Network Intrusion Detection: A Comprehensive Analysis of CIC-IDS2017

Arnaud Rosay[1][a], Eloïse Cheval[2][b], Florent Carlier[3][c] and Pascal Leroux[3][d]
[1]*STMicroelectronics, Rue Pierre-Félix Delarue, Le Mans, France*
[2]*Polytech Nantes, Nantes University, Rue Christian Pauc, Nantes, France*
[3]*CREN, Le Mans University, Avenue Olivier Messiaen, Le Mans, France*

Keywords: Network Intrusion Detection, CIC-IDS2017, CSE-CIC-IDS2018, CICFlowMeter, LycoSTand, LYCOS-IDS2017, Machine Learning.

Abstract: With an ever increasing number of connected devices, network intrusion detection is more important than ever. Over the past few decades, several datasets were created to address this security issue. Analysis of older datasets, such as KDD-Cup99 and NSL-KDD, uncovered problems, paving the way for newer datasets that solved the identified issues. Among the recent datasets for network intrusion detection, CIC-IDS2017 is now widely used. It presents the advantage of being available as raw data and as flow-based features in CSV files. In this paper, we analyze this dataset in detail and report several problems we discovered in the flows extracted from the network packets. To address these issues, we propose a new feature extraction tool called LycoSTand, available as open source. We create LYCOS-IDS2017 dataset by extracting features from CIC-IDS2017 raw data files. The performance comparison between the original and the new datasets shows significant improvements for all machine learning algorithms we tested. Beyond the improvements on CIC-IDS2017, we discuss other datasets that are affected by the same problems and for which LycoSTand could be used to generate improved network intrusion detection datasets.

1 INTRODUCTION

We live in a world where interactions with connected devices are ubiquitous. Computers, smartphones, IoT, connected cars are used every day. But this connected world is under threat from cyberattacks making network security more important than ever. The impacts of computer network intrusions can be severe. This is typically the case when healthcare infrastructures are blocked or when hackers remotely take control of a connected vehicle.

Such cyberattacks can be prevented by the use of Network Intrusion Detection Systems (NIDS). To study network intrusion detection, KDD-Cup99 (Lee et al., 1999) was one of the earliest publicly available datasets, released in 1999, and has been widely used. An initial review of this dataset (McHugh, 2000) has put in light some issues that were further analyzed (Tavallaee et al., 2009). A derived dataset referred to as NSL-KDD addresses some of the shortcomings of KDD-Cup99. Subsequently, several datasets were published by the Canadian Institute for Cybersecurity (CIC), University of New Brunswick, Canada, such as CIC-IDS2017 (Canadian Institute for Cybersecurity, 2017b), CSE-CIC-IDS2018 and CIC-DDOS2019 (Sharafaldin et al., 2019). These datasets present the advantage of being available in two forms: a record of Ethernet frames in Packet CAPture (PCAP) files and a list of network flow characteristics extracted from the Ethernet frames in CSV files.

Our main contributions consist of four parts. First, we present an in-depth analysis of CIC-IDS2017 showing some inherent issues in its CSV files. Five classes of problems were identified: feature duplication, feature calculation errors, erroneous protocol detection, inconsistent TCP termination flows and doubts on labels. Second, since this dataset is not provided with the tool to put labels, we propose a new network flow extractor called LycoSTand working with a labelling solution specific to this dataset. LycoSTand solves all the issues discovered during

[a] https://orcid.org/0000-0001-5937-5331
[b] https://orcid.org/0000-0003-2961-4973
[c] https://orcid.org/0000-0003-0314-3667
[d] https://orcid.org/0000-0002-4447-7244

Rosay, A., Cheval, E., Carlier, F. and Leroux, P.
Network Intrusion Detection: A Comprehensive Analysis of CIC-IDS2017.
DOI: 10.5220/0010774000003120
In *Proceedings of the 8th International Conference on Information Systems Security and Privacy (ICISSP 2022)*, pages 25-36
ISBN: 978-989-758-553-1; ISSN: 2184-4356

our analysis. CIC-IDS2017 PCAP files were processed with our tools to generate a new dataset called LYCOS-IDS2017. The tools and the resulting dataset are publicly available: http://lycos-ids.univ-lemans.fr/. Then, Machine Learning (ML) algorithms were used to measure performance of both datasets. We show that the corrections provided by LycoSTand improve standard metrics for all the tested algorithms. At last, we considered four other recent datasets based on the same features extractor as CIC-IDS2017. We found several artifacts in each showing that they suffer from the problems identified on CIC-IDS2017. All these datasets could very likely be enhanced by using LycoSTand to extract features from available PCAP files.

This paper presents in Section 2 works related to IDS. A comprehensive analysis of CIC-IDS2017 is provided in Section 3. Section 4 presents our tools used to create LYCOS-IDS2017, a corrected dataset detailed in 5. The performance impacts of all the correction we brought are provided in Section 6. A short analysis in Section 7 shows that several other CIC datasets are affected by the issues identified in this paper and could benefit from feature extraction with LycoSTand. Finally, Section 8 concludes this paper and identifies ideas for future work.

2 RELATED WORK

With the rise of computer networks and the Internet, many research have been conducted on network intrusion detection using different datasets. KDD-Cup dataset has been used since 1999 and still appears in recent publications. However, it presents several issues (McHugh, 2000). To enable researchers to investigate new IDS models, a part of its shortcomings were solved in a new dataset called NSL-KDD (Tavallaee et al., 2009). This dataset contains selected records from its predecessor to remove redundant records and to better balance the size of classes by difficulty level. Although it does not address all problems of KDD-Cup99, NSL-KDD is a better dataset.

Several other IDS datasets have been created. University of Twente proposed a dataset (Sperotto et al., 2009) aiming to be representative of real traffic and complete from a labelling perspective. UNSW-NB15 (Moustafa and Slay, 2015) is a simulation-generated IDS dataset containing modern attacks grouped into nine different attack families. Although the tool generating the traffic is based on information about real attacks listed in security vulnerability website (CVE), the simulation may not perfectly reflect the network traffic generated by the tools used by hackers.

Many IDS datasets were reviewed and evaluated against 11 criteria (Gharib et al., 2016). These characteristics were used to create CIC-IDS2017 (Sharafaldin et al., 2018), a new dataset released by the Canadian Institute for Cyber Security (CIC) at the University of New Brunswick. This dataset contains traffic recorded for 5 days, including normal traffic and attacks launched by machines running Kali Linux with its set of penetration testing toolset. CIC-IDS2017 is now widely used to study performance of machine learning algorithms in network intrusion detection (Gamage and Samarabandu, 2020; Yang et al., 2021; Ho et al., 2021; Maseer et al., 2021).

Although CIC-IDS2017 was intended to address the issues of previous IDS datasets, a very limited number of study have been conducted to identify potential problems. A detailed analysis of this dataset (Panigrahi and Borah, 2018) revealed four shortcomings of different importance. CIC-IDS2017 data is scattered in many files, meaning that users have to aggregate all files to use them. This generates a dataset that is huge and heavy to process. These first two problems are rather common in machine learning and do not affect performances of intrusion detection. It is also reported that many samples have missing values and that classes are highly imbalanced. Another study (Rosay et al., 2021) identified that some features extracted from PCAP files are poorly computed. One can expect these issues to impact the performance of ML algorithms. Other issues covering several aspect were found (Engelen et al., 2021): attack simulation, feature extraction, flow construction and labelling.

It should be noted that a vast majority of publications rely on the CSV files provided by CIC to study the performance of machine learning algorithms. Based on the problems mentioned above, it is natural to wonder whether the findings remain valid if the CSV are badly generated.

We propose to go further in the analysis to provide more insights about the flow construction issue, to highlight more problems affecting CIC-IDS2017 and to provide our solution as open source software to generate a corrected dataset from the raw data available in the PCAP files of CIC-IDS2017 and some other dataset released by the CIC. In addition, we propose a benchmark of machine learning algorithms to compare CIC-IDS2017 and LYCOS-IDS2017.

3 COMPREHENSIVE ANALYSIS

A first step of the analysis consists of understanding how the dataset was created and its content. Then,

in an attempt to reproduce the flow-based features of CIC-IDS2017 from the PCAP files, we discovered several problems both in the original CSV files and CICFlowMeter (Canadian Institute for Cybersecurity, 2017a), the tool generating the features.

3.1 CIC-IDS2017 Description

The dataset was recorded on a real network whose infrastructure is divided into two parts. The first one contains 4 machines launching the attacks towards the second one containing 10 victim machines. 50 GByte of raw data in PCAP files are provided together with a set of 84 features in CSV files. Network traffic was recorded for 5 days, resulting in a total of 2,830,743 instances.

The network traffic is classified into 15 classes. One of them corresponds to normal traffic, while the others 14 are different attacks.

Features were extracted from the PCAP files with a tool called CICFlowMeter (formerly ISCXFlowMeter) (Canadian Institute for Cybersecurity, 2017a; Draper-Gil. et al., 2016; Lashkari et al., 2017). The source code of this tool is available. The tool can generate up to 84 features related to bidirectional flows. Forward and backward directions are defined by the source and destination of the first packet of a flow. Flows are identified by a source IP and port, a destination IP and port, and a protocol forming the FlowID. Some characteristics such as packet length or inter-arrival time generate statistical features: minimum, maximum, mean, standard deviation and variance values in both directions. In a Transmission Control Protocol (TCP) connection, synchronization (SYN), acknowledgment (ACK), finish (FIN) or reset (RST) flags are used to indicate the status of the connection. Some other flags like push (PSH) or urgent (URG) provide additional useful information. Statistics on these flags are used as features of the dataset. Flow duration is measured between SYN and FIN flags for most TCP connections. In case of User Datagram Protocol (UDP), the end of a flow is a timeout, as for some TCP communications. The exhaustive list of features generated by CICFlowMeter is presented in (Rosay et al., 2021).

3.2 Discovered Issues

The CICFlowMeter source code has evolved over time without a clear version tag, making it impossible to know the exact version to use to reproduce the features extracted from the dataset. However, all available versions of the tool generate files with a suffix "_Flow.csv" while the CSV files of the dataset were generated with a different suffix "_ISCX.csv". This suffix corresponds to an earlier version of the tool that is not available in the source code. Therefore, it is impossible to reproduce the CSV files from the provided PCAP files.

Comparisons between the CSV files generated by CICFlowMeter, hereafter referred to as CFM files, and the original CSV files, hereafter referred to as ISCX files, revealed some discrepancies. We found that some issues appear in both CFM and ISCX files, while others appear only in one of them. They are classified into five categories: feature duplication, feature miscalculations, wrong protocol detection, inadequate TCP session termination and labelling issue.

3.2.1 Feature Duplication

We identified 4 duplicated features in ISCX files. The first duplicated feature is the average packet length per flow which is calculated for both features: "Average Packet Size" and "Packet Length Mean". Then, the average forward packet length in the flow "Fwd Packet Length Mean" which is the same as what is called segment in "Fwd Segment Size Avg". The same thing is observed in the opposite direction with "Bwd Packet Length Mean" and "Bwd Segment Size Avg". The fourth duplicated feature in ISCX files, corrected in CICFlowMeter code, is the "Fwd Header Length" with "Fwd Header Length.1". Since the code used to generate ISCX files is not available, we will only explain the remaining miscalculations in CFM files using CICFlowMeter code.

3.2.2 Feature Miscalculations

We found 34 miscalculated features in ISCX files. Most likely due to updates in the CICFlowMeter code, some features were corrected in the latest version of the tool (downloaded on 03-23-2021) on which we based our study. We found 23 remaining mistakes in feature calculation, this number lowers to 21 if we keep only the non-duplicated features.

- The most common mistake is due to a type issue in calculation (integer division instead of floating-point division). This affects 11 features: 6 related to bulks, 4 related to subflows and 1 to downlink/uplink ratio.
- For each new packet, backward bulk features are updated, independently from the packet direction. The same way, forward bulk features are never updated. This leads to erroneous values for all 6 bulk-related features.
- The subflow count is updated depending on a test on timestamp values. Due to misplaced parenthe-

sis, the test is always true and so the subflow count is increased for each new packet received.

- The first packet of each flow is used twice to update the packet length features, this impacts 3 features which are the mean, standard deviation and total packet length.
- TCP-related features contain errors. The SYN and PSH flag counts are inverted, as are the FIN and URG flag counts. Both forward and backward counts for PSH and URG are never updated. The initial TCP window size in backward direction is updated with each packet received and therefore contains the last value instead of the initial one.

3.2.3 Wrong Protocol Detection

CICFlowMeter processes packets according to the protocol that is detected. Unfortunately, the tool sometimes fails to detect it. We observed two types of protocol detection issues due to poor analysis of the Ethernet frames.

- A first level of detection should depend on the Ethertype field of the Ethernet header. CICFlowMeter assigns all packets in the PCAP file to IPv4 or IPv6. Instead of using the Ethertype field, the tool checks whether the packet contains an IPv4 or IPv6 header. If a sequence of bytes in the payload looks like an IPv4 or IPv6 header, the packet is classified accordingly, whatever the Ethertype field. An analysis of the PCAP file contents shows that Address Resolution Protocol (ARP), Link Layer Discovery Protocol (LLDP) and Cisco Discovery Protocol (CDP) packets are present and misclassified. To illustrate this, some ARP packets are interpreted as IPv4 packets with IP addresses 8.0.6.4 and 8.6.0.1 corresponding to values of the ARP header fields.
- A second level of analysis should be based on the protocol field of the IPv4 header. Similarly to IPv4 or IPv6 detection, CICFlowMeter checks whether the packet contains a TCP or UDP header. A sequence of bytes in the payload can be misinterpreted as a TCP or UDP header. If the tool does not detect TCP or UDP, the packet is classified in another group that does not correspond to any IPv4 protocol. The analysis shows that the PCAP files contain Internet Control Message Protocol (ICMP), Internet Group Management Protocol (IGMP) and Stream Control Transmission Protocol (SCTP) packets. TCP and UDP packets are classified according to IANA protocol numbers while most other packets are associated with a '0' protocol. We observed that CICFlowMeter classifies ICMP packets sometimes as TCP, sometimes as UDP and sometimes as neither. Another side effect appears with fragmented frames. UDP fragments are not recognized as UDP and are treated like other protocols with a '0' tag, except for the first packet that contains the UDP header.

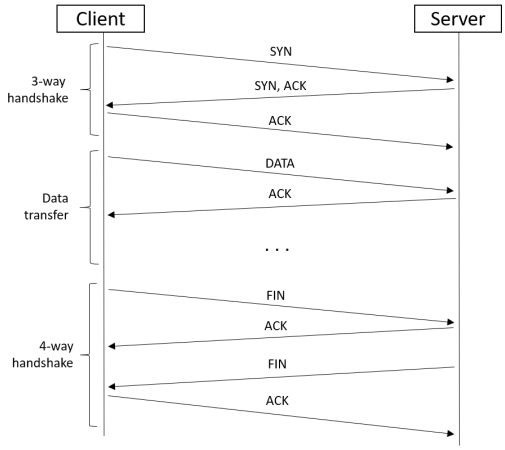

Figure 1: TCP sequence diagram.

3.2.4 Inconsistent TCP Termination Flows

In CICFlowMeter, flows are closed either when the duration exceeds a predefined timeout of 120s, or if the received packet carries a 'FIN' flag. TCP is a connection-oriented protocol with handshakes. A typical example is depicted in Figure 1.

When a new TCP communication is started, a first flow is created. It will contain all the packets from the 3-way handshake and the data transfer. This flow is closed when the first packet of the final 4-way handshake is received. A second flow is created containing the second and third packets of this handshake. Finally, the last packet will create a third flow. The TCP state machine is complex and many other session termination cases can occur. This behavior causes three problems:

- If no other communication using the same addresses/ports is started, this third flow will be closed by timeout. Otherwise, the last packet of the flow will be aggregated with those of the new communication.
- Since the flow-based features are similar for four-way handshakes under normal and attack traffic, they may negatively impact the performance of machine learning depending on the label attached to the second and third flows. In addition, we observed inconsistency in the attack labels in ISCX files: some second and third flows are labelled as benign while others are labelled as attacks, creat-

ing even more confusion for ML algorithms.

- If a TCP communication is aborted by a 'RST' flag, CICFlowMeter does not consider it as closing the flow. Any further communication with the same identifier will be merged into the flow that should have been closed by the 'RST' flag.

3.2.5 Doubts About Labelling

A comparison between the information available on the website presenting CIC-IDS2017 data (Canadian Institute for Cybersecurity, 2017b) and the dataset content revealed some discrepancies, raising doubts about the correctness of the dataset labelling. This point can be illustrated by portscan attacks.

- The website mentioned that portscan attacks occur with and without firewall enabled while all portscan attacks in ISCX use the firewall IP address (172.16.0.1).
- Some flows using the attackers IP address when the firewall is disabled (205.174.165.73) are labelled as benign. These flows were analyzed with Wireshark and the traffic pattern is very similar to the portscan attack: many short flows on many different ports and terminating with a TCP reset. This seems to confirm that the information on the website is correct but that some of the labels are wrong.

A CIC-IDS2017 author confirmed that if there are discrepancies between the website and the ISCX labels, priority should be given to the labels. He also confirmed that the labelling was achieved using a proprietary package that is not publicly available. Although the patterns in PCAP files clearly show that some flows labelled as benign correspond to portscan behaviour, it is not possible to formally confirm the labelling issue.

In summary, the aggregation of packets to create flows is incorrect due to poor protocol detection. Many features are miscalculated. Improper termination of TCP sessions creates similar flows with different labels. These major problems impact the performance of machine learning algorithms that are commonly used for network intrusion detection.

4 LycoSTand: A NEW FLOW-BASED EXTRACTOR

Given all the problems to be resolved and the lack of a clear baseline for CICFlowMeter, we decided to create and publicly release a C program called LycoSTand so that it could be used to reproduce our results or to process PCAP files from other datasets as an alternative to CICFlowMeter. It relies on the libpcap library to parse Ethernet packets. This tool was developed and validated on Ubuntu 18.04, using libpcap-dev version 1.8.1-6ubuntu1.18.04.2 and gcc compiler version 7.5.0.

This program takes the PCAP files, analyzes the packets and calculates 82 flow-based features. We mainly reused the features from CIC-IDS2017 except for the miscalculations mentioned in Section 3.2.2. It creates one CSV file containing all the features for each PCAP file. The complete list of features is described at http://lycos-ids.univ-lemans.fr/.

A major difference between LycoSTand and CICFlowMeter is the handling of TCP session terminations. UDP flows are exclusively closed with a timeout of 120s (configurable in LycoSTand). For TCP, we decided to close flows as soon as a packet with the 'RST' or 'FIN' flag is received. In such a case, the flowID is stored in a 'TCP_terminated' list. When a TCP packet is received, we check if its flowID is in this list. If it is in the TCP_terminated list, packets belonging to the TCP session termination are dropped until a packet with a 'SYN' flag is received, indicating that a new flow is starting. The 'TCP_terminated' list is cleaned up with each new packet by comparing the timestamp difference with the maximum duration of a TCP session termination. The packet processing is described in Algorithms 1 and 2.

When a flow begins, all its features are initialized to 0. They are updated when a subsequent packet belonging to the same flow is received. The computation of the running statistics is based on the incremental technique originally proposed by Welford (Welford, 1962).

The protocol detection is based on the Ethertype extracted from the Ethernet header and the IP protocol field extracted from the IPv4 header. This solves the issues described in Section 3.2.3.

The current version of LycoSTand is publicly available at http://lycos-ids.univ-lemans.fr/. It generates flows from IPv4 packets but it could be extended to process other packet types such as IPv6, ARP or LLDP. Another limitation of our tool is related to runtime. Our focus was on creating reliable datasets, not on being time-optimized. Attacks such as portscan or DoS/DDoS create many flows in a short period of time, and due to the use of singly linked lists, the amount of time it takes to search through the lists is in $O(n)$, where n is the number of ongoing flows. We ran LycoSTand on a laptop based on an Intel Core i7-8750H processor with 16GByte of RAM and the analysis of CIC-IDS2017 PCAP files took about 16 hours. A future version of the tool could use other

Algorithm 1: Packet processing.

```
while True do
    wait for next packet
    clean TCP terminated list
    extract flow identifiers from received packet
    search flow in list of flows
    if IPv4 packet then
        if flow is not in list of flows then
            if TCP packet then
                if flow is not in TCP terminated
                  list then
                    if packet hasn't 'FIN' nor
                      'RST' flag then
                        create new flow
                        calculate its initial
                          statistics
                    else
                        add ID to TCP
                          terminated list
                else
                    if packet has 'SYN' flag
                      then
                        create new flow
                        calculate its initial
                          statistics
                        remove ID from TCP
                          terminated list
            else
                create new flow
                calculate its initial statistics
        else
            if flow not terminated then
                update running statistics
            else
                terminateFlow(packet, flow)
```

Algorithm 2: Flow termination.

```
Procedure terminateFlow(packet, flow)
    close existing flow
    if flow ended by timeout then
        if TCP packet then
            if packet has 'FIN' or 'RST' flag
              then
                add ID to TCP terminated list if
                  it is not there yet
            else
                create new flow
                calculate its initial statistics
        else
            create new flow
            calculate its initial statistics
    else
        add ID to TCP terminated list if it is not
          there yet
    return
```

data structures such as the hashmap to reach an amortized research duration in $O(1)$.

5 LycoS-IDS2017: A CORRECTED DATASET

To confirm the corrections made on the feature extractor have an impact on machine learning, we created a new dataset by processing the PCAP files from CIC-IDS2017 with LycoSTand. Since the original dataset was labelled with a proprietary package, we developed our own solution. This program is written in Python. Our labelling solution relies on the traffic analysis we carried out with Wireshark, which revealed that labels can be assigned using source and destination addresses and ports coupled with timestamps. We did not use time information provided on CIC website but adjusted time according to the traffic analysis.

The resulting dataset and program are publicly available at http://lycos-ids.univ-lemans.fr/. We use different techniques to attach a label to each flow:

- The reference (Canadian Institute for Cybersecurity, 2017b) provides information about the time period and the addresses or ports involved in the various attacks.
- We analyzed the PCAP files with Wireshark to confirm the labels. By observing the attack patterns, we were able to adjust the duration of the attacks.
- We found out that a lot of flows recorded on Thursday afternoon were labelled as benign in CIC-IDS2017, while the pattern is most likely a portscan attack. As it corresponds to infiltration attacks, this behavior can be explained by the fact that this attack is composed of two phases. As explained in (Sharafaldin et al., 2018), a file containing a vulnerability is downloaded and then, the attacker runs a portscan attack. We believe that CIC-IDS2017 authors labelled the second phase as benign traffic. In the absence of detailed information from the authors, it is impossible to identify with absolute certainty which flows are truly benign traffic and which should have been

Table 1: CIC-IDS2017 traffic instances.

Traffic	CIC-IDS2017	LYCOS-IDS2017
BENIGN	2,273,097	1,395,659
Bot	1,966	735
DDoS	128,027	95,683
DoS GoldenEye	10,293	6,765
DoS Hulk	231,073	158,988
DoS Slowhttptest	5,499	4,866
DoS Slowloris	5,796	5,674
FTP-PATATOR	7,938	4,003
Heartbleed	11	11
Infiltration	36	0
PortScan	158,930	160,106
SSH-PATATOR	5,897	2,959
WebAttack BruteForce	1,507	1,360
WebAttack SQL Injection	21	12
WebAttack XSS	652	661
Total	2,830,743	1,837,500

labelled infiltration. To avoid errors in our labels, we decided to drop all packets from Thursday afternoon. Consequently, LYCOS-IDS2017 does not contain any infiltration attacks.

The list of attacks and the number of instances for each dataset is presented in Table 1. We observe that almost every label has less instances in LYCOS-IDS2017 than in CIC-IDS2017. It is consistent to have less instances for the majority of labels basically because of the TCP session terminations discarded.

For Portscan, there is a certain amount of TCP sessions that end with a 'RST' flag and then another one begins with a 'SYN' flag for the same flowID, this behavior leads to more flows in the Lycos file than in the ISCX file.

An uncertainty remains for our labelling program in one specific case. The number of WebAttack XSS in Lycos file is surprisingly high compared to ISCX. We suspect that CIC-IDS2017 contains some benign traffic using the same IP addresses and ports as the attack. Here again, we regret not having access to the original labelling package. Despite doubts about our labels for this attack, we decided to keep it in the dataset. By publishing our tools, we hope that more people can investigate and potentially improve our work.

6 PERFORMANCE IMPACTS FOR INTRUSION DETECTION

As explained in Section 5, LYCOS-IDS2017 is created from the CIC-IDS2017 PCAP files to solve the identified issues. In this section, we define an experimental setup to compare performance of both datasets with several ML algorithms. The creation of training, cross-validation and test sets is explained in Section 6.1. In a second step, we detail the ML algorithms in Section 6.3. Then, the metrics used to evaluate the performance are developed in Section 6.3. Finally, the performance comparison between CIC-IDS2017 and LYCOS-IDS2017 is presented in Section 6.4.

6.1 Data Preparation

This section presents the different steps to prepare the data injected as input to our model. After data cleaning, flows from different classes are sampled in training, cross-validation and test sets. Then, a feature selection is applied before a standardization step.

6.1.1 Data Clean-up

CIC-IDS2017 contains more than 280,000 instances with all empty features. These unwanted samples were removed. A negligible number of instances containing 'NaN' or 'Infinity' were dropped. Due to the miscalculation of features, some features are always null. This is the case for 'Bwd PSH Flags', 'Bwd URG Flags', 'Fwd Avg Bytes/Bulk', 'Fwd Avg Packets/Bulk', 'Fwd Avg Bulk Rate', 'Bwd Avg Bytes/Bulk', 'Bwd Avg Packets/Bulk', 'Bwd Avg Bulk Rate'. Therefore, we decided to remove these features. Finally, we removed infiltration because this attack is not available in LYCOS-IDS2017.

For LYCOS-IDS2017, we only removed the three URG flag related features that are always null as all the other features deleted from CIC-IDS2017 were due to calculation errors and they become informative in LYCOS-IDS2017. A cross-check in the PCAP files confirmed that URG flag is not set in any packet.

Time information in a dataset can be useful for concept drift analysis but knowing that the Ethernet frames were recorded for 5 consecutive days, this dataset is not suitable for studying concept drift. In addition, since we do not want the model to learn when an attack occurs, the 'Timestamp' feature cannot be considered informative.

Each instance is characterized by its source and destination ports and IP addresses, its protocol and all these information gathered in a flowID. As the flow identifier is redundant with other features, it was removed from the dataset. We also decided to drop the source and destination IP addresses and source port because these features are not relevant in a generic intrusion detection system and can be learned by the IDS.

Table 2: Dataset split.

Traffic	Training set	Cross-validation set	Test set
benign	220,316	110,158	110,158
bot	367	183	183
ddos	47,841	23,920	23,920
dos_goldeneye	3,382	1,691	1,691
dos_hulk	79,494	39,747	39,747
dos_slowhttptest	2,433	1,216	1,216
dos_slowloris	2,837	1,418	1,418
ftp_patator	2,001	1,000	1,000
heartbleed	5	2	2
portscan	79,465	39,732	39,732
ssh_patator	1,479	739	739
webattack_bruteforce	680	340	340
webattack_sql_injection	6	3	3
webattack_xss	317	158	158
Total	440,632	220,312	220,312

6.1.2 Creation of Training Set, Cross-validation Set and Test Set

As shown in Table 1, the datasets are highly imbalanced at two different levels. First, the normal traffic (BENIGN) represents ∼80% of the whole traffic and second, some attacks (Heartbleed, Infiltration, WebAttack SQL injection) are represented by a very limited number of instances. This is known to be a challenge for supervised learning. The general idea is to build training and test sets by randomly selecting 50% of each attack for the training set, 25% for the cross-validation set and 25% for the test set, ensuring that each instance is used only once. Since CIC-IDS2017 and LYCOS-IDS2017 do not contain exactly the same number of attacks, we tuned the composition of training and test sets to obtain the same number of instances and allow for a fair comparison. Then, each set is supplemented with randomly selected instances of normal traffic without replacement. The result is a dataset that is balanced in term of normal traffic versus attacks, but still imbalanced in terms of attack types. Only training and cross-validation sets are used during the training phase. Final performance is measured with data from the test set that was never used during the training. The exact composition of the training, cross-validation and test sets is provided in Table 2.

6.1.3 Data Pre-processing

Data pre-processing is essential to prepare the datasets for an efficient training. In particular, the neural network learns best when all features are scaled to the same range. This is especially useful when the inputs are at very different scales. In our implementation, Z-score normalization was selected as it has better results than min-max scaling on the selected dataset. Since some features may contain high values, the relevant information may be compressed by the min-max scaler into a narrow range, negatively impacting the classifier performance. For each feature $\mathcal{F}_j$, the transformation of each x_i value to its standardized value $x_{n,i}$ is given by (1) where $\mu^{(\mathcal{F}_j)}$ and $\sigma^{(\mathcal{F}_j)}$ are respectively the mean and standard deviation values of feature $\mathcal{F}_j$.

$$x_{n,i}^{(\mathcal{F}_j)} = \frac{x_i^{(\mathcal{F}_j)} - \mu^{(\mathcal{F}_j)}}{\sigma^{(\mathcal{F}_j)}} \quad (1)$$

6.2 Algorithms

The performance of the datasets is evaluated on traditional ML algorithms and a simple neural network.

6.2.1 Classical Machine Leaning

The same training, cross-validation and test sets were used for various classical ML algorithms available in the scikit-learn Python library: i) Decision Tree, ii) Random Forest, iii) k-nearest neighbors, iv) Linear Discriminant Analysis, v) Quadratic Discriminant Analysis, vi) Support Vector Machine.

The Decision Tree (DT) classifier is based on CART algorithm which produces a binary tree. The gini criterion is used to measure impurity. Random Forest (RF) contains 100 trees and the classifier also uses gini impurity to split the data. K-Nearest Neighbors (k-NN) is configured to sort data based on the five nearest neighbors using the Euclidean distance. Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA) are classifiers able to generate linear and quadratic boundaries between classes respectively. Due to the size of the training set, it would not be adequate to use a non-linear kernel. Therefore, we used linear Support Vector Machine (SVM).

6.2.2 Neural Network

Multi-layer perceptron (MLP) is a fully connected, feed-forward neural network classifier. Our MLP takes as inputs the normalized values of the selected features and contains two hidden layers of 256 nodes with SELU (Klambauer et al., 2017) activation function. The output layer is composed of 14 nodes and uses a softmax activation function.

The model was implemented in Python, using TensorFlow-2.3 as deep learning framework. The training set is divided into mini-batches of 32 instances. MLP learns classification by adjusting the w weights between the neural network nodes to reduce the cross-entropy loss function $\mathcal{L}(w)$ as defined in (2) where y is the ground truth label and $\hat{y}$ the predicted class. $\mathcal{L}(w)$ is optimized with Adam algorithm.

Three parameters α, β_1 and β_2 described in (Kingma and Ba, 2015) allow to configure this optimizer.

$$\mathcal{L}(w) = -\left[y \cdot \log(\hat{y}) + (1-y) \cdot \log(1-\hat{y})\right] \quad (2)$$

The MLP was trained during 25 epochs. Potential over-fitting of the neural network is controlled by evaluating the cross-validation set. Dropout rate is adjusted so that the difference in performance between the training and cross-validation sets remains small. Hyper-parameters tuning is a tricky task that involves finding the right set of values to get the highest performance. A random search strategy (Bergstra and Bengio, 2012) was applied to tune dropout rate, α, β_1 while keeping the default value $\beta_2 = 0.999$. The search was conducted in two stages. In the first phase, we look for the best parameters in a wide range of values which are refined in a second phase. These ranges are $[0.5-0.01]$ for $dropout\_rate$, $[0.0001-0.002]$ for α and $[0.80-0.99]$ for β_1. A second range of parameters was tested around the best parameters observed during the first phase. The procedure was used twice to tune α and β_1 at the same time, and then to tune the dropout rate. We applied this method for both dataset. The hyper-parameter values are given in (3) and (4).

$$\text{our dataset} \begin{cases} \alpha = 0.0003175219400915 \\ \beta_1 = 0.9158410351985389 \\ dropout\_rate = 0.1185905448 \end{cases} \quad (3)$$

$$\text{cic dataset} \begin{cases} \alpha = 0.0003590920789688 \\ \beta_1 = 0.8662527597085028 \\ dropout\_rate = 0.20886178816 \end{cases} \quad (4)$$

6.3 Evaluation Metrics

Several key insights can be extracted by comparing the predictions and the actual classes. True Positive (TP) is the number of attacks correctly predicted as attacks. True Negative (TN) is the number of normal instances classified as normal traffic while False Positive (FP) and False Negative (FN) are respectively the number of normal instances classified as attacks and the number of attacks predicted as normal traffic. Different metrics can be derived from the information contained in the confusion matrix. True Negative Rate (TNR), Accuracy (Acc), Precision (Prec) and Recall (Rec) are defined in (Maniriho and Ahmad, 2018). The f_1 score is the harmonic mean of Precision and Recall. False Positive Rate (FPR) in (5) is the percentage of benign traffic classified as attacks.

$$FPR = \frac{FP}{TN+FP} \quad (5)$$

The Matthews Correlation Coefficient (MCC) was first introduced by B.W. Matthews (Matthews, 1975) in the context of biomedical research. It is an interesting measure taking into account all elements of the confusion matrix in a correct way for imbalanced dataset. In such a situation, accuracy is not adequate because it may be a high value even if the entire minority class is misclassified. Since intrusion detection datasets are intrinsically imbalanced, MCC is an essential metric for such an application. It is also known as a robust metric when positive and negative cases are of equal importance (Chicco et al., 2021). It provides a value between -1 and $+1$. A perfect prediction corresponds to $MCC = 1$. At the opposite, $MCC = -1$ denotes a total disagreement between the predictions and the actual classes. A random prediction would correspond to $MCC = 0$.

6.4 Results

After training the different classifiers, the test set was used to measure their performance as a 14-class classifier. The resulting confusion matrices were simplified by aggregating all attacks into a single class to obtain a binary classifier. Table 3 details the computed metrics for each dataset.

While classifiers differ in performance with CIC-IDS2017, all but one reach the same level of performance with LYCOS-IDS2017. The difference is particularly important for classifiers that performed worse with CIC-IDS2017. LDA is the algorithm that improved the most but remains the worst. Its accuracy increased from 88.10% to 96.93% while the false positive rate decreased from 22.08% to 6.27%. Resolving the CIC-IDS2017 problems has most likely improved the possibility to linearly separate the classes, but the dataset is still not completely linearly separable. This hypothesis is confirmed by the improved results of SVM using a linear kernel. QDA produces better results than LDA. LDA assumes a covariance matrix that is common to all classes in a dataset. Since QDA performs better than LDA, we can conclude that the assumption of a common covariance matrix is not fulfilled in our dataset. The best results are obtained with the decision tree and the random forest. These algorithms behave like rule-based intrusion detection systems that are known to produce good results. The metrics of neural network are very slightly below the best algorithms. This can be explained by the very limited training phase, with only 25 epochs. We can expect the MLP to get better results with a longer training.

Table 4 presents the training duration and the time needed to infer the complete test set which contains

Table 3: Performance comparison.

Dataset	Algorithm	Accuracy (%)	Precision (%)	Recall (%)	f_1 score (%)	False Positive Rate (%)	MCC
LYCOS-IDS2017	LDA	96.59	94.00	99.54	96.69	6.35	0.9335
	QDA	99.83	99.93	99.73	99.83	0.08	0.9966
	SVM	99.50	99.61	99.40	99.51	0.38	0.9901
	k-NN	99.95	99.91	99.96	99.93	0.10	0.9986
	DT	99.92	99.92	99.92	99.85	0.08	0.9985
	RF	99.96	99.95	99.98	99.96	0.06	0.9996
	MLP	99.72	99.88	99.55	99.72	0.12	0.9943
CIC-IDS2017	LDA	88.10	81.63	98.33	89.21	22.12	0.7785
	QDA	97.72	96.00	99.59	97.76	4.15	0.9550
	SVM	96.98	95.78	98.30	97.02	4.33	0.9400
	k-NN	99.31	98.87	98.75	98.61	1.10	0.9864
	DT	99.88	99.87	99.89	99.88	0.13	0.9976
	RF	99.90	99.90	99.91	99.90	0.10	0.9981
	MLP	99.10	98.44	99.79	99.11	1.58	0.9821

Table 4: Execution time.

Algorithm	Training (s)	Inference (s)
LDA	4.874	0.933
QDA	3.138	2.782
SVM	336.908	0.071
k-NN	not applicable	1,519.312
DT	9.879	0.036
RF	64.731	1.850
MLP	231.748	0.801

220,312 instances.

Among the algorithms achieving the best metrics, we observe that k-NN is the worst in term of inference time. This is explained by the nature of the algorithm. k-NN is a lazy classifier that does not need a training phase but measure the distance between the instance to be classified and all other instances in order to find the majority class of k nearest neighbors. This algorithm is not suitable for big datasets. Clearly, Random Forest requires more time than Decision Tree, both for training and for inference. Since these algorithms perform similarly, there is no need to use RF. The longest training is observed with the neural network. Nevertheless, as more and more computers contain hardware accelerators for neural network, the inference time might be significantly reduced and MLP could remain an interesting alternative to other algorithms.

7 BEYOND CIC-IDS2017

The CIC website contains 19 datasets. CICFlowMeter has been used to generate 5 of them: (1) CIC-IDS2017, (2) CIC-AndMal2017 (Lashkari et al., 2018), (3) CSE-CIC-IDS2018, (4) CIC-InvesAndMal2019 (Taheri et al., 2019) and (5) CIC-DDoS2019 (Sharafaldin et al., 2019). We have shown that the features extracted from the PCAP files for (1) are affected by the problems we have identified. One can expect the same issues to be present in all datasets generated by CICFlowMeter. A complete analysis of the PCAP files for each of these datasets would be an overwhelming task. We preferred to focus on the detection of four artifacts in the CSV files.

- The first artifact corresponds to the presence of a constant value for the bulk features, indicating miscalculation.
- Another marker is the detection of the IP address 8.0.6.4 as explained in Section 3.2.3, indicating that ARP packets are classified as IP packets confirming Ethertype detection problems.
- All flows indicating an IP protocol, source and destination ports equal to 0 but not having an IP address 8.0.6.4 are IP packets that are neither TCP, nor UDP. This situation corresponds to an erroneous protocol detection.
- The fourth artifact corresponds to the presence of a flow having two packets with a FIN flag that are sometimes classified as benign and sometimes as an attack. This indicates inconsistent TCP termination flows.

We found that all datasets containing CSV files generated by CICFlowMeter contain these artifacts. This confirms that CICFlowMeter is the source of the issues. Since the authors of these datasets provided raw data, LycoSTand presents an interesting alternative to extract flow-based features from existing PCAP files.

8 CONCLUSION

Having a reliable and publicly available IDS dataset is an important concern for researchers in this domain. The Canadian Institute for Cybersecurity is a major provider of IDS datasets. We analyzed one of them and identified the key issues. Some features are not calculated correctly, the protocols are partly incorrect, and the way TCP packets are grouped into flows is not suitable for machine learning processing. Next, we proposed LycoSTand as a tool for processing PCAP files and generated a new dataset from the CIC-IDS2017 PCAP files. The tool and dataset are publicly available so that they can be used to replicate and improve our results.

A fair comparison of the original dataset with LYCOS-IDS2017 demonstrated that the corrections made by our tool have a positive impact on all tested machine learning algorithms. Metrics such as accuracy, precision and recall are above 99.5% for all algorithms, with the exception of LDA, which, in any case, improves significantly from 88% to 96%. The best results are obtained with Random Forest which outperforms all other algorithms for all metrics. Finally, we observe that SVM or QDA that do not rank well for CIC-IDS2017 become interesting algorithms for LYCOS-IDS2017. This shows that a corrected dataset may help researchers to reconsider the choice of algorithms in their IDS studies.

The issues we identified in CIC-IDS2017 only exist in CSV files. This shows how important it is to provide raw data along with the flow-based features. Problem markers were found in all five datasets generated with CICFlowMeter. However, they are still interesting because their PCAP files can be processed with LycoSTand to obtain better flow-based datasets. Because most of the publications related to these datasets rely on algorithms processing the CSV files, it is important to make available a corrected version so that findings of research work are not impacted by the erroneous CSV files.

Future work could be considered on the datasets published by CIC and listed in section 7 with the goal to confirm suspected issues. In addition, with the rise of IoT devices, it would be interesting to study how such devices can be protected from network intrusions. We intend to improve LycoSTand execution time, deploy the model on a resource-constrained system and replicate attacks to investigate whether the training on the dataset can detect intrusion attacks launched with a penetration testing toolset. Such research could highlight some limitations of the ML-based IDS solution for embedded system and determine if the current dataset is prone to concept drift.

REFERENCES

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13:281–305.

Canadian Institute for Cybersecurity (2017a). Applications - cicflowmeter (formerly iscxflowmeter). https://www.unb.ca/cic/research/applications.html. Last checked on Nov 27, 2021.

Canadian Institute for Cybersecurity (2017b). Intrusion detection evaluation dataset (cicids2017). https://www.unb.ca/cic/datasets/ids-2017.html. Last checked on Nov 27, 2021.

Chicco, D., Tötsch, N., and Jurman, G. (2021). The matthews correlation coefficient (mcc) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation. *BioData Mining*, 14(1):13.

Draper-Gil., G., Lashkari., A. H., Mamun., M. S. I., and Ghorbani., A. A. (2016). Characterization of encrypted and vpn traffic using time-related features. In *Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP)*, volume 1, pages 407–414. INSTICC, SciTePress.

Engelen, G., Rimmer, V., and Joosen, W. (2021). Troubleshooting an intrusion detection dataset: the cicids2017 case study. In *2021 IEEE Security and Privacy Workshops (SPW)*, pages 7–12.

Gamage, S. and Samarabandu, J. (2020). Deep learning methods in network intrusion detection: A survey and an objective comparison. *Journal of Network and Computer Applications*, 169:102767.

Gharib, A., Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. A. (2016). An Evaluation Framework for Intrusion Detection Dataset. In *International Conference on Information Science and Security (ICISS)*, pages 1–6.

Ho, S., Jufout, S. A., Dajani, K., and Mozumdar, M. (2021). A novel intrusion detection model for detecting known and innovative cyberattacks using convolutional neural network. *IEEE Open Journal of the Computer Society*, 2:14–25.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. *3rd International Conference for Learning Representations*.

Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. (2017). Self-normalizing neural networks. In *Advances in Neural Information Processing Systems*, pages 971–980.

Lashkari, A. H., Gil, G. D., Mamun, M. S. I., and Ghorbani, A. A. (2017). Characterization of tor traffic using time based features. In *Proceedings of the 3rd International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,*, page 253–262. SciTePress.

Lashkari, A. H., Kadir, A. F. A., Taheri, L., and Ghorbani, A. A. (2018). Toward developing a systematic approach to generate benchmark android malware datasets and classification. In *2018 International Carnahan Conference on Security Technology (ICCST)*, pages 1–7.

Lee, W., Stolfo, S. J., and Mok, K. W. (1999). Mining in a data-flow environment: Experience in network intrusion detection. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, pages 114–124, New York, NY, USA. ACM.

Maniriho, P. and Ahmad, T. (2018). Analyzing the Performance of Machine Learning Algorithms in Anomaly Network Intrusion Detection Systems. In *4th International Conference on Science and Technology (ICST)*, pages 1–6.

Maseer, Z. K., Yusof, R., Bahaman, N., Mostafa, S. A., and Foozy, C. F. M. (2021). Benchmarking of machine learning for anomaly based intrusion detection systems in the cicids2017 dataset. *IEEE Access*, 9:22351–22370.

Matthews, B. (1975). Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, 405(2):442–451.

McHugh, J. (2000). Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations As Performed by Lincoln Laboratory. *ACM Trans. Inf. Syst. Secur.*, 3(4):262–294.

Moustafa, N. and Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *Military Communications and Information Systems Conference (MilCIS)*, pages 1–6.

Panigrahi, R. and Borah, S. (2018). A detailed analysis of cicids2017 dataset for designing intrusion detection systems. *International Journal of Engineering & Technology*, 7(3.24):479–482.

Rosay, A., Riou, K., Carlier, F., and Leroux, P. (2021). Multi-layer perceptron for network intrusion detection. *Annals of Telecommunications - annales des télécommunications*.

Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. A. (2018). Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, volume 1, pages 108–116. SciTePress.

Sharafaldin, I., Lashkari, A. H., Hakak, S., and Ghorbani, A. A. (2019). Developing realistic distributed denial of service (ddos) attack dataset and taxonomy. In *International Carnahan Conference on Security Technology (ICCST)*, pages 1–8.

Sperotto, A., Sadre, R., van Vliet, F., and Pras, A. (2009). A labeled data set for flow-based intrusion detection. In Nunzi, G., Scoglio, C., and Li, X., editors, *IP Operations and Management*, pages 39–50, Berlin, Heidelberg. Springer Berlin Heidelberg.

Taheri, L., Kadir, A. F. A., and Lashkari, A. H. (2019). Extensible android malware detection and family classification using network-flows and api-calls. In *International Carnahan Conference on Security Technology (ICCST)*, pages 1–8.

Tavallaee, M., Bagheri, E., Lu, W., and Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. In *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pages 1–6.

Welford, B. (1962). Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420.

Yang, L., Moubayed, A., and Shami, A. (2021). Mth-ids: A multi-tiered hybrid intrusion detection system for internet of vehicles. *IEEE Internet of Things Journal*, pages 1–1.