Errors in the CICIDS2017 dataset and the significant differences in detection performances it makes

Maxime Lanvin, Pierre-François Gimenez, Yufei Han, Frédéric Majorczyk, Ludovic Mé, Eric Totel

Errors in the CICIDS2017 dataset and the significant differences in detection performances it makes*

Maxime Lanvin[2], Pierre-François Gimenez[2], Yufei Han[1], Frédéric Majorczyk[3], Ludovic Mé[1], and Éric Totel[4]

[1] Inria, Univ. Rennes, IRISA, {firstname.lastname}@inria.com
[2] CentraleSupélec, Univ. Rennes, IRISA, {firstname.lastname}@centralesupelec.fr
[3] DGA-MI, Univ. Rennes, IRISA, frederic.majorczyk@intradef.gouv.fr
[4] Samovar, Télécom SudParis, Institut Polytechnique de Paris, eric.totel@telecom-sudparis.eu

Abstract. Among the difficulties encountered in building datasets to evaluate intrusion detection tools, a tricky part is the process of labelling the events into malicious and benign classes. The labelling correctness is paramount for the quality of the evaluation of intrusion detection systems but is often considered as the ground truth by practitioners and is rarely verified. Another difficulty lies in the correct capture of the network packets. If it is not the case, the characteristics of the network flows generated from the capture could be modified and lead to false results. In this paper, we present several flaws we identified in the labelling of the CICIDS2017 dataset and in the traffic capture, such as packet misorder, packet duplication and attack that were performed but not correctly labelled. Finally, we assess the impact of these different corrections on the evaluation of supervised intrusion detection approaches.

Keywords: intrusion detection · dataset labelling · machine learning

1 Introduction

Information technologies revolutionized our communication, collaboration, production, and consumption. Since they are now so profoundly connected with critical systems and crucial data, they are regularly targeted by malicious users that seek to break information confidentiality, integrity or availability. Many security mechanisms have been proposed against such attacks, notably Network Intrusion Detection Systems (NIDS) that aim at identifying attacks by monitoring network traffic in the target system. This detection involves the analysis of network traffic, generally by looking for traces of known attacks. Unfortunately, NIDS are prone to false positives and false negatives that can significantly impact cost and performance. For this reason, their performance must be carefully

* This work has been partly realised thanks to a doctoral grant from the cyber excellence pole (PEC : DGA, Brittany Region).

evaluated. This evaluation relies extensively on the use of benchmark datasets of network traffic. These datasets consist of two parts: the network data (either raw network packets or a more high-level network flow description) and the labels, i.e., the class (benign or attack) in which each packet or flow belongs.

Due to privacy and confidentiality reasons, there are only a few public datasets of real traffic for evaluating NIDS [12]. To circumvent those constraints, other datasets are generally obtained by generating network traffic in a testbed. One of these datasets is CICIDS2017 [14]. Though currently considered to be of good quality and widely used, it has nevertheless been criticized. Engelen et al. [4] notably pointed out some flaws in CICFlowMeter, the tool used to create flow descriptions from raw traffic capture, as well as issues with labels of some network flows that should not be labelled as attacks (network flows without a payload).

We discovered several new problems in CICIDS2017: most notably, several port scan attacks were not properly labelled, and a non-negligible part of the traffic capture was duplicated, leading to feature extraction and labelling issues. In addition to providing corrected traffic captures and labels, we took advantage of this opportunity to investigate why some references of the literature [11] exhibited high recall and precision values even though the dataset has serious labelling issues. We thus present in this paper three contributions:

- we first release a fixed version of the CICIDS2017 dataset for both labels and network captures,
- we propose a patch for CICFlowMeter that avoids processing malformed input data,
- we evaluate the consequences of the different dataset corrections on the evaluation of several popular intrusion detection models.

The rest is organized as follows. Section 2 presents some related works on network datasets and intrusion detection models. Section 3 highlights the identified errors of labelling and how to fix them. Finally, we measured the impact of the corrections on the evaluation of supervised approaches in Section 4.

2 Related works

2.1 Datasets

Several datasets have been proposed to evaluate the performances of intrusion detection tools. DARPA98 [8] was one of the first datasets provided to the academic community. Several datasets like KDD99 or NSLKDD [15] were then derived from this first dataset. Even though they are still widely used by the research community, these datasets have been heavily criticized [16,5] and are generally considered obsolete. In 2015, the UNSW-NB15 dataset [10] was proposed to offer modern traffic to evaluate NIDSes. This dataset is not well fitted for anomaly-based intrusion detection as the experiment duration is only about 30 hours, and there is no period of time free of attacks. In 2018, the Canadian Institue for Cybersecurity (CIC) provided the CICIDS2017 dataset [14] and, together with the Communications Security Establishment, the CSE-CIC-IDS2018 [3].

CICIDS2017 uses a network architecture with machines using several common Operating Systems (OS), namely GNU/Linux, macOS and Windows, along with a firewall, switches and routers. The traffic is emulated through a testbed architecture. This architecture is divided into a victim network with four machines and an attacker network with fifteen machines. The traffic was collected on work hours during five days, from Monday to Friday. Only the first day of the week is free from attack. During the four remaining days, a large variety of attacks was conducted. The attacks in the datasets are brute force attacks (FTP and SSH), Web attacks like XSS and SQL injection, Deny of Service (DoS) attacks and its distributed version (DDoS), port scan, botnet communications and infiltration. The CSE-CIC-IDS2018 includes the same attacks, but the network architecture is much larger and more complex. The network traffic is captured for ten days instead of only five in CICIDS2017.

In CICIDS2017 and CSE-CIC-IDS2018, the authors provided the raw network captures as pcap files and the network flow descriptions as CSV files. These network flow descriptions contain high-level descriptions of a network flow between a source (that initiated the communication) and a destination. The descriptions include various network statistics, notably source IP, destination IP, source port, destination port, protocol, packet number and flow duration. These flows are bidirectional, meaning that each one contains information on both sides of the communication, from source to destination and from destination to source (in contrast, for example, to the NetFlow format proposed by Cisco). The translation from network traffic to network flow descriptions is performed by the CICFlowMeter tool[5].

2.2 Machine learning use on CICDS2017

Most papers that use these datasets rely on machine learning models to learn and detect attacks. In that case, the datasets are generally split in two: one part is used for learning the model (called the "train set"), and the other part is used for evaluation (called the "test set"). The popular models [7], [11], [9] for these datasets include decision trees, k nearest neighbors, naive Bayes classifier, Random Forest [1], SVM [2], and multilayer perceptron [6]. These methods are used in a supervised learning setting, where the train set is labelled and contains both benign and malicious traffic. For example, Maseet et al. [9] obtain very high performances on CICIDS2017: out of the seven experimented supervised methods, five of them have an F1-Score, a recall and a precision higher than 0.99. In almost all these works, researchers based the learning and the evaluation on the network flow descriptions and not the raw network captures.

2.3 Previous criticism on CICIDS2017

In 2021, Engelen et al. [4] revealed several issues they found in the CICIDS2017 intrusion detection dataset. They found several flaws in the CICFlowMeter tool and that some attacks in the dataset were not well executed and thus ineffective.

[5] https://github.com/ahlashkari/CICFlowMeter

About the first issue, CICFlowMeter wrongly splits TCP connections because of a wrong implementation of the TCP connection termination. This phenomenon has two consequences. The first one is to create a lot of erroneous network flows since it splits a unique network connection into multiple ones. The second consequence is that the direction of the network flow description can be inverted. In that case, all the forward and backward data are swapped, including the IP addresses, which is damageable when it comes to labelling the dataset since the network flows are labelled based on their source and destination IP addresses. Engelen et al. released a fixed version of CICFlowMeter that avoids many labelling issues. In the rest of the article, we will only use this updated tool, not the original one.

The authors also found that some attacks were conducted without sending malicious payload. This is an issue because the attacks become ineffective, so the maliciousness of these packets is debatable. To overcome this issue, they decided to create another class of labels to account for these attack attempts.

In 2022, Rosay et al. [13] presented other issues related to CICFlowMeter, such as feature duplication, miscalculations and wrong protocol detection, as well as labels issues for several attacks. However, their handling of TCP termination is not perfect and misses some packets leading to distorted statistics. For this reason, we work with Engelen et al. flow descriptions.

3 Errors in the CICIDS2017 dataset and the CICFlowMeter tool, and their fixes

Pursuing the work of Engelen et al., we found four different issues in the CICIDS2017 dataset: a case where CICFlowMeter failed to properly create correct flow descriptions, incoherent timestamps, some duplication in the network captures, and an attack that is omitted from the labels.

The first two issues have consequences on the network flow descriptions and lead to an inversion of the source and the destination of the network flow descriptions that may impact labels. The third issue has only an impact on the network flow descriptions. The last one has an impact on the labels directly.

To explain why the first two issues may impact the labels, we must explain the labelling process we used. It must be noted that we do not have insights into how the authors of CICIDS2017 labelled those network flow descriptions after they generated the network flow descriptions from the network capture with the CICFlowMeter tool. However, Engelen et al. provide an automated script to label the network flows as attacks using their source IP address, destination IP address and timestamp. Indeed, the documentation of the CICIDS2017 dataset provides the time periods and the IP addresses concerned by the attacks. We used the same process to label the network flows. As this process takes into account the source and destination, an inversion of those IP addresses may lead to errors in the labels of the network flows.

Those four issues are presented in the next subsections. The fixed version is available on the repository https://gitlab.inria.fr/mlanvin/crisis2022.

```
16:01:11.009724 IP 192.168.10.50.http > 172.16.0.1.20823: Flags [S.]
16:01:11.009723 IP 172.16.0.1.20823 > 192.168.10.50.http: Flags [S]
16:01:11.023740 IP 172.16.0.1.20823 > 192.168.10.50.http: Flags [.]
16:01:11.023744 IP 172.16.0.1.20823 > 192.168.10.50.http: Flags [P.]
```

Listing 1: Misordered packets from CICIDS2017. The timestamp is the leftmost column, and the flags are the rightmost column. Flags `[S]` mean SYN, `[.]` mean ACK and `[S.]` means SYN-ACK.

3.1 CICFlowMeter issue with misordered packets

CICFlowMeter is a tool that extracts network flow descriptions from pcap files that contain network captures. The network flow descriptions generated by CICFlowMeter are bidirectional and distinguish a source (the machine that initiates the communication) and a destination. As a reminder, the initial TCP handshake consists in exchanging three messages (SYN from source to destination, SYN-ACK from destination to source, and ACK from source to destination) to establish a connection. We identified a remaining flaw in CICFlowMeter that occurs when pcap files are not sorted by timestamps, as it happens in the original dataset of CICIDS2017. In that case, the tool reads the network packets in the order of the network capture files, but the SYN-ACK packet can sometimes be stored before the SYN packet in the pcap file, even though, according to the timestamp, the SYN packet did occur before the SYN-ACK one.

Src IP	Src Port	Dst IP	Dst Port	Protocol	Timestamp
192.168.10.50	80	172.16.0.1	20823	6	07/07/2017 16:01:11

Table 1. Flow description of Listing 1. CICFlowMeter inverted source and destination.

As an example, the Listing 1 illustrates this phenomenon with an extract of one network connection of the pcap files of CICIDS2017. We can observe that the first packet in the network capture is a SYN-ACK even if its timestamp is not the earliest. CICFlowMeter uses the first received packet to infer the source and destination. Therefore, the source and destination are exchanged in the resulting network flow description provided in the CSV files, cf. Table 3.1. Table 2 shows the number of misordered frames in the network capture per day (the number of frames can be considered as very close to the number of packets in our case). The figures show that Wednesday and Friday are the two days with the maximum number of misordered packets, with about twice as many misordered packets as the other days. We know Dos/DDoS attacks are performed on these two days, so our hypothesis is that the packet misordering seems to be related to the kind of attack that is performed. Since a high number of packets characterizes these attacks, there might be a race condition during the packet capture and store.

```
14:48:12.894976 IP 192.168.10.3.88 > 192.168.10.8.49173: Flags [S.]
14:48:12.895030 IP 192.168.10.8.49173 > 192.168.10.3.88: Flags [S]
14:48:12.895032 IP 192.168.10.8.49173 > 192.168.10.3.88: Flags [.]
14:48:12.895095 IP 192.168.10.3.88 > 192.168.10.8.49173: Flags [.]
```

Listing 2: Excerpt of a network connection from CICIDS2017 with ordered packets in the pcap file but with a disordered logic. The timestamp is the leftmost column, and the flags are the rightmost column. Flags `[S]` mean SYN, `[.]` mean ACK and `[S.]` means SYN-ACK

A solution to this misbehaviour is sorting the pcap files before processing them with CICFlowMeter. The tool *reordercap*[6] can perform such an operation. The version of CICFlowMeter[7] proposed by Engelen et al. now includes our patch that verifies the packets' order in the network capture to avoid this issue.

Pcap files	#Misordered frames	#Frames	Proportion(%)
Monday-WorkingHours.pcap	3234	11709971	0.028
Tuesday-WorkingHours.pcap	3721	11551954	0.032
Wednesday-WorkingHours.pcap	12654	13788878	0.092
Thursday-WorkingHours.pcap	3655	9322025	0.039
Friday-WorkingHours.pcap	7094	9997874	0.071

Table 2. Numbers of misordered frames in the different pcap files

3.2 Incoherent timestamps

Another issue we found in the CICIDS2017 network captures is that the timestamps can be incoherent with the protocol. For example, in Listing 2, the packet SYN-ACK has a lower timestamp than the packet SYN, even though, according to the TCP protocol, such configuration should not happen. This produces the inverted flow description of the Table 3 for the same reason as in the previous subsection.

Src IP	Src Port	Dst IP	Dst Port	Protocol	Timestamp
192.168.10.3	88	192.168.10.8	49173	6	06/07/2017 14:48:12

Table 3. Extracted flow description from the misordered packets presented on Listing 2. CICFlowMeter inverted source and destination.

[6] https://www.wireshark.org/docs/man-pages/reordercap.html
[7] https://github.com/GintsEngelen/CICFlowMeter

We have no hypothesis on what produced this issue, and it is difficult to fix automatically. For this reason, we did not fix it. However, such incoherent timestamps can cause CICFlowMeter to invert source and destination because, in this case, it considers the sender of the SYN-ACK packet to be the source.

3.3 Dealing with data duplication

Observing the network captures, we found many duplicated packets in the data. Listing 3 contains an example of such duplicated packets. We can see the repetition of the SYN and RST packets: the time interval between two identical packets is only a few microseconds. Besides, their content is the same.

We cannot be sure of the cause of that phenomenon as we do not have enough detailed information on the network capture. As the time interval between two identical packets is very small and as UDP and ICMP packets are duplicated, we can rule out the hypothesis that this behaviour is normal due to the TCP retransmission mechanism. For now, our main hypothesis is that the port mirroring on the main switch of the CICIDS2017 testbed was not configured correctly. We did not analyze the network capture entirely, but we only saw duplicated packets between the testbed's internal hosts. That could be explained by the fact that all the ports of the switch connected to internal hosts are configured to mirror incoming *and* outcoming packets to the mirror port. That hypothesis is reinforced by the fact that broadcast packets to the internal subnetwork are duplicated 13 times.

We corrected this issue with the tool *editcap*[8] that can find and remove duplicated packets within a given time window. Using this tool with a time window of $500\mu s$, we measured how many packets were duplicated during the whole week of the CICIDS2017 dataset. The Table 5 reports the number of duplicated packets per day. On average, more than 497000 packets are duplicated per day, representing 4.5% of the packets per day. The duplication modifies the network flow description that is extracted by the tool CICFlowMeter. For example, the traffic shown in Listing 3 is transformed by CICFlowMeter into the flow description shown in Table 3.3. Its numbers of forward and backward packets are two because of the duplication, even though only one forward and one backward packets were actually exchanged in the network. As our experiment will show in Section 4, this duplication has serious impacts on the performances of the classifiers.

Src IP	Src Port	Dst IP	Dst Port	Total Fwd Pkts	Total Bwd Pkts
192.168.10.8	3632	192.168.10.9	28316	2	2

Table 4. Excerpt of the flow description of Listing 3

[8] https://www.wireshark.org/docs/man-pages/editcap.html

```
15:45:30.347074 IP 192.168.10.8.distcc > 192.168.10.9.28316: Flags [S],
seq 2582752148, win 8192, options [mss 1460,nop,nop,sackOK], length 0
15:45:30.347078 IP 192.168.10.8.distcc > 192.168.10.9.28316: Flags [S],
seq 2582752148, win 8192, options [mss 1460,nop,nop,sackOK], length 0
15:45:30.347258 IP 192.168.10.9.28316 > 192.168.10.8.distcc: Flags [R.],
seq 0, ack 2582752149, win 0, length 0
15:45:30.347261 IP 192.168.10.9.28316 > 192.168.10.8.distcc: Flags [R.],
seq 0, ack 2582752149, win 0, length 0
```

Listing 3: Example of duplicated network packets CICIDS2017

Day	Number of duplicated packets	Total number of packets	Proportion of duplicated packets
Monday	514,241	11,709,971	4.39%
Tuesday	482,553	11,551,954	4.18%
Wednesday	480,209	13,788,878	3.48%
Thursday	556,013	9,322,025	5.96%
Friday	466,448	9,997,874	4.67%

Table 5. Number and proportion of duplicated packets per day.

3.4 Attack omission: labelling issues and correction

While using CICIDS2017 to evaluate machine learning models, we noticed an excessive number of false positives when processing Thursday's traffic. According to CICIDS2017 documentation, there is an infiltration step where the victim is meant first to download a malicious file or use an infected USB flash memory from 2:19 PM to 3:45 PM and then the infected machine is meant to perform a port scan afterwards. In the original CSV files, only 36 network flows are labelled as part of the infiltration. However, by analyzing the network flows corresponding to our false positives, we found some common network characteristics that led us to find a port scan that was not correctly labelled. We estimate that several tens of thousands of network flows are related to this port scan.

This preliminary experiment was done manually. However, manually labelling tens of thousands of network flows related to these attacks would have been too expensive and prone to error, so we decided to use an automated method. We could have used a rough method and labelled all the traffic between the infected machine and the other machines during the period of the infiltration attack. This is for example the method used by Engelen et al. [4] to label attacks. However, such a method would have introduced labelling errors as there is also benign traffic between the infected machine and the other machines. We estimate that this rough method would label about $7,000$ benign flows as attacks out of the $80,000$ total flows between the infected machine and the others, so close to 10% of wrong labels.We thus decided to deduce from the port scan attack what network characteristics we could use to label the network flows correctly.

With duplication			Without duplication		
Fwd Packet	**Bwd packets**	**Count**	**Fwd Packet**	**Bwd packets**	**Count**
2	0	31436	1	1	38228
2	2	30042	1	0	31138
1	1	13840	5	4	315
1	2	1099	1	6	191

Table 6. Number of emitted and received packets and associated number of network flows for the traffic between 192.168.10.8 and all the machines belonging to the subdomain 192.168.10.0/24 from 2:15 PM to 3:50 PM on Thursday.

We know that the attacks take place on Thursday between 2:15 PM and 3:50 PM and that the infected machine's IP address is 192.168.10.8. Due to some source/destination inversion that is difficult to fix (see Subsection 3.2), we will look into every network flow with this IP address either as the source or destination IP address. Our labelling method is refined by taking into account the number of *forward* (from source to destination) and *backward* (from destination to source) packets characteristics.

There are multiple ways of performing a port scan, but the general idea is that the attacker will probe a port with a packet and deduce from the behaviour of the scanned machine whether the port is open or not. There are two typical situations: the scanned machine either replies by emitting one packet or does not reply, depending on the port's status, the kind of scan and the network configuration. With certain port scan techniques like SYN scan, the attacker expects an answer from the scanned machine to infer the port's status, and with others like Null, FIN, or Xmas scans, an opened port will be revealed by the absence of response[9]. An absence of response from the scanned machine can also be observed if the dedicated firewall or the host-based firewall filters the packet. With these considerations in mind, we can propose patterns to filter the network flows based on the number of forward or backward packets: we expect attacks to have one emitted packet and either one or zero received packets.

The Table 6 presents the top four patterns that gather the maximum number of network flows from and to the victim, either with or without the duplicated packets. On the right part of the table, when there is no duplication, the patterns "1 forward - 1 backward" and "1 forward - 0 backward" are indeed the most frequent patterns, and other patterns are negligible (the patterns "5 forward - 4 backward" and "1 forward - 6 backward" are mostly not related to the port scan according to the manually inspected examples). There are also about a hundred occurrences of the patterns "2 forward - 2 backward" and "2 forward - 0 backward" that we believe are also part of the port scan. There are port scans that use protocol quirks and whose flow is not properly reconstructed by CICFlowMeter. These incorrectly reconstructed flows include ICMP reconnaissance by `Nmap`, ACK scans and UDP scans. So, for the dataset without duplication, we use the

[9] https://nmap.org/book/man-port-scanning-techniques.html

patterns "1 forward - 1 backward", "1 forward - 0 backward" and "2 forward - 2 backward" and "2 forward - 0 backward".

When taking duplicated packets into account (left part of the table), the expected pattern "1 forward - 1 backward" was only the third most frequent pattern. Indeed, the first two patterns, "2 forward - 0 backward" and "2 forward - 2 backward" are the duplicated equivalent of the expected pattern ("1 forward - 1 backward" and "1 forward - 0 backward") but seen twice, i.e., with two emitted and/or received packets instead of just one. This is the consequence of the duplication problem described in Subsection 3.3.

In addition to the duplication phenomenon, sometimes timestamps are shifted between the two observations of the same network flow. When these two effects combine, it can produce some unexpected behaviours where a simple SYN and RST connection is duplicated into SYN - RST - SYN - RST packets, and then transformed into SYN - RST - RST - SYN packets due to timestamp errors. In this case, there are one emitted packet (a SYN packet) and two received packets (two RST packets), so a "1 forward - 2 backward" pattern. The last SYN packet triggers the creation of a new flow having the pattern "1 forward - 0 backward". So, for the dataset with duplication, we use the same patterns as for the dataset with duplication (that are still valid because not all packets are duplicated), and we add the "2 forward - 2 backward", "2 forward - 0 backward", "1 forward - 2 backward" patterns.

Besides, we manually exclude the network flows associated with legitimate protocols (DNS, LDAP, NTP, and NETBIOS-NS) that are present in the network traffic. Our heuristic could have mislabeled them.

To summarize, we labelled every network flow as a port scan attack if the flow happened on Thursday between 2:15 PM and 3:50 PM, if it comes from or to IP 192.168.10.8, if its protocol is neither DNS, LDAP, NTP or NETBIOS-NS, and if its numbers of forward and backward packets respect the patterns mentioned earlier.

We applied this filter and counted the numbers of network flows related to port scan per IP address. We counted about $4,060$ port scans for all six Ubuntu machines, about $5,300$ for the Windows 7 machine, about $6,900$ for the Mac machine, about $8,000$ for the two Windows 10 machines and about $12,000$ for the Window 8.1 machine.The number of ports that are scanned by a default Nmap scan is about one thousand. Since the number of identified flows for half of the machines is around four thousand, our hypothesis is that they are four scan attacks of the network.

4 Assessment of the consequences on intrusion detection models performances

Finding and fixing these issues in CICIDS2017 is a great opportunity to examine the current experimental evaluation performed on this dataset and assess its consequences on the performances. More specifically, we would like to answer the following research questions:

- **Q1**: How does the pcap ordering fix affect the performances of these models?
- **Q2**: How does the correct labelling of the port scan affect the performances of these models?
- **Q3**: How does the flow deduplication affect the performances of these models?
- **Q4**: How could previous articles obtain very high performances with machine learning algorithms in the presence of these issues?

In this section, we first describe our experimental evaluation protocol, evaluate several supervised classification models, namely decision trees, random forest, naive Bayes and support-vector machine classifiers, and finally answer the four research questions we defined.

4.1 Experimental evaluation protocol

To answer the research questions raised previously, we build several train sets and test sets based on the network captures files with different corrections to assess their effect. The corrections include the reordering of the pcap file before applying CICFlowMeter (denoted as **R** for the rest of the article). If the pcap file isn't reordered, then the labels provided by [4] are used. Our second correction is the addition of the new port scan labels denoted as **P**. We will also assess the impact of the duplicated network packets, and we denote the presence of duplicated packets by **D**.

As we described in section 3.1, the labelling script proposed by [4] sets three kinds of labels: "benign", "malicious" and "attempted". To simplify the experiments and the analysis of their results, we decided to label these attempts as benign. We also ran our experiments by labelling them as attacks: the differences were slight and did not impact our conclusions. The train and test set configurations are detailed in Table 7. We distinguish three sets of experiments. The goal of the first set (**RD** and **RPD**) is to observe the effect of the reordering of the pcap file by comparing with **D** and **PD** and answer **Q1**. The second set (**RD** and **RPD**) is built to assess the impact of the port scan addition on the labels provided by Engelen et al. and answer **Q2**. The last set of experiments (**R** and **RP**) consists in assessing the removal of the duplicated packets in the network capture and answering question **Q3**. It must be noted that we think the correct version of the dataset is the one noted **RP**, i.e. with the port scan correctly labelled, the reordering of the packets and without duplication.

We evaluated four supervised models: a naive Bayes classifier, a support-vector machine (SVM), a decision tree and a random forest. We chose these supervised models since they obtained the best performances for intrusion detection given the survey [11]. A benchmark on CICIDS2017 [9] also highlighted the good performances of the decision tree and naive Bayes models on the dataset. We used the scikit-learn library implementation for all these models. We used standard configuration for the different models except for tree-based models, for which we limited their maximum depth to 15 to prevent overfitting.

We adopted a particular strategy for splitting the dataset into train and test sets to conduct a fair study. Indeed, we did not want to put mislabelled flows in

Sets name	Reordered (**R**)	Scan Port added (**P**)	Has Duplicates (**D**)
D			✓
PD		✓	✓
RD	✓		✓
RPD	✓	✓	✓
R	✓		
RP	✓	✓	

Table 7. Experimental configurations depending on capture and labelling corrections

the train set. Otherwise, if the train set were mislabelled, the supervised methods would certainly have poor performances on the test set.

To train the models on consistent data, the train set only includes correctly labelled flows but not any flow that Engelen et al. classified as "attempts". More precisely, we first collected about 65% of each attack, except for the newly discovered port scan attack. This means that the attack types (except the new port scan attack) are present in the training data with about the same proportion as in the dataset. Then, we added as many benign flows as to have a balanced dataset with as many benign examples as malicious examples. This dataset depends on the applied correction, so there are several train sets. However, each one of the three sets of experiments shares a common train set.

From the raw features provided by CICFlowMeter we drop features with constant values: `Fwd URG Flags`, `Bwd URG Flags`, `URG Flag Count`. We also drop `Flow ID` that is unique to each flow. Besides, we drop `Src IP`, `Dst IP`, `Timestamp` since they could make the models learn shortcuts such as identifying malicious IP addresses or attack campaign periods. The `Src Port` feature is removed since it is random and could lead to overfitting. Flow descriptions containing NaN values are dropped. Finally, we normalize the data by centring and scaling each feature as it is necessary for SVM.

To assess the impact of the labelling corrections, we rely on classic metrics used in intrusion detection: the numbers of True Positives (attacks correctly detected), True Negatives (benign traffic correctly identified), False Positives (alarms caused by benign traffic) and False Negative (attacks not detected). More specifically, we use the True Positive Rate (TPR, or recall), which is the proportion of attacks that are correctly detected, and the False Positive Rate (FPR), which is the proportion of benign traffic that generates false alarms. We could not use the classic area under the ROC (Receiver Operating Characteristic) curve as it is not easily obtainable on ordinal models such as decision trees.

4.2 Experiments results

The Figures 1 and 2 show respectively the TPR and the FPR obtained by the models for the different labelling corrections.

***Q1**: How does the pcap ordering fix affect the performances of these models?* This experiment relies on train and test sets **D** and **RD**, as well as **PD** and

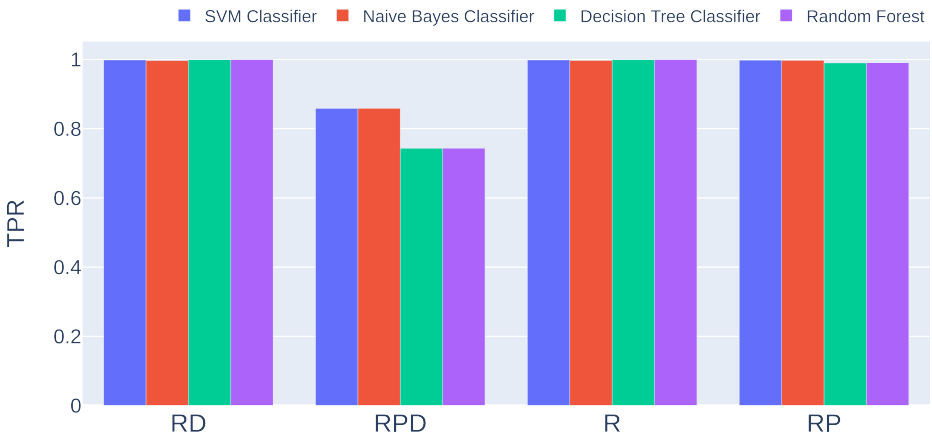

Fig. 1. TPR of the different supervised models given the different experimental configurations.

RPD. The reordering affects the creation or deletion of only about four thousand network flows, which is less than 0.2% of the total count for the whole week. We also measured a bit more than four thousand network flows with the source and destination that are swapped after reordering. This also represents about 0.2%, which is only a small percentage of the total number of network flows. Reordering the pcap files fixes the source/destination inversion issue and produces more accurate labelling. With the reordering, all the metrics are slightly better, but the differences are so slight (the mean change on all metrics is less than 0.1%) that we decided not to include them on the charts for the sake of brevity.

***Q2**: How does the correct labelling of the port scan affect the performances of these models?* This experiment relies on train and test sets **RD** (no port scan label) and **RPD** (with port scan label). We can consider two cases, depending on the labels of the considered dataset used as the ground truth for computing the metrics. With **RD** labels, the port scan attack flows that are detected are counted as false positives because the port scan is not labelled (even if the port scan is really present in the network data). With **RPD** labels, the port scan attack flows that are detected are counted as true positives because the port scan is labelled. Once the port scan attack is correctly labelled, we observe in the Figure 1 that the tested supervised models lose, on average, about 20% of recall. It means that these models do not correctly detect a vast proportion of the newly labelled attack. Since the FPR from all models is reduced by about 3% for all models after correctly labelling this attack, we can understand that these models detected at least some part of the port scan.

The drop of the recall is very surprising because there is already a correctly labelled port scan attack in the train set, so the models should be able to detect this new attack correctly. This question is discussed along with **Q3**.

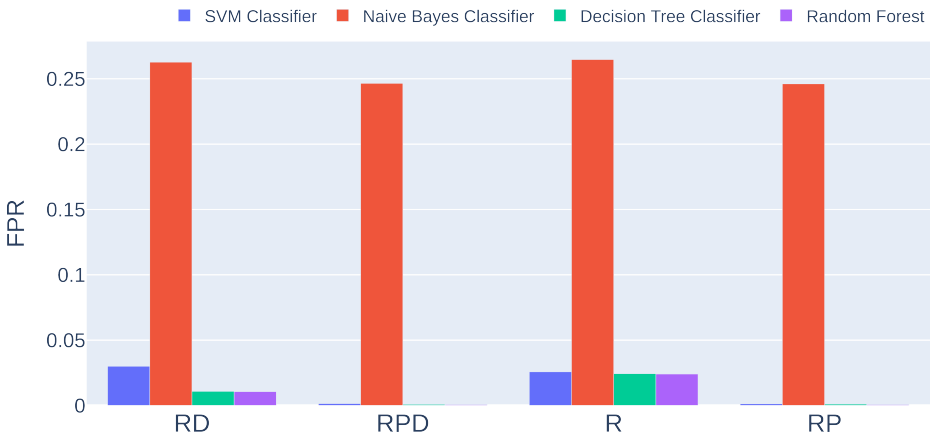

Fig. 2. FPR of the different supervised models given the different experimental configurations.

***Q3**: How do the duplicated packets affect the performances of these models?* This experiment relies on train and test sets **RD** and **R**, as well as **PRD** and **RP**. The Figure 1 shows that without duplication, the models are able to detect the new relabelled port scan attack correctly: we can see that the recall goes from about 80% on **RPD** to close to 100% on **RP**. In other words, the duplication issue prevented the models from identifying this attack correctly. The duplication has little effect on other attack detection: the recall is about the same between **RD** and **R**. The TPR change a little (upward or downward depending on the model) between **RD** and **R**, but it seems difficult to draw any conclusion.

To explain that effect on the recall, we analysed the explanation of the decision tree prediction for the two port scan attacks with and without duplication. The decision paths are similar, with some differences on the decision related to the following features: `Flow IAT Min`, `Flow IAT Mean`, `Bwd Packets/s` and `Flow Packets/s`. The duplication directly impacts these features because, as we saw in Subsection 3.3, the duplication causes two packets in both ways instead of having one emitted packet and one received packet. Therefore the backward number of packets per second is doubled as well as the number of packets per second. For the two other features, the Inter-Arrival Time (IAT) is modified since the duplication makes duplicated packets very close in time. This reduces a lot the minimum and the mean IAT values. So, the duplication of the packets disturbs significantly the flows extracted by CICFlowMeter.

So, with duplication, the models do not detect the port scan attack correctly because it mostly consists of flows with duplicated packets that do not match the behaviours learned on the other correctly labelled port scan, which do not have duplicated packets. Without duplication, the newly labelled scan port matches the learned attack behaviour.

***Q4**: How could previous articles obtain very high performances with machine learning algorithms in presence of these issues?*

Previous experiments use the same scenario as our dataset **RD**. As we can see, we can achieve very high recalls with classic models without fine-tuning. Besides, except for the Naive Bayes model, all of them have a relatively small FPR. As we can see on Figure 2, part of these false positives are related to the newly labelled scan attack. For this reason, results such as those obtained by [9] look like overfitting. Overfitting is also one of the conclusions of the survey [7] on intrusion detection models used on CSE-CIC-IDS2018. For the recall, we observe that the results on the correct dataset **RP** are as good as on the original dataset **RD**, while the recall drops when we label correctly the missing port scan (dataset **RPD**). We also observe that the FPR raises for the dataset **R** compared to **RD**. It seems that two issues we found (the missing port scan and the duplication of packets) offset each other and allow the ML models to obtain good results on the original dataset **RD**.

5 Conclusion

CICIDS2017 is often used to evaluate the performances of NIDS. However, it has several flaws in both its traffic captures and its labelling. First, the tool CICFlowMeter misbehaves when packets are misordered in the traffic capture, which leads to source/destination inversion and wrong feature values in the network flows. Second, the traffic capture contains incoherent timestamps. Third, about 5% of the packets are duplicated, modifying the network characteristics of the flows used for detection. Finally, the infiltration step contains a port scan attack that was not labelled as such either in the original authors' CSV files or in the revision of the labels provided by [4]. Once the dataset was fixed and relabelled according to these modifications, we measured the intrusion detection performances of several supervised models that are often used. The newly labelled port scan attack is not entirely detected by the models, leading to a loss of recall of about 20%. However, we get back to good performances on the new port scan attack by removing duplicated packets. We allow us to conclude that previous work could not obtain a precision close to 100% without overfitting. Finally, the security community could benefit from merging our corrections with the features calculations and protocol handling of [13].

Through this article, we highlighted the importance of the quality of the network intrusion detection datasets to evaluate the NIDS accurately. A good quality comes from a clean labelling process, and an accurate network captures management to prevent packet duplication, for instance. Without these prerequisites, the evaluation is distorted, and the models learnt on the dataset may be unfit for realistic network characteristics.

We would advise dataset authors to provide as many details as possible on their labelling strategy, how they perform attacks, their network infrastructure and post-processing steps on the provided data. For future work, we want to improve detection explainability to understand false positives better and help

analyse the corresponding alarms. This would allow to detect such data and labelling mistakes more easily.

References

1. Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)
2. Cortes, C., Vapnik, V.: Support-vector networks. Machine learning **20**(3), 273–297 (1995)
3. CSE-CIC: A realistic cyber defense dataset (CSE-CIC-IDS2018) (2018), `https://registry.opendata.aws/cse-cic-ids2018`
4. Engelen, G., Rimmer, V., Joosen, W.: Troubleshooting an intrusion detection dataset: the CICIDS2017 case study. In: SPW. pp. 7–12 (2021). `https://doi.org/10.1109/SPW53761.2021.00009`
5. Kumar, V., Das, A.K., Sinha, D.: Statistical analysis of the unsw-nb15 dataset for intrusion detection. In: Computational intelligence in pattern recognition, pp. 279–294. Springer (2020)
6. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436–444 (2015)
7. Leevy, J., Khoshgoftaar, T.: A survey and analysis of intrusion detection models based on cse-cic-ids2018 big data. Journal of Big Data **7** (11 2020). `https://doi.org/10.1186/s40537-020-00382-x`
8. Lippmann, R., Fried, D., Graf, I., Haines, J., Kendall, K., McClung, D., Weber, D., Webster, S., Wyschogrod, D., Cunningham, R., Zissman, M.: Evaluating intrusion detection systems: the 1998 darpa off-line intrusion detection evaluation. In: Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00. vol. 2, pp. 12–26 vol.2 (2000)
9. Maseer, Z.K., Yusof, R., Bahaman, N., Mostafa, S.A., Foozy, C.F.M.: Benchmarking of machine learning for anomaly based intrusion detection systems in the cicids2017 dataset. IEEE Access **9**, 22351–22370 (2021)
10. Moustafa, N., Slay, J.: UNSW-NB15: a comprehensive data set for network intrusion detection systems. In: MilCIS. pp. 1–6 (2015). `https://doi.org/10.1109/MilCIS.2015.7348942`
11. Panigrahi, R., Borah, S., Bhoi, A.K., Ijaz, M.F., Pramanik, M., Jhaveri, R.H., Chowdhary, C.L.: Performance assessment of supervised classifiers for designing intrusion detection systems: a comprehensive review and recommendations for future research. Mathematics **9**(6), 690 (2021)
12. Ring, M., Wunderlich, S., Scheuring, D., Landes, D., Hotho, A.: A survey of network-based intrusion detection data sets. Computers & Security **86**, 147–167 (2019)
13. Rosay, A., Cheval, E., Carlier, F., Leroux, P.: Network intrusion detection: A comprehensive analysis of cic-ids2017. In: ICISSP (2022)
14. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: ICISSP (2018)
15. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the kdd cup 99 data set. 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications pp. 1–6 (2009)
16. Wang, Y., Yang, K., Jing, X., Jin, H.L.: Problems of kdd cup 99 dataset existed and data preprocessing. In: Applied Mechanics and Materials. vol. 667, pp. 218–225. Trans Tech Publ (2014)