple flows spanning a single (long) TCP connection. There is however a complex problem in this dataset where a TCP appendix timing out can sometimes cause the subsequent flow to go in the wrong direction (that is, the *Source* and *Destination* features of the flow are swapped). A detailed description of this phenomenon is however out of scope for this work, and we once again refer the interested reader to our detailed documentation [10].

As we will see in section III-B, the labelling logic used in this dataset relies on correct flow direction to assign its labels. Consequently, when attack flows go in the wrong direction, they risk being mislabelled as benign.

3) Ignoring RST. According to the specification [17], the RST packet is used to reset a TCP connection. We observe, however, that the CICFlowMeter tool does not consider the RST packet as a valid way to terminate a TCP flow.

B. Labelling

The authors provide information about the IP addresses of attackers and victims as well as the time window during which each attack was executed, which should allow for reproducibility of the labelling logic.

1) Time frames. We found that the time frames reported on the dataset information page [15] are not precise, therefore we had to tune them to reproduce the labelling. We report these details along with the new version of the dataset [10].

2) Attempted attack flows. A crucial observation is that the employed labelling strategy relies solely on a flow's source and destination IP and a specific data collection window, and thus labels a traffic flow between an attacker and a victim as malicious simply on the grounds of being collected at a certain moment in time. Consequently, a resulting flow's content and characteristics are not verified to ensure that malicious or suspicious activity is taking place.

We found at least one potential issue with that, which affects most attack classes. All attacks, except for PortScan, rely on an established TCP connection in order to deploy malicious functionality. Within this established TCP connection, there needs to be some kind of communication in the form of a payload for the attack to take place.

However, we observe that *Web Attack - Brute Force* and *Web Attack - XSS* largely consist of flows that have no data transfer in the forward direction, which means that the attacker never actually executes an attack within those flows. *Bot* traffic also contains a large number of failed or empty TCP connections, which do not contain meaningful botnet traffic and thus should not be labelled as such. Moreover, when a DoS attack brings a web server down, making it unable to properly respond to new incoming connection requests, these numerous TCP connections without a HTTP payload are still labelled as an instance of a HTTP-based DoS attack.

We conclude that labelling these examples as attacks is inappropriate for the type of a flow-based NIDS which analyzes each flow in isolation from another and cannot judge whether an observed failed or empty TCP connection is a part of a larger ongoing malicious campaign. That is, CICIDS2017 flows that are generated by a connection initiated by an attack simulation tool but do not contain any forward data transfer should be distinguished from self-contained malicious connections. We thus decide to label these flows as *'X - Attempted'*, where *X* refers to the original attack label (IV-B).

C. Attack simulation

Most attacks are generated by executing an automated tool from one or more hosts in the attacker network, while other attacks are executed by Python scripts written by the authors. Some attack categories (such as Bot, Infiltration) assume that the victim host is compromised, after which the victim will be initiating connections with the attacker.

1) DoS Hulk misimplementation. Our analysis reveals that the DoS Hulk tool used in this dataset simulation is deprecated and should not be used to execute a DoS attack. The DoS Hulk tool aims to cause the target web-server to keep a connection open during 110-120 seconds. However, instead of setting the *Connection* header field in the HTTP request to *Keep-Alive*, this implementation sets it to *Close*. As a result, the web server sets the FIN flag on the outgoing HTTP response packet, initiating to close the connection. As this renders the DoS Hulk attack ineffective, we do not believe this attack class in CICIDS2017 can be used as intended.

2) Diversity. The authors of the dataset ensure diversity in benign behaviour and include a wide range of attacks. However, diversity of data within each attack category is not openly discussed. We are not aware whether the attack tools were used with a certain fixed configuration or with varying parameters, which affects how representative each attack class is of the overall attack strategy. A supervised NIDS trained on attack traces that were simulated using one configuration of the attack tool may not generalize to other configurations, making NIDS behaviour unpredictable on slightly differing attack implementations.

For the sake of diversity within a single attack category, we recommend combining several tools and their configurations for attack simulation in future research.

D. Feature extraction

The dataset uses flow-level statistical features as a traffic representation. These features are calculated in the CICFlowMeter feature extraction tool.

Shortcut learning. The original CSV files still contain attributes such as *Flow ID, Source IP, Destination IP, Source Port, Destination Port,* and *Timestamp* of the flow. Whereas using the destination port as a NIDS feature is up for debate, the other attributes should be excluded from the feature representation during training. This should prevent a flow-based NIDS from associating a certain timestamp or some host-specific information with a certain class without actually learning the underlying problem.

This ties in to a larger problem in ML, where features unrelated to the underlying problem happen to be the most discerning between each class. This phenomenon is generally referred to as *shortcut learning*. While reliable detection of shortcut learning is still an open research problem, we recommend analyzing the *importance* of various features used in prediction as a sanity check for any feature-based classifier.

We evaluate the issue of shortcut learning at length within the context of this dataset in Section IV.

E. ML benchmarks

The authors of CICIDS2017 evaluate standard ML classifiers [5]: Random Forest (RF), Multi-Layer Perceptron (MLP), K-Nearest Neighbors, etc. Apart from execution speed for training and testing, they report aggregated performance metrics for the whole test set, based on the total number of true positives (TP), false positives (FP) and false negatives (FN):
Precision: the ratio of correctly detected attacks to all traces classified as malicious $TP/(TP + FP)$;
Recall: the ratio of correctly detected attacks to all actually malicious traces $TP/(TP + FN)$;
F1-score: the harmonic mean of precision and recall.

We list a number of issues with the conducted benchmarks, which affect reliability and reproducibility of reported performance estimates. First, the lack of per-class performance measurements, such as a confusion matrix or per-class F1-score, does not allow to fully assess utility of CICIDS2017 for NIDS. Providing only aggregated precision, recall and F1-score for a given highly imbalanced dataset obscures effectiveness of the ML algorithms for different attack classes. Our evaluation shows that a ML-based NIDS fails on numerous flows of some of the attacks, encouraging deeper investigation. We provide revised ML benchmarks according to the suggested methodology in the next section.

IV. Evaluation

In this section we evaluate a Random Forest (RF) classifier on the original and corrected datasets and analyze the difference. Through these benchmarks we also empirically investigate the impact of TCP appendices and mislabelled traces on ML-based NIDS, and illustrate why it is so challenging to detect such errors.

A. Regenerating the dataset

We analyze the impact of improvements suggested in Section III by applying dataset transformations in three subsequent stages. Table I reports on the number of traces per category for each transformation step.

1) Original. We reconstruct the *Original* dataset with the original CICFlowMeter tool and label it in such a way that it matches the public dataset version as closely as possible.

2) Intermediate: fixing TCP-related issues. We correct flow construction errors (III-A) by fixing CICFlowMeter so that the TCP appendices are systematically reunited with the "head" parts of the underlying TCP connections, while also making sure that an RST packet correctly terminates a flow. Our goal here is to have TCP flows accurately reflect their underlying TCP connection. Subsequently, we see a sharp decrease in flow counts across most classes in the second column of Table I, where the biggest change happens in classes where

TABLE I: Structures of the three versions of the dataset (the most prominent changes of flow counts in bold blue). *The “attempted” category combines all *Attempted* attack flows.

Label	**Original**	**Intermediate** (appendices fixed)	**Final** (payload filter)
BENIGN	2271326	1823964	1823964
ATTEMPTED*	0	0	**447362**
FTP-Patator	7934	**3984**	3973
SSH-Patator	5898	**2988**	2980
DoS GoldenEye	10293	**7647**	7567
DoS Hulk	230124	**159048**	158469
DoS Slowhttptest	5499	5109	**1742**
DoS slowloris	5791	5707	**4001**
Heartbleed	11	11	11
Web Brute Force	1507	1365	**151**
Web Attack XSS	652	679	**27**
Web Attack SQL	21	12	12
Infiltration	36	48	32
Bot	1956	2208	**738**
PortScan	158842	159023	159023
DDoS	128022	**95123**	95123

TCP connections most often finish with FIN packets. In some classes we also see an increase of the number of flows. This happens on one hand due to the aforementioned RST packet fix, as well as due to the issue described in section III-A2 which caused some malicious traces to be mislabelled as *benign*, and which is resolved upon fixing the TCP appendices. Note that we did not alter the original *flow timeout* value.

3) Final: payload filter. Following our conclusion in III-B2, we apply the payload filter to all classes except for *Benign* and *PortScan*. This filter ensures that attack classes predominantly contain flows that actually exhibit malicious activity. We notice sharp drops in several categories (Table I).

Web Attacks: Brute Force and XSS. Manual inspection of these attacks showed that the large majority of their flows do not contain any packet data. The attacks are largely executed through a select few flows, which each of them containing many brute force or XSS attempts.

Bot. Manual inspection showed that when the infected victims attempt to connect to the *Botnet master*, they are rejected in 2 out of 3 cases. These rejected connections carry no meaningful malicious content, and were thus relabelled.

DoS: Slowhttptest and Slowloris. With both attacks, our inspection reveals a large amount of failed TCP connections. Unlike with Bot, this can be explained by the attack strategy itself, which appears to have succeeded at bringing the victim server down, at which point it is no longer able to respond to incoming connection requests.

B. Experimental results

We perform ML benchmarks on the three dataset versions following the evaluation methodology introduced in Section III-E and report results of RF – the classifier with the best average F1-Score – in Table II. For these experiments we decided to assign all *“Attempted”* flows to the BENIGN category. We used a train-test split of 75-25. For the source code, hyperparameter details and random seeds we refer the reader to our extended documentation [10]. Note that results of the analysis might differ based on the used ML model.

TABLE II: RF classifier performance metrics with the test set ratio 25% (the most impacted attack classes in bold blue). *Test sets of these classes are extremely small [10].

	Original			**Intermediate**			**Final**		
Label	**Pr**	**Re**	**F1**	**Pr**	**Re**	**F1**	**Pr**	**Re**	**F1**
BENIGN	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
FTP-Patator	0.99	0.99	0.99	1.00	1.00	1.00	1.00	1.00	1.00
SSH-Patator	0.99	0.98	0.99	1.00	0.99	0.99	1.00	0.99	0.99
DoS GoldenEye	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
DoS Hulk	0.99	0.99	0.99	0.99	0.99	0.99	1.00	0.99	0.99
DoS Slowhttptest	0.98	0.99	0.98	0.97	0.99	0.98	0.99	0.99	0.99
DoS slowloris	0.99	0.99	0.99	0.99	0.99	0.99	1.00	0.99	0.99
Heartbleed*	1.00	0.67	0.77	1.00	0.67	0.78	1.00	**0.89**	**0.93**
Web Brute Force	0.70	0.85	0.77	**0.69**	**0.91**	**0.78**	**1.00**	**0.91**	**0.95**
Web Attack XSS*	0.52	0.11	0.18	**0.52**	**0.16**	**0.25**	**1.00**	**0.67**	**0.79**
Web Attack SQL*	0.33	0.07	0.11	**0.00**	**0.00**	**0.00**	0.00	0.00	0.00
Infiltration*	0.96	0.70	0.81	**1.00**	**0.50**	**0.66**	1.00	**0.83**	**0.91**
Bot	0.92	0.45	0.60	**0.79**	**0.40**	**0.53**	**1.00**	**0.99**	**0.99**
PortScan	0.99	0.99	0.99	0.96	0.97	0.97	0.96	0.97	0.97
DDoS	0.99	0.99	0.99	1.00	1.00	1.00	1.00	1.00	1.00
Accuracy		0.99			0.99			0.99	
Weighted Avg	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99

While the aggregated performance metrics at the bottom of Table II show no difference, the applied transformations have significantly affected NIDS performance for various categories. This illustrates why analyzing the imbalanced dataset at the level of aggregated ML metrics is inadequate.

1) TCP appendix impact. We see that removing appendices marginally improves performance for some classes, but also gives a performance *decrease* for other classes. Note that *Web Attack - SQL Injection* ends up with 12 traces in total, and all 3 test traces are mispredicted as *Benign*. This may be due to the attack being hardly distinguishable from benign traces and poorly represented in the training data. For other classes where performance worsened, the appendices in the *Original* dataset are more likely to be classified correctly than normal attack flows. This points to the model *overfitting* on the appendix instances, memorizing their features instead of learning actual attack patterns. The model will be able to overfit to such data only if (1) appendices of different attack classes can be differentiated, and (2) appendices can be distinguished from regular flows of the same class. To assess whether these conditions hold, we apply extensive *feature importance* measurements for the RF classifier, the details of which can be found online [10].

TCP appendix features indeed have widely varying distributions across different classes, and most importantly, in all cases these features do not have any semantic connection with the intended malicious activity, but rather stem from the specifics of the used attack tools. We find that the model overfits through *shortcut learning*: it uses four prominent but largely irrelevant features to crisply differentiate between appendices of different classes (*Fwd Header Len*, *Total Fwd Pkt*, *Min Seg Size Fwd*, and *Init Win bytes Fwd*), and mainly two features (*SYN Flag Count*, *Bwd Pkt Len Mean*) to separate appendices from regular flows.

2) Payload filter impact. The final transformation through the targeted payload filter triggers a large change towards much higher performance on Brute Force, XSS and Bot attacks. This indicates that Bot flows without an established TCP connection were a source of confusion for the RF classifier. Our experiments revealed that the presence of RST packets was a learned shortcut for Bot traffic in the *Original* dataset, which means that the classifier learned to recognise failed TCP connections instead of actual Botnet traffic. In the *Final* dataset with near-perfect benchmark results the *absence* of RST was an important Bot traffic feature.

We also see no significant change in performance for DoS Slowhttptest, despite the large (almost 70%) decrease in total number of flows. However, through additional feature importance analysis we see that the model starts to learn actually relevant features, such as *Inter-packet arrival time*. Given that a DoS Slowhttptest attack sends intermittent fragmented HTTP packets, this feature is a strong indicator of this type of attack, and is something a human operator would look for as well.

The experiments further confirm that aggregated ML metrics computed on a largely imbalanced multi-class dataset are inadequate to judge its utility for NIDS. Class-based metrics, feature importance analysis and manual investigation of mispredictions are crucial in performance evaluation and will aid in bringing attention to suspicious data points and spurious correlations that would otherwise have gone unnoticed.

V. Conclusion

In this paper we revised the creation process of the widely adopted CICIDS2017 dataset. Through analyzing traffic generation, attack simulation, flow construction, feature extraction, labeling and benchmarking stages of data collection, we discovered a series of problems that violate some of the established properties of a high quality NIDS dataset. The misimplementation of the DoS Hulk attack, the misunderstanding of the TCP protocol in flow construction and errors in feature extraction by the *flawed* CICFlowMeter tool violate the requirement for *correct implementation of network traffic*. Notwithstanding the great effort by the CICIDS2017 team to provide documentation on the infrastructure and data collection, we observed a *lack of documentation* concerning flow construction and parameters of attacks. Moreover, an overly general labelling strategy without additional validation of malicious traces caused a significant number of flows to be *mislabelled*. We correct the CICFlowMeter tool, and regenerate and relabel the new version of CICIDS2017 based on the original traffic. Refined machine learning benchmarks and feature importance analysis indicate improved validity and utility of the dataset. Crucially, our preliminary investigation into CSE-CIC-IDS2018, the successor of CICIDS2017, also revealed errors in flow construction. We strongly recommend to further analyze this dataset and only use the corrected version of CICFlowMeter for flow-based analysis.

We introduced an *"Attempted"* label for each attack class in order to decouple *intent* from *effect*. As this dichotomy applies to many security datasets, future research should investigate its other possible sources and devise appropriate practices.

Our analysis provides more insight into the process of dataset creation, which not only allows one to make informed decision when selecting data for lab evaluations, but also – crucially – sets correct expectations and overall contributes to designing better security solutions. We expect our findings and recommendations to support researchers and practitioners in the pursuit of better datasets and evaluation approaches.

VI. Acknowledgments

We thank Jin Li for providing us with the labelling code. We also thank Lieven Desmet for proofreading the paper and for his helpful insights, and the reviewers for their valuable suggestions. This research is funded by the Research Fund KU Leuven and by the Flemish Research Programme Cybersecurity.

References

[1] H. Hindy, D. Brosset, E. Bayne, A. K. Seeam, C. Tachtatzis, R. Atkinson, and X. Bellekens, "A taxonomy of network threats and the effect of current datasets on intrusion detection systems," *IEEE Access*, vol. 8, pp. 104 650–104 675, 2020.

[2] I. Sharafaldin, A. Gharib, A. H. Lashkari, and A. A. Ghorbani, "Towards a reliable intrusion detection benchmark dataset," *Software Networking*, vol. 2018, no. 1, pp. 177–200, 2018.

[3] E. K. Viegas, A. O. Santin, and L. S. Oliveira, "Toward a reliable anomaly-based intrusion detection in real-world environments," *Computer Networks*, vol. 127, pp. 200–216, 2017.

[4] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, pp. 147–167, 2019.

[5] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." in *ICISSP*, 2018, pp. 108–116.

[6] L. Leichtnam, E. Totel, N. Prigent, and L. Mé, "Sec2graph: Network attack detection based on novelty detection on graph structured data," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2020, pp. 238–258.

[7] A. Rosay, F. Carlier, and P. Leroux, "MLP4NIDS: An efficient MLP-Based network intrusion detection for CICIDS2017 dataset," in *International Conference on Machine Learning for Networking*. Springer, 2019, pp. 240–254.

[8] D. Stiawan, M. Y. B. Idris, A. M. Bamhdi, R. Budiarto *et al.*, "CICIDS-2017 dataset feature analysis with information gain for anomaly detection," *IEEE Access*, vol. 8, pp. 132 911–132 921, 2020.

[9] J. L. Leevy and T. M. Khoshgoftaar, "A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 big data," *Journal of Big Data*, vol. 7, no. 1, pp. 1–19, 2020.

[10] "Extended documentation of the corrected CICFlowMeter, generated CICIDS dataset, and the source code of the paper," https://downloads.distrinet-research.be/WTMC2021.

[11] A. Gharib, I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "An evaluation framework for intrusion detection dataset," in *2016 International Conference on Information Science and Security (ICISS)*. IEEE, 2016, pp. 1–6.

[12] G. Apruzzese, M. Colajanni, and M. Marchetti, "Evaluating the effectiveness of adversarial attacks against botnet detectors," in *2019 IEEE 18th International Symposium on Network Computing and Applications (NCA)*. IEEE, 2019, pp. 1–8.

[13] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "A detailed analysis of the CICIDS2017 data set," in *International Conference on Information Systems Security and Privacy*. Springer, 2018, pp. 172–188.

[14] A. Abdullah Alfrhan, R. Hamad Alhusain, and R. Ulah Khan, "SMOTE: Class imbalance problem in intrusion detection system," in *2020 International Conference on Computing and Information Technology (ICCIT-1441)*, 2020, pp. 1–5.

[15] C. I. for Cybersecurity, "Intrusion detection evaluation dataset (CICIDS2017)," https://www.unb.ca/cic/datasets/ids-2017.html, 2017, accessed: 2021-01-22.

[16] "CICFlowMeter tool," https://www.unb.ca/cic/research/applications.html, accessed: 2021-01-09.

[17] J. Postel *et al.*, "RFC 793: Transmission control protocol specification," 1981.